TESIS

PENJADWALAN PROYEK PENGEMBANGAN SOFTWARE SECARA DINAMIS DAN MULTI PROYEK DENGAN PENDEKATAN HEURISTIK

Oleh :

BAMBANG NURDEWANTO
NRP : 5102 201 011

PROGRAM PASCASARJANA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2006

PENJADWALAN PROYEK PENGEMBANGAN SOFTWARE SECARA DINAMIS DAN MULTI PROYEK DENGAN PENDEKATAN HEURISTIK

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Komputer (M.Kom.)
di
Institut Teknologi Sepuluh Nopember Surabaya

Oleh:

Bambang Nurdewanto
NRP: 5102 201 011

Disetujui oleh Tim Penguji Tesis:

1. **Daniel Oranova Siahaan**
2. **Febriliyan Samopa, S.Kom,M.Kom.**
   NIP. 132 206 858
3. **Ir. Siti Rochimah, MT.**
   NIP. 132 103 631

Dosen Pembimbing:

Rully Soelaiman,S.Kom., M.Kom.
NIP. 132 085 802

Tanggal Ujian: 08 Februari 2006
Periode Wisuda: Maret 2006

Direktur Program Pascasarjana

Prof. Ir. Happy Ratna S., M.Sc., Ph.D.
NIP. 130 541 829

PENJADWALAN PROYEK PENGEMBANGAN SOFTWARE SECARA DINAMIS DAN MULTI PROYEK DENGAN PENDEKATAN HEURISTIK

Nama Mahasiswa : Bambang Nurdewanto
NRP : 5102 201 011
Dosen Pembimbing : Rully Soelaiman, S.Kom, M.Kom

ABSTRAK

Sebelum proyek pengembangan perangkat lunak biasanya dilakukan estimasi : berapa lama dibutuhkan waktu untuk mengerjakan proyek (*schedule*), berapa lama usaha (*effort*) yang diperlukan untuk mengerjakan proyek, dan berapa banyak orang (*staff*) akan dilibatkan dalam pengerjaan proyek. Dengan menggunakan metode COCOMO II suatu perangkat lunak dapat diestimasi kebutuhan *effort* dan *schedule*. Dengan diketahuinya *effort* dan *schedule*, maka dapat dibuat jadwal multi proyek untuk beberapa perangkat lunak yang sedang dikembangkan.

Selama ini estimasi waktu dan penempatan sumber daya secara dinamis dan multi proyek sulit diperhitungkan. Oleh karena itu dicari solusi dengan pendekatan heuristik. Dengan membandingkan beberapa pendekatan heuristik akan dicari solusi yang paling optimal. Bila terdapat perubahan pada penjadwalan, maka semua aktivitas proyek akan dijadwalkan ulang dengan cepat.

Kata Kunci : ***effort, schedule, COCOMO,* multi proyek, heuristik**

Scheduling of Development Project of Software Dynamic and Multi Project by Heuristic Approach

Name of Student	: Bambang Nurdewanto
Identity Number	: 5102 201 011
Advisor Lecturer	: Rully Soelaiman, S.Kom, M.Kom

ABSTRACT

Before development project of software is generally done by estimated: how long a time to do project required; how long the effort that is needed to do project, and how many people would involved in a project workmanship. By using method COCOMO II, software can be estimated effort and schedule requirement. Known of effort and schedule, hence can be scheduled for multi project for a few the software which is being developed.

During the time estimate resource location and time dynamic and multi project difficult to calculate. Therefore search solution by heuristic approach. Compared some approach heuristic will be searched the most optimal solution. When there are changed of scheduling, hence all project activity will be rescheduled swiftly.

Keyword: effort, schedule, COCOMO, multi project, heuristic

KATA PENGANTAR

Dengan mengucap syukur Alhamdullilah, karena anugerahNya yang sangat besar telah diberikan kepada penulis baik itu kesehatan, waktu, hikmat serta ketekunan sehingga dapat menyelesaikan tesis ini.

Rasa terima kasih yang mendalam penulis ucapkan kepada:

1. **Ir. F.X. Arunanto, M.Sc.**, selaku Ketua Program Studi Program Pasca Sarjana Teknik Informatika.
2. **Rully Soelaiman, S.Kom.,M.Kom.,** selaku Dosen Pembimbing Tesis yang telah banyak memotivasi dan membimbing selama mengerjakan tesis.
3. **Daniel Oranova Siahaan, M.Sc., Ph.D, Febriliyan Samopa, S.Kom,M.Kom, Ir. Siti Rochimah, MT** yang telah meluangkan waktu membantu dalam menguji tesis ini.
4. **Istriku, Nurul Aini** yang telah memberikan dukungan,dorongan, dan semangat
5. **Bapak dan Ibu** yang senantiasa mendoakan dan memberi dukungan bagi penulis.
6. **Teman-teman sekalian** yang telah membantu.

Kritik dan saran membangun penulis harapkan untuk pengembangan lebih lanjut dari penelitian ini.

Surabaya, Januari 2006

Penulis

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

Halaman

BAB I

PENDAHULUAN

1.1 Latar Belakang

Sebelum proyek pengembangan perangkat lunak dimulai, seorang perencana proyek (*project planner*) biasanya melakukan estimasi atas tiga hal yang paling menentukan besarnya biaya pengembangan perangkat lunak, yaitu : berapa lama dibutuhkan waktu untuk mengerjakan proyek (*schedule*), berapa lama usaha (*effort*) yang diperlukan untuk mengerjakan proyek, dan berapa banyak orang (*staff*) akan dilibatkan dalam pengerjaan proyek. Selain tiga hal tersebut, perencanaan proyek juga harus memprediksi keperluan perangkat keras dan perangkat lunak pembantu dan juga resiko-resiko yang mungkin terjadi.

Estimasi adalah aktifitas yang penting dalam proyek pengembangan perangkat lunak. Tanpa estimasi tidak akan didapatkan perencanaan dan kontrol yang baik. Kebanyakan industri perangkat lunak tidak melakukan estimasi dengan baik. Terdapat dua kesalahan dalam mengestimasi yang hasilnya sama buruknya. Hasil estimasi yang kurang dari seharusnya dapat menyebabkan kekurangan staff, kualitas berkurang, melebihi waktu jatuh tempo, yang menyebabkan hilangnya kepercayaan dari konsumen. Sedangkan estimasi yang berlebihan dapat menyebabkan biaya yang berlebihan dan pemakaian sumber daya berlebihan dapat menyebabkan proyek lain yang memakai sumber daya yang sama tertunda.

Model estimasi biaya merupakan hal yang sangat penting nilainya dalam setiap proyek pengembangan perangkat lunak karena dapat dijadikan sarana untuk mengkomunikasikan keputusan-keputusan bisnis antara berbagai pihak yang berkepentingan (*stakeholders*) dalam proyek tersebut. Model biaya ini membantu para pengembang perangkat lunak untuk memperkirakan implikasi biaya dan jadwal yang diakibatkan oleh berbagai keputusan proyek pengembangan perangkat lunak.

Model estimasi biaya COCOMO II dapat digunakan untuk berbagai keperluan estimasi seperti proses negosiasi kontrak, analisa perbaikan proses, pembelian perangkat bantu (*tools*), perubahan arsitektur, keputusan mengembangkan sendiri atau membeli perangkat lunak, dan keputusan yang menyangkut *return-on-investment* lainnya dalam proyek yang berkaitan dengan aplikasi perangkat lunak, dengan perhitungan estimasi yang layak dipercaya.

Penjadwalan merupakan titik puncak dari aktivitas perencanaan yang menjadi komponen utama dari manajemen proyek. Selama ini alat bantu penjadwalan proyek yang sering digunakan adalah *Gantt Chart*, CPM (*Critical Path Method)*, dan PERT (*Program Evaluation and Review Technique*). Tapi pada aplikasi-aplikasi tersebut kekurangan yang serius, yaitu mengasumsikan sumber daya yang tidak terbatas untuk ditempatkan pada aktivitas proyek. Pada prakteknya, sumber daya harus dibatasi dan lebih dari satu proyek yang aktif pada saat yang sama.

Selama ini estimasi waktu dan penempatan sumber daya secara dinamis dan multi proyek sulit diperhitungkan. Oleh karena itu dicari solusi dengan pendekatan heuristik. Dengan membandingkan beberapa pendekatan heuristik akan dicari solusi yang paling optimal. Bila terdapat perubahan pada penjadwalan, maka semua aktivitas proyek akan dijadwalkan ulang dengan cepat.

1.2 Perumusan Masalah

Dengan menggunakan metode COCOMO II suatu perangkat lunak dapat diestimasi kebutuhan *effort* dan *schedule*. Dengan diketahuinya *effort* dan *schedule*, maka dapat dibuat jadwal multi proyek untuk beberapa perangkat lunak yang sedang dikembangkan dengan sumber daya yang terbatas secara heuristik.

1.3 Tujuan Penelitian

Tujuan penelitian ini adalah untuk mengestimasi biaya pengembangan perangkat lunak dengan menggunakan metode COCOMO II, kemudian mengkalibrasi pada konstanta A dan C dengan data lokal untuk mendapatkan hasil estimasi biaya pengembangan perangkat lunak yang lebih akurat sesuai dengan kondisi organisasi. Kemudian dibuat jadwal secara multi proyek dengan sumber daya terbatas secara heuristik.

1.4 Batasan Masalah

Penelitian akan dilaksanakan dengan batasan-batasan masalah sebagai berikut :

a. Jumlah perangkat lunak yang digunakan berjumlah empat buah. Jumlah ini didasarkan pada alasan kesediaan dan kelengkapan dokumentasi proyek pengembangan perangkat lunak tersebut
b. Pengukuran besarnya proyek dengan menggunakan *Function Point*, dikarenakan *Function Point* dianggap dapat memberikan hasil yang cukup akurat meskipun data yang diambil masih terbatas
c. Hanya Konstanta A dan C dari model COCOMO II yang dikalibrasi dengan menggunakan metode Logaritma Natural (Ln) dan Bayesian.

d. Metode penjadwalan multi proyek yang digunakan adalah SPT, *Due Date*, NPV, dan *Permutation*

e. Sumber daya yang ditempatkan hanya sumber daya manusia sebagai tenaga *surveyor*, *analist*, *programmer* dan *operator*

f. Perhitungan hari libur pada sumber daya hanya untuk Sabtu dan Minggu saja, hari libur nasional tidak diperhitungkan.

g. Proyek software menggunakan *waterfall* model

1.5 Kontribusi

Penelitian ini memberikan kontribusi antara lain:

- Untuk mengestimasi biaya pengembangan perangkat lunak didalam penelitian ini dangan metode COCOMO II dan melakukan kalibrasi konstanta A dan C dengan data lokal sehingga didapatkan hasil yang lebih akurat.
- Penelitian ini juga mempunyai kontribusi didalam pembuatan jadwal multi proyek untuk beberapa pengembangan perangkat lunak secara bersama-sama dengan metode heuristik

1.6 Sistematika Pembahasan

Tesis ini dibagi dalam beberapa bab sebagai berikut :

BAB I, Pendahuluan, yang dibahas mengenai latar belakang diadakannya penelitian dalam tesis ini, Perumusan Masalah, Tujuan Penelitian, Batasan Masalah, Kontribusi dari penelitian yang dilakukan serta Sistematika Pembahasan.

BAB II, Studi Pustaka berisi tentang cara-cara untuk estimasi biaya pengembangan perangkat lunak dan melakukan pembuatan jadwal multi proyek serta profil data yang digunakan untuk uji coba.

BAB III, Analisa Data dan Desain Perangkat Lunak berisi penentuan data UFP dari data software yang diambil untuk ujicoba. Pada Bab III ini berisi pula desain perangkat lunak yang digunakan untuk estimasi biaya dan pembuatan jadwal multi proyek.

BAB IV, Hasil dan Pembahasan, berisi implementasi program yang digunakan untuk mengestimasi dan membuat jadwal multi proyek pada empat software yang akan diujicoba. Pada Bab IV ini berisi pula tentang kalibrasi untuk menentukan konstanta lokal A dan C agar dalam mengestimasi biaya pengembangan perangkat lunak menjadi lebih akurat dan penerapan metode heuristik agar pada penjadwalan multi proyek menjadi lebih optimal.

Bab V, Penutup, berisi kesimpulan dari penelitian yang telah dilakukan.

BAB II

STUDI PUSTAKA

2.1 Estimasi Proyek Perangkat Lunak

Estimasi adalah aktifitas yang penting dalam proyek pengembangan perangkat lunak. Tanpa estimasi tidak akan didapatkan perencanaan dan kontrol yang baik. Kebanyakan industri perangkat lunak tidak melakukan estimasi dengan baik. Terdapat dua kesalahan dalam mengestimasi yang hasilnya sama buruknya. Hasil estimasi yang kurang dari seharusnya dapat menyebabkan kekurangan staff, kualitas berkurang, melebihi waktu jatuh tempo, yang menyebabkan hilangnya kepercayaan dari konsumen. Sedangkan estimasi yang berlebihan dapat menyebabkan biaya yang berlebihan dan pemakaian sumber daya berlebihan dapat menyebabkan proyek lain yang memakai sumber daya yang sama tertunda.

2.1.1 Langkah-Langkah Estimasi Proyek Perangkat Lunak

Terdapat empat langkah dalam mengestimasi proyek perangkat lunak, yaitu :

a) Estimasi ukuran atau *size* dari produk dalam LOC (*Lines of Code*).

b) Estimasi beban kerja (*effort*) dalam PM (*person-months*) atau Pd (*people-day*)

c) Estimasi jadwal dalam kalender bulanan

d) Estimasi biaya proyek dalam dolar atau mata uang setempat

2.1.1.1 Estimasi Ukuran

Dua cara dalam melakukan estimasi ukuran :

a) Dengan analogi. Dengan cara membandingkan bagian-bagian proyek baru dengan proyek lama yang mirip dan telah diketahui ukurannya. Dapat dihasilkan estimasi ini baik jika dilakukan oleh seorang ahli estimasi yang berpengalaman dan adanya data proyek lama yang mirip dengan proyek yang baru.

b) Dengan menghitung fitur produk dan pendekatan algoritma seperti *function points* untuk mengkonversi ke LOC.

2.1.1.2 Estimasi Effort

Dua cara dalam melakukan estimasi effort :

a) Cara terbaik adalah menggunakan data proyek sebelumnya, dengan asumsi organisasi telah mendokumendasikan hasil proyek sebelumnya yang mirip dengan proyek yang baru. Proyek baru dapat dikerjakan sesuai dengan proyek lama dengan menggunakan metodologi dan sumber daya yang sama.

b) Jika data tersebut tidak ada karena proyek baru yang dikerjakan berbeda, dapat digunakan pendekatan algoritma seperti COCOMO untuk mengkonversi dari estimasi ukuran ke estimasi effort.

2.1.1.3 Estimasi Jadwal

Estimasi ini meliputi jumlah orang yang akan bekerja pada proyek, apa yang akan dikerjakan yang dinyatakan dalam WBS (*Work Breakdowm Structure*), kapan akan mulai bekerja dan kapan selesai pekerjaannya yang dinyatakan dalam kalender jadwal.

2.1.1.4 Estimasi Biaya

Estimasi total biaya tergantung dari bagaimana organisasi mengalokasikan biaya. Biaya dapat dialokasikan untuk masing-masing proyek dan menambahkan

biaya overhead ke biaya tenaga kerja ($ per jam). Sering kali manajer proyek hanya melakukan estimasi biaya tenaga kerja dan biaya tambahan lain tanpa biaya overhead organisasi. Cara yang mudah adalah dengan mengalikan estimasi effort (dalam jam) dengan biaya tenaga kerja ($ per jam). Tapi untuk lebih detailnya menggunakan biaya tenaga kerja sesuai dengan posisinya misalnya programmer, analis, administrasi, dan sebagainya.

2.1.2 Metode Estimasi Biaya Perangkat Lunak

Terdapat enam metode dalam mengestimasi biaya perangkat lunak, yaitu :

a) Model Algoritmik (Parametrik)

b) *Expert Judgment* (berdasarkan para ahli)

c) Top-Down

d) Bottom Up

e) Dengan analogi

f) Price to win estimation

2.1.3 Model Algoritmik (Parametrik)

Model ini menggunakan persamaan matematika yang berdasarkan teori dan data historis. Model ini menggunakan input SLOC (*Size Lines of Code*) dan jumlah fungsi, juga dengan sejumlah penentu biaya lain (*cost driver*). Akurasi model diperbaiki dengan melakukan kalibrasi/penyesuaian dengan lingkungan organisasi. Contoh model ini adalah COCOMO (*Constructive Cost Model*) yang dikembangkan oleh Boehm pada tahun 1981.

Keuntungan model ini adalah : dapat menghasilkan estimasi yang dapat diulang, mudah untuk memodifikasi data input, mudah untuk melakukan kalibrasi formula disesuaikan dengan organisasi. Sedang kerugiannya adalah : model ini tidak

dapat menangani suatu kondisi-kondisi yang merupakan pengecualian dan tidak dapat mengukur beberapa faktor yang merupakan pengalaman.

2.1.4 Expert Judgment

Metode ini menggunakan pengetahuan dan pengalaman praktisi dan melakukan estimasi berdasar pada proyek-proyek dimana tenaga ahli digunakan. Contohnya : WBS.

Keuntungannya bermanfaat jika tidak ditemukan data empiris, dapat menilai dan menentukan perbedaan antara proyek lama dan proyek yang akan dilakukan, dan dapat menilai dampak dari teknologi baru, bahasa dan aplikasi. Kerugiannya estimasi sangat ditentukan pendapat para ahli dan sulit melakukan dokumentasi faktor tenaga ahli.

2.1.5 Top-Down

Metode ini disebut juga model makro. Model ini diturunkan dari produk global dan dipecah menjadi komponen-komponen kecil. Contohnya model Putnam.

Keuntungannya : model ini memerlukan detail proyek minimal, biasanya lebih cepat dan mudah diimplentasikan, dan berfokus pada aktifitasi level sistem. Kerugiannya: komponen level rendah sering terlewatkan, dan tidak ada basis yang terperinci.

2.1.6 Bottom-Up

Metode ini dilakukan dengan cara memperkirakan biaya dari tiap komponen perangkat lunak dan kemudian digabungkan hasilnya sebagai biaya total proyek. Tujuannya adalah membangun estimasi dari gabungan komponen yang kecil yang dikumpulkan dan interaksinya. Contohnya : COCOMO *detailed model.*

Keuntungannya : model ini lebih stabil, lebih terperinci, dan mengijinkan masing-masing komponen perangkat lunak diestimasi. Kerugiannya : dimungkinkan melewatkan biaya sistem level, dan butuh waktu lebih untuk mengimplementasikan.

2.1.7 Dengan Analogi

Metode ini dilakukan dengan cara membandingkan proyek baru dengan proyek lama yang mirip dan diselesaikan dengan aplikasi yang sama. Metode ini dapat dilakukan bila data dari proyek lama telah didokumentasikan dengan baik dan terpola. Metode ini dapat digunakan baik pada sistem maupun tingkatan komponen.

Keuntungan metode ini didasarkan pada data proyek yang aktual. Kerugiannya mustahil dilakukan jika tidak ada proyek pada masa lampau yang dapat diperbandingkan dan tergantung pada seberapa baik proyek sebelumnya didokumentasikan.

2.1.8 Price to Win Estimation

Metode ini hanya memperhitungkan harga sekecil-kecilnya untuk memenangkan kontrak kerja. Keuntungannya adalah biasanya kontrak kerja didapatkan, tetapi kerugiannya adalah waktu dan uang akan habis sebelum pekerjaan diselesaikan.

2.1.9 COCOMO Sebagai Pilihan Terbaik

COCOMO sebagai pilihan terbaik dikarenakan :

a) COCOMO adalah metode yang paling populer dalam estimasi perangkat lunak meskipun seharusnya digunakan lebih dari satu metode.

b) Terbaik dalam menggunakan metoda lain yang berbeda dari COCOMO sehingga estimasi dapat diuji lebih dari satu sudut pandang. Bahkan

perusahaan yang menjual COCOMO merekomendasikan penggunaan lebih dari satu metode.

c) Mudah dilakukan, estimasi kecil bisa dilakukan dengan manual

d) USC mempunyai suatu versi grafis dan gratis yang tersedia untuk didownload

e) Banyak versi komersil yang berbeda tapi berdasar pada COCOMO; mereka menyediakan dukungan dan lebih banyak data, tetapi mahal

2.2 COCOMO II [Boehm, 2000]

Model estimasi biaya pengembangan perangkat lunak yang dikenal dengan nama COCOMO II merupakan salah satu model estimasi biaya algoritmik. Model ini digunakan untuk mengestimasi biaya pengembangan proyek mulai dari tahap awal sampai dengan tahap akhir pengembangan. COCOMO II sebenarnya merupakan suatu hirarki model estimasi biaya sebagai berikut :

- Model komposisi (*Application Composition Model*) atau model purwarupa awal (*Early Prototyping Model*); model ini dipergunakan pada tahap awal pengembangan perangkat lunak yaitu pada saat purwarupa perangkat lunak sedang dikembangkan
- Model rancangan awal (*Early Design Model*); model ini digunakan pada saat kebutuhan user sudah relatif stabil dan arsitektur dasar perangkat lunak sudah dibangun
- Model arsitektur akhir (*Post Architecture Model*); model ini digunakan pada saat proses kontruksi perangkat lunak sedang berlangsung atau pada saat perangkat lunak sudah selesai dikembangkan

Model estimasi biaya COCOMO II dapat digunakan untuk berbagai keperluan estimasi seperti proses negosiasi kontrak, analisa perbaikan proses,

pembelian perangkat bantu (*tools*), perubahan arsitektur, keputusan mengembangkan sendiri atau membeli perangkat lunak, dan keputusan yang menyangkut *return-on-investment* lainnya dalam proyek yang berkaitan dengan aplikasi perangkat lunak, dengan perhitungan estimasi yang layak dipercaya.

Secara umum dikenal dua model dalam COCOMO II yaitu : *Post Architecture* dan *Early Design*. *Post Architecture* adalah model detail proyek yang pernah dibuat untuk dikembangkan kembali dan menopang dari sistem. Sistem telah mempunyai paket arsitektur yang menyediakan informasi detail dari input cost driver dengan estimasi biaya yang lebih akurat. *Early Design* adalah model tingkat tinggi yang digunakan untuk mengeksplorasi alternatif dari arsitektur atau strategi pengembangan.

2.2.1 Persamaan Estimasi Jadwal Nominal

Post Architecture dan Early Design menggunakan bentuk formula yang sama untuk melakukan estimasi dari effort dan waktu kalender dalam mengembangkan perangkat lunak. Formula NS (*Nominal Schedule*) ini diluar dari Cost Driver SCED (*Schedule*). Jumlah effort dalam person months, PM_{NS}, mempunyai rumus :

$$PM_{NS} = A \times Size^{E} \times \prod_{i=1}^{n} EM_i$$
$$\text{where } E = B + 0.01 \times \sum_{j=1}^{5} SF_j \qquad \text{.......................................(1)}$$

Sedangkan untuk mencari jumlah waktu kalender, $TDEV_{NS}$, menggunakan rumus :

$$TDEV_{NS} = C \times (PM_{NS})^{F}$$
$$\text{where } F = D + 0.2 \times 0.01 \times \sum_{j=1}^{5} SF_j$$
$$= D + 0.2 \times (E - B) \qquad \text{.......................................(2)}$$

Pada Post Architecture nilai n pada EM_i adalah 16, dan untuk Early Design adalah 6. Nilai A, B, EM, SF untuk COCOMO II.2000 model Post Architecture diambil dengan kalibrasi parameter aktual dan nilai effort dari 161 proyek yang terdapat pada database COCOMO II. Nilai C dan D untuk persamaan jadwal COCOMO II.2000 juga diambil dengan kalibrasi nilai jadwal aktual dari 161 proyek pada database COCOMO II. Ukuran program dinyatakan dalam KSLOC (*Kilo Source Lines of Code*) atau UFP (*Unadjust Function Point*). Nilai A, B, C, dan D dari COCOMO II.2000 adalah :

A=2.94 B=0.91 C=3.67 D=0.28

2.2.2 Metode Penghitungan Ukuran Besarnya Proyek (Project Sizing)

Estimasi ukuran program yang baik penting untuk mengestimasi model yang baik. Proyek biasanya terdiri dari kode baru dan kode *reuse* dari sumber lain, dengan atau tanpa modifikasi dan kode translasi otomatis. COCOMO II hanya menggunakan ukuran data yang mempengaruhi kode baru dan kode yang dikopi dan dimodifikasi.

2.2.3 Penghitungan Baris Kode Program (SLOC)

Ukuran program dinyatakan dalam ribuan baris dari kode program atau KSLOC (*Kilo Source Lines of Code*). Penghitungan KSLOC ini termasuk juga program pendukung seperti test drivers, program review, rencana pengetesan, dan dokumentasi dan sebagainya. Tujuannya untuk mengukur jumlah pekerjaan intelektual dari pengembangan program. Penentuan SLOC ini sulit karena perbedaan konsep dalam penghitungan program eksekusi dan data untuk bahasa pemrograman yang berbeda. Kesulitan muncul pada penentuan ukuran yang konsisten untuk bahasa pemrograman yang berbeda.

2.2.4 Penghitungan Unadjusted Function Points (UFP)

Pendekatan estimasi biaya FP (*function point*) didasarkan dari jumlah fungsionalitas dari proyek software dan kumpulan faktor proyek individu. Function Point baik digunakan sebagai estimasi karena berdasar pada informasi yang dibutuhkan sebelumnya dalam *life cycle* proyek.

Function point mengukur proyek software dengan penghitungan informasi yang berhubungan dengan data eksternal atau input kontrol, ouput, atau tipe file. Lima tipe dari function point sebagai berikut :

a) EI (*External Input*), yaitu menghitung setiap tipe data atau input kontrol user yang masuk ke batas eksternal sistem software yang diukur.

b) EO (*External Output*), yaitu menghitung setiap tipe data atau output kontrol user yang masuk ke batas eksternal sistem software yang diukur.

c) ILF (*Internal Logical File*), yaitu menghitung setiap grup logik data atau kontrol user dalam sistem software sebagai tipe file internal logik. Didalamnya termasuk setiap file logik yang dihasilkan, digunakan, atau dipelihara sistem software.

d) EIF (*External Interface Files*), yaitu file yang dilewatkan atau untuk pemakaian bersama antar software sebagai tipe file penghubung eksternal.

e) EQ (*External Inquiry*), yaitu menghitung setiap kombinasi input dan output dimana input menghasilkan output secara langsung sebagai tipe *Inquiry* eksternal.

Prosedur penghitungan function point pada COCOMO II sebagai berikut :

a) Tentukan tipe function point. Penghitungan ini dilakukan oleh orang teknik berdasarkan dokumen requirement dan desain software.

b) Tentukan level kompleksitasnya. Tentukan level setiap function point dari *low, average*, dan *high* berdasarkan jumlah elemen data dan jumlah tipe file yang digunakannya

c) Hitung ukuran (*Weight*) kompleksitasnya dengan menggunakan tabel 2.1

d) Hitung UFP dengan menambahkan semua perhitungan tersebut.

e) Kemudian konversikan UFP ke SLOC sesuai dengan bahasa pemrograman yang akan diimplementasikan seperti C, C++, atau Pascal.

Tabel 2.1 Kompleksitas UFP

For Internal Logical Files and External Interface Files

	Data Elements		
Record Elements	1 - 19	20 - 50	51+
1	Low	Low	Avg.
2 - 5	Low	Avg.	High
6+	Avg.	High	High

For External Output and External Inquiry

	Data Elements		
File Types	1 - 5	6 - 19	20+
0 or 1	Low	Low	Avg.
2 - 3	Low	Avg.	High
4+	Avg.	High	High

For External Input

	Data Elements		
File Types	1 - 4	5 - 15	16+
0 or 1	Low	Low	Avg.
2 - 3	Low	Avg.	High
3+	Avg.	High	High

	Complexity-Weight		
Function Type	Low	Average	High
Internal Logical Files	7	10	15
External Interfaces Files	5	7	10
External Inputs	3	4	6
External Outputs	4	5	7
External Inquiries	3	4	6

2.2.5 Model *Reuse*

COCOMO tidak hanya dapat melakukan estimasi biaya dan jadwal untuk pengembangan software yang dimulai dari awal, tapi juga dapat melakukan estimasi biaya dan jadwal untuk produk yang dibuat dengan kode program telah ada. Adaptasi perlu dilakukan untuk menghitung KSLOC, yaitu dengan persamaan :

$$\text{Equivalent KSLOC} = \text{Adapted KSLOC} \times \left(1 - \frac{\text{AT}}{100}\right) \times \text{AAM}$$

$$\text{where AAM} = \begin{cases} \dfrac{[\text{AA} + \text{AAF}(1 + (0.02 \times \text{SU} \times \text{UNFM}))]}{100}, & \text{for AAF} \leq 50 \\ \dfrac{[\text{AA} + \text{AAF} + (\text{SU} \times \text{UNFM})]}{100}, & \text{for AAF} > 50 \end{cases}$$

$$\text{AAF} = (0.4 \times \text{DM}) + (0.3 \times \text{CM}) + (0.3 \times \text{IM}) \quad \ldots (3)$$

- ASLOC (*Adapted Source Lines of Code*) adalah jumlah dari baris kode program yang diadaptasi dari software yang telah ada untuk mengembangkan produk baru
- DM (*Design Modification*) adalah persentase dari desain software yang diadaptasi yang menerima modifikasi dari produk baru
- CM (*Code Modification*) adalah persentase dari kode software yang diadaptasi yang menerima modifikasi dari produk baru
- IM (*Integration Requirement for Modified Software*) adalah persentase dari effort untuk melakukan integrasi dan tes dalam penyatuan dengan produk baru.
- SU (*Software Understanding*) adalah persentase dari pemahaman terhadap software yang diadaptasi
- AA (*Assessment and Assimilation*) adalah persentase dari tingkat penyatuan dari software yang diadaptasi

- UNFM (*Unfamiliarity with Software*) adalah tingkat ketidakpahaman programer terhadap software yang diadaptasi
- Jika tidak ada DM atau CM (komponen yang belum dimodifikasi) maka SU tidak dibutuhkan, atau jika DM atau CM nilainya 0 maka SU juga nilainya 0.

2.2.6 REVL (Requirement Evolution and Volatility)

COCOMO menggunakan faktor REVL untuk menyesuaikan ukuran efektif produk yang disebabkan oleh kebutuhan evolusi seperti evolusi misi, interface user, atau upgrade teknologi. Penggunaan REVL dihitung dengan formula sebagai berikut:

$$Size = \left(1 + \frac{REVL}{100}\right) \times Size_D \quad \text{.............................(4)}$$

$Size_D$ adalah ukuran ekivalen *reuse* dari software yang dikirimkan.

2.2.7 Kode Translasi Otomatis

Model *reuse* COCOMO II membutuhkan tambahan untuk estimasi biaya software yang rekayasa dan dikonversi. Pendekatan COCOMO II dalam estimasi rekayasa dan konversi menggunakan faktor tambahan, yaitu AT (*Automatic Translation*), persentase dari kode yang direkayasa dengan translasi otomatis. Berdasarkan analisa data proyek sebelumnya nilai dasar dari translasi otomatis adalah 2400 tiap orang tiap bulan. Nilai ini bervariasi untuk teknologi yang berbeda, maka COCOMO II menggunakan faktor lain yaitu ATPROD. Formulanya adalah sebagai berikut :

$$PM_{Auto} = \frac{Adapted\,SLOC \times \left(AT/100\right)}{ATPROD} \quad \text{........................(5)}$$

2.2.8 Estimasi Effort

Effort dalam COCOMO II dinyatakan dalam PM (*person-months*). PM adalah jumlah waktu dari satu orang yang bekerja di proyek pengembangan software dalam satu bulan. COCOMO II menggunakan faktor penyesuaian dengan nilai 152 jam/PM. Jumlah ini tidak termasuk hari libur dan cuti. Nilai PM lain dengan waktu untuk menyelesaikan proyek atau yang disebut TDEV (*Time to Develop*). Persamaan estimasi effort COCOMO sebagai berikut :

$$PM = A \times Size^{E} \times \prod_{i=1}^{n} EM_i$$

where A = 2.94 (for COCOMO II.2000) ..... (6)

Input yang dibutuhkan adalah : ukuran (*size*) dari software, nilai konstan A, nilai eksponensial E, dan jumlah nilai dari EM (*Effort Multiplier*).

Ukuran software dinyatakan dalam KSLOC, diperoleh dari jumlah perhitungan SLOC atau estimasi UFP konversikan ke SLOC, kemudian dibagi dengan seribu. Nilai konstan A adalah nilai awal dari model yang dikalibrasi dengan database proyek yang menyatakan rata-rata produksitas global, dan dapat dikalibrasi dengan data lokal untuk memperbaiki keakuratan model.

Cost driver digunakan untuk menangkap karakteristik pengembangan software yang mempengaruhi effort untuk menyelesaikan proyek. Cost driver ini memiliki tingkat dari sangat rendah (*Very Low*) sampai ekstra tinggi (*Extra High*). Setiap level memiliki nilai yang disebut EM (*Effort Multiplier*). Untuk melakukan perhitungan estimasi biaya pengembangan perangkat lunak, model COCOMO II menggunakan beberapa *Scale Factors* dan *Cost Drivers*.

Selain nilai Effort (E) dicari juga nilai estimasi optimistic dan pessimistic dari E dengan persentase sesuai dengan tabel 2.2.

Tabel 2.2 Persentase Estimasi Optimistic dan Pessimistic

Model	Optimistic Estimate	Pessimistic Estimate
Application Composition	0.50 E	2.0 E
Early Design	0.67 E	1.5 E
Post-Architecture	0.80 E	1.25 E

2.2.9 Faktor Skala (*Scale Factors*)

Nilai eksponen dari persamaan (6) merupakan gabungan dari lima faktor skala (SF) yang menghitung skala ekonomis relatif untuk proyek software yang berbeda ukuran. Jika E<1 maka skala ekonomisnya menguntungkan. Jika ukuran produk berlipat dua, maka effort produk kurang dari dua kalinya. Jika E=1, maka skala ekonomisnya seimbang. Model ini biasanya untuk proyek kecil. Jika E>1, maka skala ekonomisnya merugikan. Biasanya disebabkan dua faktor utama, yaitu overhead komunikasi interpersonal dan overhead integrasi sistem yang besar.

Setiap faktor skala mempunyai tingkat dari sangat rendah sampai ekstra tinggi, dan setiap tingkat mempunyai nilai yang disebut dengan SF (*Scale Factor*). SF digunakan dalam formula untuk menghitung E sebagai berikut :

$$E = B + 0.01 \times \sum_{j=1}^{5} SF_j$$

where B = 0.91 (for COCOMO II.2000) ..... (7)

Berikut ini ada 5 scale factors yang diidentifikasi pada COCOMO.

Tabel 2.3 *Scale Factors* untuk Model COCOMO II

Scale Factors	Very Low	Low	Nominal	High	Very High	Extra High
PREC	thoroughly unprecedented	largely unprecedented	somewhat unprecedented	generally familiar	largely familiar	thoroughly familiar
FLEX	rigorous	occasional relaxation	some relaxation	general conformity	some conformity	general goals
RESL	little (20%)	some (40%)	often (60%)	generally (75%)	mostly (90%)	full (100%)
TEAM	very difficult interactions	some difficult interactions	basically cooperative interactions	largely cooperative	highly cooperative	seamless interactions
PMAT	The estimated Equivalent Process Maturity Level (EPML) or					
	SW-CMM Level 1 Lower	SW-CMM Level 1 Upper	SW-CMM Level 2	SW-CMM Level 3	SW-CMM Level 4	SW-CMM Level 5

Secara lebih lengkap akan dijelaskan berikut ini.

a. PREC (*Precedentedness*)

PREC digunakan untuk mengukur tingkat kemiripan dari sebuah perangkat lunak yang akan dikembangkan. Apabila sebuah perangkat lunak yang akan dikembangkan serupa dengan beberapa perangkat lunak yang pernah dikembangkan sebelumnya maka tingkat kemiripan dari perangkat lunak yang akan dikembangkan adalah tinggi.

b. FLEX (*Development Flexibility*)

FLEX dipakai untuk mengukur tingkat fleksibilitas dari software yang akan dikembangkan untuk disesuaikan dengan *requirement* yang sudah ditentukan.

c. RESL (*Architecture / Risk Resolution*)

RESL digunakan untuk mengukur *Design Thoroughness* dan *Risk Elimination* oleh *Product Design Review* (PDR) dari perangkat lunak yang dikembangkan.

d. TEAM (*Team Cohesion*)

TEAM digunakan untuk mengukur tingkat kesulitan dalam membangun tim proyek yang kuat dalam menyatukan pihak-pihak yang berkepentingan

(*stakeholders*), seperti : pengguna, pelanggan, pemelihara, dan lainnya. Kesulitan muncul bila ada perbedaan budaya masing-masing *stakeholders*, termasuk menyatukan perbedaan tujuan dan kurangnya pengalaman *stakeholders* dalam bekerja sama secara tim.

e. PMAT (*Process Maturity*)

PMAT digunakan untuk mengukur tingkat kematangan sebuah proses teknologi informasi di suatu organisasi berdasarkan kriteria yang ditetapkan oleh CMM. **CMM** *(Capability Maturity Model)* adalah suatu prinsip dan praktek yang mendasari kematangan (*maturity*) suatu proses pengembangan software. Tingkat kematangan suatu proses mencerminkan tingkat kemajuan disiplin yang diperlukan untuk perbaikan yang terus menerus. Fokus dari CMM adalah mengidentifikasi area proses yang utama (*key process area*). Area Proses utama menyatakan kelompok aktivitas yang berhubungan dan apabila dilaksanakan secara bersama akan memungkinkan tercapainya sejumlah tujuan yang dianggap penting. Berdasar penilaian yang diperoleh untuk setiap maka tingkat kematangan suatu organisasi dalam proses pengembangan software dapat dikelompokkan seperti pada berikut ini.

Tabel 2.4 Level Kematangan CMM

Level CMM	**Penjelasan Level Kematangan Software CMM**
1. Initial	Proses software berkarakter ad hoc, dan malah terkadang kacau. Sedikit dari proses terdefinisikan dan berhasil bergantung pada usaha individu yang heroik.
2. Repeatable	Proses manajemen proyek dasar dibuat untuk menyusun biaya, jadwal, dan fungsi-fungsi. Disiplin proses yang diperlukan telah ada untuk diulangi keberhasilan proyek terdahulunya.

Tabel 2.4a Lanjutan Tabel 2.4 Level Kematangan CMM

3. Defined	Manajemen dan aktifitas rekayasa proses software didokumentasikan, distandarisasikan, dan diintegrasikan dalam proses software standar. Semua proyek menggunakan persetujuan, penyesuaian versi dai proses software standar untuk pengembangan dan pemeliharaan software
4. Managed	Pengukuran detail proses software dan kualitas produk dikumpulkan. Produk dan proses software secara kuantitas dimengerti dan dikontrol
5. Optimizing	Adanya perbaikan proses terus-menerus dengan umpan balik kuantitas dari proses dan ide inovatif dan teknologi

2.2.10 Effort Multiplier atau Cost Driver

Cost Driver digunakan untuk menangkap karakteristik proyek pengembangan software yang berpengaruh pada besarnya effort untuk menyelesaikan proyek.

2.2.11 Effort Multiplier pada Post Architecture

Terdapat 17 EM (*Effort Multiplier*) pada post architecture yang digunakan COCOMO II untuk menghitung effort nominal, PM, antara lain :

a) Faktor produk, menghitung variasi dalam effort yang dibutuhkan yang dipengaruhi oleh karakter dari produk. Produk yang komplek mempunyai kebutuhan kelayakan yang tinggi atau bekerja dengan database besar sehingga butuh effort lebih untuk menyelesaikan. Terdapat lima faktor produk yang berpengaruh besar, yaitu :

   - RELY (*Requirement software Reliability*), yaitu ukuran dari tingkat dimana software dapat berjalan sesuai dengan yang diharapkan dalam jangka waktu tertentu. Jika efek kegagalan software hanya berupa ketidaknyamanan maka

nilai RELY sangat rendah (*Very Low*). Jika kegagalannya dapat menyebabkan resiko kematian, maka RELY bernilai sangat tinggi (*Very High*).

Tabel 2.5 RELY Cost Driver

RELY Descriptors:	slight inconven-ience	low, easily recoverable losses	moderate, easily recoverable losses	high financial loss	risk to human life	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	0.82	0.92	1.00	1.10	1.26	n/a

- DATA (*Database Size*), yaitu menghitung pengaruh adanya tes data dalam jumlah yang besar pada effort pengembangan perangkat lunak. Usaha yang dikeluarkan untuk mempersiapkan tes data akan mempengaruhi effort proyek secara keseluruhan. Penilaian ditentukan dengan cara menghitung D/P yaitu perbandingan besarnya tes data (satuan *byte*) dengan besarnya SLOC (*Source Line of Code*) dalam perangkat lunak.

Tabel 2.6 DATA Cost Driver

DATA* Descriptors		Testing DB bytes/Pgm SLOC < 10	10 ≤ D/P < 100	100 ≤ D/P < 1000	D/P ≥ 1000	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	n/a	0.90	1.00	1.14	1.28	n/a

- RUSE (*Developed for Reusability*), yaitu mengukur effort tambahan yang dibutuhkan untuk menggabungkan komponen yang bisa digunakan lagi (*reuse*) dengan proyek yang akan dikembangkan. Tambahan usaha dikeluarkan untuk membuat agar desain perangkat lunak sifatnya umum (*generic*), dokumentasi yang lengkap dan rinci, dan untuk melakukan testing yang menyeluruh (*extensive*) untuk memastikan agar komponen perangkat

luna bisa digunakan untuk perangkat lunak lainnya. Termasuk *across project* bila komponen digunakan lagi pada modul lain dalam perangkat lunak yang sama. Termasuk *across program* bila komponen digunakan lagi pada perangkat lunak yang lainnya tapi masih dalam organisasi yang sama. Termasuk *across product line* bila komponen digunakan lagi pada organisasi lain. Termasuk *across multiple product lines* bila komponen dipakai lintas perangkat lunak seperti perangkat lunak keuangan, penjualan, dan pemasaran.

Tabel 2.7 RUSE Cost Driver

RUSE Descriptors:		none	across project	across program	across product line	across multiple product lines
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	n/a	0.95	1.00	1.07	1.15	1.24

- DOCU (*Documentation Match to Life-Cycle Needs*), yaitu mengukur tingkat dokumentasi sesuai dengan siklus pengembangan perangkat lunak. Bila dokomentasi dibuat secara menyeluruh dan lengkap sesuai dengan siklus perangkat lunak, maka diberi nilai Sangat Tinggi (*Very High*).

Tabel 2.8 DOCU Cost Driver

DOCU Descriptors:	Many life-cycle needs uncovered	Some life-cycle needs uncovered.	Right-sized to life-cycle needs	Excessive for life-cycle needs	Very excessive for life-cycle needs	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	0.81	0.91	1.00	1.11	1.23	n/a

- CPLX (*Product Complexity*), yaitu mengukur kompleksitas yang dibagi menjadi 5 bidang, yaitu : operasi kontrol, operasi komputasi, operasi peralatan, operasi manajemen data, dan operasi manajemen interface user.

Tabel 2.9 Komponen Level CPLX

	Control Operations	Computational Operations	Device-dependent Operations	Data Management Operations	User Interface Management Operations
Very Low	Straight-line code with a few non-nested structured programming operators: DOs, CASEs, IF-THEN-ELSEs. Simple module composition via procedure calls or simple scripts.	Evaluation of simple expressions: e.g., A=B+C*(D-E)	Simple read, write statements with simple formats.	Simple arrays in main memory. Simple COTS-DB queries, updates.	Simple input forms, report generators.
Low	Straightforward nesting of structured programming operators. Mostly simple predicates	Evaluation of moderate-level expressions: e.g., D=SQRT(B**2-4.*A*C)	No cognizance needed of particular processor or I/O device characteristics. I/O done at GET/PUT level.	Single file subsetting with no data structure changes, no edits, no intermediate files. Moderately complex COTS-DB queries, updates.	Use of simple graphic user interface (GUI) builders.
Nominal	Mostly simple nesting. Some intermodule control. Decision tables. Simple callbacks or message passing, including middleware-supported distributed processing	Use of standard math and statistical routines. Basic matrix/vector operations.	I/O processing includes device selection, status checking and error processing.	Multi-file input and single file output. Simple structural changes, simple edits. Complex COTS-DB queries, updates.	Simple use of widget set.

Tabel 2.9a Lanjutan Table 2.9 Komponen Level CPLX

	Control Operations	Computational Operations	Device-dependent Operations	Data Management Operations	User Interface Management Operations
High	Highly nested structured programming operators with many compound predicates. Queue and stack control. Homogeneous, distributed processing. Single processor soft real-time control.	Basic numerical analysis: multivariate interpolation, ordinary differential equations. Basic truncation, round-off concerns.	Operations at physical I/O level (physical storage address translations; seeks, reads, etc.). Optimized I/O overlap.	Simple triggers activated by data stream contents. Complex data restructuring.	Widget set development and extension. Simple voice I/O, multimedia.
Very High	Reentrant and recursive coding. Fixed-priority interrupt handling. Task synchronization, complex callbacks, heterogeneous distributed processing. Single-processor hard real-time control.	Difficult but structured numerical analysis: near-singular matrix equations, partial differential equations. Simple parallelization.	Routines for interrupt diagnosis, servicing, masking. Communication line handling. Performance-intensive embedded systems.	Distributed database coordination. Complex triggers. Search optimization.	Moderately complex 2D/3D, dynamic graphics, multimedia.
Extra High	Multiple resource scheduling with dynamically changing priorities. Microcode-level control. Distributed hard real-time control.	Difficult and unstructured numerical analysis: highly accurate analysis of noisy, stochastic data. Complex parallelization.	Device timing-dependent coding, micro-programmed operations. Performance-critical embedded systems.	Highly coupled, dynamic relational and object structures. Natural language data management.	Complex multimedia, virtual reality, natural language interface.

b) Faktor Platform, yaitu memperhitungkan kompleksitas perangkat keras dan infrastruktur perangkat lunak. Faktor yang mempengaruhinya antara lain :

- TIME (*Execution Time Constraint*), yaitu mengukur batas waktu eksekusi sistem software. Tingkatannya menyatakan persentase waktu eksekusi yang diharapkan yang digunakan sistem atau sub sistem mengkonsumsi sumber daya waktu eksekusi.

Tabel 2.10 TIME Cost Driver

TIME Descriptors:			≤ 50% use of available execution time	70% use of available execution time	85% use of available execution time	95% use of available execution time
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	n/a	n/a	1.00	1.11	1.29	1.63

- STOR (*Main Storage Constraint*), yaitu menunjukkan tingkat batas tempat penyimpanan utama yang digunakan sistem atau sub sistem software.

Tabel 2.11 STOR Cost Driver

STOR Descriptors:			≤ 50% use of available storage	70% use of available storage	85% use of available storage	95% use of available storage
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	n/a	n/a	1.00	1.05	1.17	1.46

- PVOL (*Platform Volatility*), yaitu mengukur perubahan platform software. Platform bisa meliputi hardware, sistem operasi, sistem jaringan, kompiler, asembler, dan semua peralatan untuk membuat software. Nilai yang diberikan mulai dari rendah (*Low*) bila perubahan besar platform dilakukan 12 bulan sekali, sampai ke nilai Sangat Tinggi (*Very High*) bila perubahan besar dilakukan setiap dua minggu sekali.

Tabel 2.12 PVOL Cost Driver

PVOL Descriptors:		Major change every 12 mo.; Minor change every 1 mo.	Major: 6 mo.; Minor: 2 wk.	Major: 2 mo.;Minor: 1 wk.	Major: 2 wk.;Minor: 2 days	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	n/a	0.87	1.00	1.15	1.30	n/a

c) Faktor personal, setelah ukuran produk, faktor orang mempunyai pengaruh yang kuat untuk menentukan jumlah effort yang dibutuhkan mengembangkan suatu

produk software. Faktor ini yang dinilai adalah kemampuan dan pengalaman tim. Faktor yang mempengaruhi yaitu :

- ACAP (*Analist Capability*), yaitu mengukur tingkat kemampuan analis untuk melakukan analisis, tingkat efisiensi dan kedalaman analisisnya, serta kemampuan untuk berkomunikasi dan bekerja sama dalam tim. Penilaian tidak memperhatikan tingkat pengalaman analis tersebut.

Tabel 2.13 ACAP Cost Driver

ACAP Descriptors:	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.42	1.19	1.00	0.85	0.71	n/a

- PCAP (*Programmer Capability*), yaitu mengukur kemampuan programer dalam efisiensi dan ketelitiannya, kemampuan dalam berkomunikasi dan bekerja sama. Penilaian tidak memperhatikan tingkat pengalaman programer tersebut.

Tabel 2.14 PCAP Cost Driver

PCAP Descriptors	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.34	1.15	1.00	0.88	0.76	n/a

- PCON (*Personnel Continuity*), yaitu mengukur tingkat pergantian atau mutasi personal pada tiap tahunnya.

Tabel 2.15 PCON Cost Driver

PCON Descriptors:	48% / year	24% / year	12% / year	6% / year	3% / year	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.29	1.12	1.00	0.90	0.81	

- APEX (*Application Experience*), yaitu mengukur tingkat pengalaman tim proyek dalam mengembangkan sistem atau sub sistem software.

Tabel 2.16 APEX Cost Driver

APEX Descriptors:	≤ 2 months	6 months	1 year	3 years	6 years	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.22	1.10	1.00	0.88	0.81	n/a

- PLEX (*Platform Experience*), yaitu mengukur tingkat pengalaman tim proyek dalam mempergunakan platform, termasuk interface grafik, database, dan jaringan.

Tabel 2.17 PLEX Cost Driver

PLEX Descriptors:	≤ 2 months	6 months	1 year	3 years	6 year	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.19	1.09	1.00	0.91	0.85	n/a

- LTEX (*Language and Tool Experience*), yaitu mengukur tingkat pengalaman tim proyek menggunakan bahasa pemrograman dan software pembantu dalam mengembangkan sistem atau sub sistem software. Software pembantu ini digunakan pada saat analisa kebutuhan, desain, analisis, manajemen konfigurasi, dokumentasi, manajemen perpustakaan, format dan bentuk program, pengecekkan konsistensi, perencanaan dan kontrol.

Tabel 2.18 LTEX Cost Driver

LTEX Descriptors:	≤ 2 months	6 months	1 year	3 years	6 year	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.20	1.09	1.00	0.91	0.84	

d) Faktor proyek, yaitu menghitung pengaruh dalam estimasi effort seperti penggunaan software pembantu modern, lokasi tim pengembang, dan percepatan jadwal proyek. Faktor yang mempengaruhi sebagai berikut :

- TOOL (*Use of Software Tools*), yaitu mengukur tingkat software pembantu dalam kemampuan, kematangan, dan integrasinya.

Tabel 2.19 TOOL Cost Driver

TOOL Descriptors	edit, code, debug	simple, frontend, backend CASE, little integration	basic life-cycle tools, moderately integrated	strong, mature life-cycle tools, moderately integrated	strong, mature, proactive life-cycle tools, well integrated with processes, methods, reuse	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.17	1.09	1.00	0.90	0.78	n/a

- SITE (*Multisite Development*), yaitu mengukur pengaruh dari penempatan lokasi tim pengembang dan dukungan komunikasinya.

Tabel 2.20 SITE Cost Driver

SITE: Collocation Descriptors:	Inter-national	Multi-city and Multi-company	Multi-city or Multi-company	Same city or metro. area	Same building or complex	Fully collocated
SITE: Communications Descriptors:	Some phone, mail	Individual phone, FAX	Narrow band email	Wideband electronic communicat ion.	Wideband elect. comm., occasional video conf.	Interactive multimedia
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1.22	1.09	1.00	0.93	0.86	0.80

- SCED (*Required Development Schedule*), yaitu mengukur batas percepatan jadwal tim proyek dalam pengembangan software. SCED pada estimasi waktu, TDEV dihitung dengan cara yang berbeda.

Tabel 2.21 SCED Cost Driver

SCED Descriptors	75% of nominal	85% of nominal	100% of nominal	130% of nominal	160% of nominal	
Rating Level	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multiplier	1.43	1.14	1.00	1.00	1.00	n/a

2.2.12 Effort Multiplier pada Early Design

Model ini digunakan dalam langkah awal proyek software, yaitu jika hanya sedikit diketahui ukuran produk yang dikembangkan, karakter platform, karakter personel yang terlibat, atau spesifikasi terperinci proses yang digunakan. Model ini menggunakan pendekatan dengan cara mengkombinasi dan mentransfer cost driver dan level post architecture.

Tabel 2.22 Cost Driver Early Design

Early Design Cost Driver	Counterpart Combined Post-Architecture Cost Drivers
PERS	ACAP, PCAP, PCON
RCPX	RELY, DATA, CPLX, DOCU
RUSE	RUSE
PDIF	TIME, STOR, PVOL
PREX	APEX, PLEX, LTEX
FCIL	TOOL, SITE
SCED	SCED

2.2.13 Estimasi Effort untuk Multi Modul

Biasanya sistem software terdiri dari sub sistem atau komponen lebih dari satu. Dengan hanya menjumlahkan estimasi tiap komponen dapat menghilangkan

effort integrasi dari komponen. Jadi COCOMO II menggunakan metode untuk penggabungan sejumlah n modul sebagai berikut :

a) Jumlahkan semua ukuran komponen dalam $Size_{Aggregated}$

$$Size_{Aggregate} = \sum_{i=1}^{n} Size_i \quad \text{.......................................(8)}$$

b) Aplikasikan faktor skala SCED dalam PM_{Basic}

$$PM_{Basic} = A \times (Size_{Aggregate})^E \times SCED \quad \text{...........................(9)}$$

c) Tentukan tiap effort dasar komponen, $PM_{Basic(i)}$ dengan membagi dengan adil effort dasar untuk tiap komponen berdasarkan kontribusinya ke $Size_{Aggregated}$

$$PM_{Basic(i)} = PM_{Basic} \times \left(\frac{Size_i}{Size_{Aggregate}} \right) \quad \text{...........................(10)}$$

d) Aplikasikan Cost Driver EM (kecuali SCED) ke tiap effort dasar komponen

$$PM_i = PM_{Basic(i)} \times \sum_{j=1}^{16} EM_j \quad \text{..................................(11)}$$

e) Jumlahkan tiap effort dasar komponen dalam $PM_{Aggregate}$ untuk PM proyek total

$$PM_{Aggregate} = \sum_{i=1}^{n} PM_i \quad \text{......................................(12)}$$

2.2.14 Estimasi Jadwal

Persamaan jadwal untuk COCOMO II adalah :

$$TDEV = [C \times (PM_{NS})^{(D+0.2 \times (E-B))}] \times \frac{SCED\%}{100}$$
$$\text{where } C = 3.67, D = 0.28, B = 0.91 \quad \text{..............(13)}$$

Konstanta C adalah koefisien yang dapat dikalibrasi. PM_{NS} adalah PM tanpa effort multiplier SCED. Konstanta D adalah skala eksponen yang dapat dikalibrasi. E adalah skala effort yang didapat dari penjumlahan skala effort. B adalah eksponen faktor skala yang dapat dikalibrasi. SCED% adalah persentase percepatan dalam effort multiplier SCED. TDEV *(Time to Develop)* adalah waktu kalender dalam bulan untuk mengembangkan software.

2.2.15 Kalibrasi Model ke Lingkungan Lokal

Penelitian terdahulu menyebutkan bahwa COCOMO II akan mendapatkan keakuratan secara signifikan bila mengkalibrasi semua parameter pada model dengan data sendiri yang ada di organisasi. Penelitian menyebutkan bahwa semua dapat dilakukan dengan mengkalibrasi konstanta A yang ada dalam persamaan estimasi effort. Cara ini cukup sederhana dan dapat dilakukan dengan kalkulator, spreadsheet, atau tool statistik regresi. Tujuan dari kalibrasi adalah menghitung distribusi produktifitas dan aktifitas dalam lingkungan lokal pengembangan software.

Kalibrasi pada lingkungan lokal terdiri dari penyesuaian konstanta A pada model. Ada lebih dari satu metode dalam mengkalibrasi A. Teknik yang digunakan di sini menggunakan log natural.

$$PM_{NS} = A \times Size^{E} \times \prod_{i=1}^{n} EM_i \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (14)$$

Data yang dibutuhkan adalah effort aktual (PM_{actual}) yang dikeluarkan antara akhir analisa requirement sampai dengan akhir dari tes dan integrasi software. Nilai dari hasil akhir ukuran produk, faktor skala, dan cost driver juga dibutuhkan. Estimasi tanpa penyesuaian (*Unadjusted Estimate*) dibuat dengan persamaan estimasi tanpa konstanta A. Kemudian log natural (ln) diambil dari effort aktual dan

Unadjusted Estimate. Untuk setiap proyek selisih antara log effort aktual dan *Unadjusted Estimate* ditentukan. Rata-rata dari selisih ini, X, akan menentukan konstanta A dengan mengambil anti log dari rata-rata : $A = e^X$.

Tabel 2.23 Contoh Kalibrasi A

PMactual	KSLOC	ΠEM_i	E	Unadjusted Estimate	$\ln(PM_{actual})$	ln(Unadjusted Estimate)	Difference
1854.6	134.5	1.89	1.20	686.7	7.53	6.53	0.99
258.5	132.0	0.49	1.08	94.3	5.55	4.55	1.01
201.0	44.0	1.06	1.13	77.7	5.30	4.35	0.95
58.9	3.6	5.05	1.09	20.3	4.08	3.01	1.07
9681.0	380.8	3.05	1.18	3338.8	9.18	8.11	1.06
7021.3	980.0	0.92	1.16	2753.5	8.86	7.92	0.94
91.7	11.2	2.45	1.15	38.9	4.52	3.66	0.86
689.7	61.6	2.38	1.17	301.1	6.54	5.71	0.83
						X=	0.96
						A=	2.62

Dari contoh ini dari konstanta COCOMO A= 2.94, kalibrasi menghasilkan konstanta lokal A = 2.62, yang dapat digunakan untuk melakukan estimasi proyek software di lingkungan lokal.

2.3 Penjadwalan Proyek

PERT dan CPM adalah dua metode penjadwalan proyek yang dapat diaplikasikan pada pengembangan software. Kedua teknik itu dikendalikan oleh informasi yang sudah dikembangkan pada aktivitas perencanaan proyek sebelumnya:

- Estimasi kerja
- Dekomposisi fungsi produk
- Pemilihan tipe proyek dan rangkaian tugas

Kesalingtergantungan antara tugas-tugas dapat ditentukan dengan menggunakan sebuah jaringan tugas. Tugas-tugas, kadang-kadang disebut WBS (*Work Breakdown Structure*), ditentukan untuk produk sebagai satu kesatuan atau untuk fungsi-fungsi

individual. Baik PERT dan CPM menyediakan peranti kuantitatif yang memperbolehkan perencana sistem informasi untuk :

(1) menentukan jalur kritis – rantai tugas yang menentukan durasi proyek;

(2) estimasi waktu yang paling mungkin bagi tugas-tugas individual;

(3) menghitung waktu batas yang membatasi suatu "jendela" waktu untuk suatu tugas tertentu.

Perhitungan waktu batas dapat menjadi sangat bermanfaat dalam penjadwalan proyek. Keterlambatan satu desain fungsi dapat memperlambat pengembangan dari fungsi-fungsi yang lain. [Pressman, 1997]

Contoh dari penjadwalan single proyek sebagai berikut :

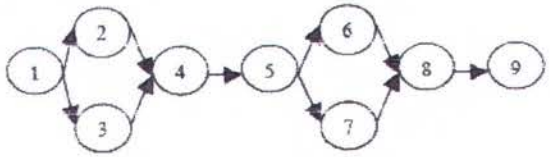
Tabel 2.24 Contoh Aktivitas untuk Proyek Customer Service

ID #	Activity Description	Immediate Predecessor
1	Customer service problem call notification	--
2	Detail interview of customer	1
3	Detail interview of original software design team And review of detailed documentation	1
4	Determination of problem root cause and reproduction of the problem	2 , 3
5	Brainstrom for solution and development of solution	4
6	Modify the database	5
7	Modify the code	5
8	Test the modification	6 , 7
9	Implement the solution	8

Pada tabel 2.24 menunjukkan contoh aktivitas-aktivitas suatu proyek dan hubungan urut-urutannya. Pada suatu aktivitas tidak dapat dilakukan bila kegiatan

yang mendahului *(predecessor)* belum selesai, misal : aktivitas 2 dan 3 baru boleh dilakukan bila aktivitas 1 selesai. Hubungan antar aktivitas dapat digambarkan diagram jaringannya.

Bila dibuat sebagai diagram jaringan digambarkan sebagai berikut :



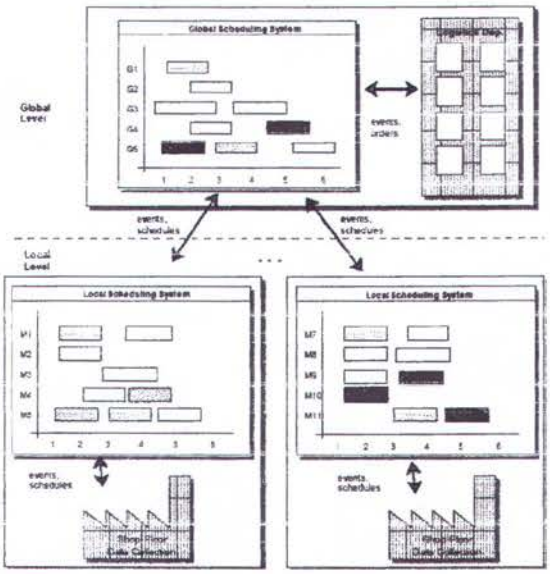
Gambar 2.1 Diagram Jaringan untuk Proyek Customer Service [Ash, 1999]

Dari gambar di atas dapat dilihat lebih jelas hubungan antar aktivitasnya, misalnya aktivitas 4 baru dapat dilakukan bila aktivitas 2 dan aktivitas 3 sudah selesai keduanya.

2.3.1 Penjadwalan Multi Proyek

Penjadwalan multi proyek adalah penjadwalan dimana sumber daya dibatasi dan dalam suatu waktu terdapat lebih dari satu proyek. Dalam penjadwalan ini akan terdapat dimana personel yang dipakai tidak mencukupi untuk melakukan semua aktivitas. Jika sumber daya sangat dibatasi maka beberapa proyek tidak dapat menyelesaikan aktivitasnya sesuai waktu yang ditentukan. Jadi proses harus memilih aktivitas mana yang sebaiknya dicukupi personelnya terlebih dahulu. Hal ini merupakan keputusan yang sulit untuk memilih antara meminimalkan keterlambatan proyek atau memaksimalkan penggunaan sumber daya. Teknik PERT/CPM dari single proyek tidak dapat membantu dalam situasi ini. [Ash, 1999]

Gambaran penjadwalan multi proyek adalah sebagai berikut :



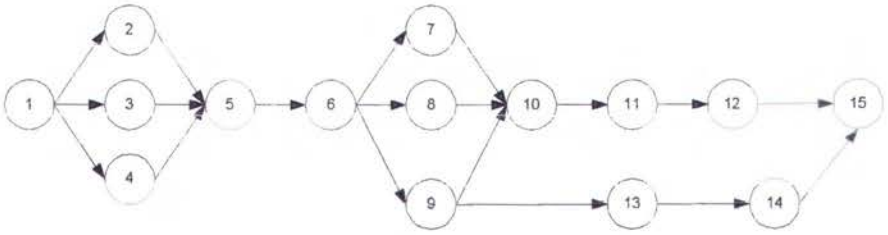
Gambar 2.2 Penjadwalan Multi Proyek[Sauer, 1998]

Pada gambar 2.2 terlihat terdapat penjadwalan global dan lokal. Penjadwalan lokal adalah penjadwalan masing-masing proyek yang ada. Setiap proyek yang datang akan dilakukan penjadwalan dan penempatan sumber dayanya. Bila suatu aktivitas proyek yang tidak mendapatkan sumber daya maka harus menunggu sampai sumber daya tersedia. Bila aktivitas tersebut terdapat di jalur kritis, maka proyek tersebut akan mengalami keterlambatan. Bila suatu aktivitas proyek sudah selesai maka sumber daya yang tadi dipakai, dikembalikan lagi agar dapat dipakai oleh aktivitas lain. Penjadwalan global adalah kumpulan dari penjadwalan lokal. Penjadwalan global biasanya digunakan oleh manajer untuk pengawasan.

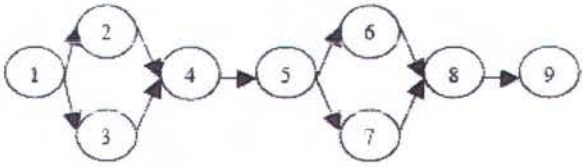
2.3.2 Penjadwalan Dinamis

Penjadwalan dinamis adalah penjadwalan multi proyek yang dilakukan setiap ada proyek yang datang secara dinamis dari waktu ke waktu. Jika ada proyek baru datang, maka penjadwalan akan dihitung kembali untuk mencari penyelesaian baru yang optimal.

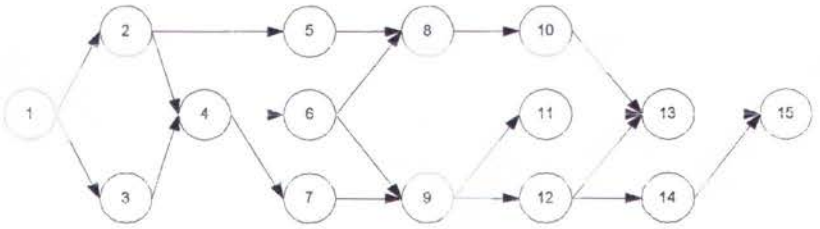
Proyek Software A



Proyek Software B



Proyek Software C



Gambar 2.3 Proyek yang Datang secara Dinamis[Ash, 1999]

Pada gambar di atas menunjukkan, proyek software A datang terlebih dahulu, kemudian datang proyek software B, lalu datang proyek software C. Pada saat proyek software A datang, maka proyek software A dibuat jadwalnya. Pada saat proyek software B datang, maka proyek software A dan B dijadwalkan kembali. Pada saat proyek software C datang, maka semua proyek dijadwalkan kembali. Penjadwalan seperti ini yang disebut penjadwalan dinamis.

2.3.3 Pendekatan Heuristik

Teknik optimasi hanya dapat digunakan untuk proyek kecil sesuai *time complexity* dan hanya memberikan solusi optimal untuk kasus tertentu saja. Bila terdapat perubahan secara dinamis pada penjadwalan, maka harus dicari lagi solusi optimalnya, sehingga membutuhkan waktu yang lama.

Keputusan untuk menempatkan sumber daya pada aktivitas multi proyek harus dibuat dengan cepat. Untuk itu dibutuhkan pendekatan heuristik yang efektif untuk meminimalkan keterlambatan proyek dan memaksimalkan pemakaian sumber daya. Terdapat beberapa pendekatan heuristik yang sering digunakan dalam penelitian untuk pengaturan sumber daya sebagai berikut : [Ash, 1999]

1. *SPT*, dengan mendahulukan aktivitas dengan durasinya terpendek
2. *MaxNPV*, dengan mendahulukan proyek yang harganya tertinggi
3. *Due Date*, dengan mendahulukan proyek yang jatuh temponya terdekat
4. *Permutation*, dengan mencoba semua kemungkinan urutan proyek dengan permutasi

Untuk mencari pendekatan heuristik mana yang paling optimal, maka harus dibandingkan keempat pendekatan tersebut.

Algoritma dalam membuat jadwal sebagai berikut :

1. Inisialisasi
   a. *Load* data proyek dari database yang dihasilkan modul Estimasi
   b. Hitung harga proyek, yaitu :

      (*finish date – start date*) / 30 hari * biaya *resource* perbulan
   c. Urutkan proyek sesuai dengan prioritas
      i. **SPT** : proyek yang terpendek durasinya didahulukan

ii. **Due Date** : proyek yang jatuh temponya terdekat didahulukan

iii. **NPV** : proyek yang harganya tertinggi didahulukan

iv. **Mixed** : dicoba SPT, Due Date, dan NPV kemudian dicari yang terendah biayanya

v. **Permutation** : dicoba semua kemungkinan urutan prioritas dengan permutasi, kemudian dicari yang terendah biayanya

d. Isi pada **Avail** (suatu array yang panjang) dengan tanggal dan jumlah *resource* yang tersedia

i. ***Surveyor*** bertugas pada *plans and requirement*

ii. ***Analist*** bertugas pada *product design*

iii. ***Programmer*** bertugas pada *programming*

iv. ***Operator*** bertugas pada *Integration and Test*

2. Untuk setiap proyek lakukan

a. Ambil *resource* (sumber daya) yang dibutuhkan dari array Avail sesuai dengan tanggal dan aktivitas resource

b. Jika resource tidak tersedia, pada *waiting time* (waktu tunggu) tambahkan 1, dan teruskan ke tanggal berikutnya sampai mendapatkan resource

c. Lakukan sampai semua aktivitas terpenuhi resourcenya

d. Update *start date (*tanggal mulai*)* , *finish date* (Tanggal selesai), dan durasi aktivitas dengan menambahkan waiting time

e. Hitung kerugiannya (akibat molornya jadwal), yaitu :

waiting time * harga proyek * 0.1%

3. Kalkulasikan biaya semua proyek

a. Hitung biaya resource (gaji staff) , yaitu :

(finish date (proyek yang terakhir) – start date (proyek pertama)) / 30 hari * biaya resource perbulan

b. Hitung total biaya, yaitu : **biaya resource + kerugian** (semua proyek)

4. Tampilkan di layar

Algoritma ini dilakukan setiap ada proyek yang datang.

2.4 Profil Perusahaan yang Dianalisa

PT X adalah salah satu anak perusahaan minyak bumi di Indonesia yang mengeksplorasi dan memproduksi minyak mentah melalui lebih dari seratus lapangan minyak. Kantor pusat perusahaan berlokasi di Jakarta, sedangkan daerah operasinya berada di daratan propinsi Riau, Sumatera Tengah, persisnya di empat distrik utama yaitu Rumbai, Minas, Duri dan Dumai. Sebagai perusahaan besar dengan jumlah pegawai sekitar 5.000 orang dan dengan tingkat produksi minyak mentah sekitar 600.000 barel per hari, PT X memerlukan cara-cara pengelolaan bisnis yang efisien agar tetap bisa mempertahankan bahkan meningkatkan keunggulan kompetitifnya. Tingkat keunggulan kompetitif yang dimiliki oleh PT X akan menentukan apakah para pemegang saham (*shareholders*) akan tetap memilih PT X untuk menanamkan modalnya.

Salah satu cara untuk memenangkan keunggulan kompetitif adalah dengan mengefisienkan proses bisnis perusahaan, dan salah satu caranya adalah dengan menerapkan berbagai perangkat lunak yang bisa digunakan untuk mempercepat proses kerja dan penyediaan data untuk pengambilan keputusan pihak manajemen perusahaan. Dalam usaha menerapkan berbagai perangkat lunak untuk mendukung operasinya, PT X menempuh dua pendekatan berdasarkan kesediaan fitur perangkat

lunak yang dibutuhkan tersebut di pasar. Apabila fitur perangkat lunak yang diperlukan bersifat baku dan tersedia di pasar, maka perusahaan akan memilih pendekatan 'membeli jadi' perangkat lunak tersebut di pasar. Sedangkan apabila fitur perangkat lunak yang diperlukan relatif unik, dalam arti hanya diterapkan di PT X, dan biasanya perangkat lunak semacam itu tidak tersedia di pasar, maka perusahaan akan memilih pendekatan 'mengembangkan sendiri' secara *in-house* perangkat lunak tersebut, yang dilakukan oleh tim teknologi Informasi yang ada dalam perusahaan atau menyerahkan pengembangan perangkat lunak tersebut kepada kontraktor pengembang perangkat lunak dengan cara kontrak (*outsourcing*).

PT X memiliki strategi perusahaan yang dikenal dengan sebutan 4+1, dibaca *four plus one*, yang mencakup strategi-strategi :

a. *Profitable Grow* (Pertumbuhan yang menguntungkan)
b. *Operational Excellence* (Keunggulan operasi)
c. *Cost Ruduction* (Penurunan biaya)
d. *Capital Stewardship* (Pengelolaan modal)
e. *Organization Capability* (Kemampuan Organisasi)

Visi yang ingin diwujudkan PT X dengan strategi tersebut adalah "Menjadi perusahaan energi Indonesia yang paling dihormati berkat dukungan para pekerja, mitra, dan kinerja", untuk itu perlu diciptakan pertumbuhan yang menguntungkan. Pertumbuhan yang menguntungkan hanya bisa dicapai dengan dukungan strategi Keunggulan Operasi, Penurunan Biaya, dan Pengelolaan Modal, dan ketiga strategi ini secara keseluruhan perlu dukungan strategi Kemampuan Organisasi.

Dalam kaitannya dengan strategi perusahaan, maka estimasi yang akurat mengenai biaya pengembangan perangkat lunak memiliki kaitan dengan strategi

pengelolaan modal perusahaan. Estimasi biaya yang lebih akurat berarti mendukung langkah perusahaan untuk mengelola modal secara lebih baik dan bertanggungjawab.

2.5 Profil Program yang Dianalisa

Terdapat empat software yang dikembangkan sendiri secara in-house di PT X, yaitu :

a. ABOM (*Activity Based OEB Management*)
b. PLF (*Profit & Loss by Field*)
c. AFE BIRDS (*Appropriation For Expenditures Business Information & Report Distribution System*)
d. FAM (*Fixed Asset Management*)

2.5.1 ABOM (Activity Based OEB Management)

Sistem ABOM adalah software yang digunakan untuk mencatat dan mengkonsolidasi data anggaran biaya operasional perusahaan serta mengawasi dan mengendalikan pengeluaran biaya operasional perusahaan. Sistem ABOM terdiri dari empat modul yaitu: (1) *Budget Inputting & Consolidation*, (2) *Expenditure Monitoring & Controlling*, (3) *Approvals*, dan (4) *References*.

Modul *Budget Inputting & Consolidation* terdiri dari lima sub-modul yaitu: (a) *Standard Rate Budget*, (b) *User Defined Rate Budget*, (c) *Project Expense Budget*, (d) *Redistributed Budget*, dan (e) *Budget Summary Reports*. Sub-modul *Expenditure Monitoring & Controlling* lebih lanjut terdiri dari dua sub-sub-modul yaitu: (a) *Expenditure Forecasting* dan (b) *Budget vs Expenditure Reports*.

Penjelasan modul dan sub-modul sistem ABOM adalah sebagai berikut:

Modul *Budget Inputting & Consolidation* digunakan untuk memasukkan data anggaran dari setiap permilik pusat-biaya (*cost-center owner*) ke dalam sistem

ABOM dan untuk mengkonsolidasikan data anggaran ke dalam jenjang-jenjang organisasi yang ada di PT X, mulai dari jenjang paling bawah, yaitu jenjang *Team Leader*, kemudian ke jenjang di atasnya yaitu jenjang *Team Manager*, selanjutnya ke jenjang di atasnya lagi yaitu jenjang *Manager*, kemudian ke jenjang *Vice President*, sampai ke jenjang yang paling tinggi yaitu jenjang *President Director*. Dengan adanya hirarki pusat-biaya (*cost-center hierarchy*) yang diterapkan di dalam sistem ABOM maka data anggaran dapat dilihat dari berbagai jenjang organisasi. Apabila dimulai dari jenjang yang paling atas (jenjang PT X), maka data anggaran dapat dilacak distribusinya ke jenjang-jenjang di bawahnya, demikian seterusnya sampai ke jenjang paling bawah (jenjang *Team Leader*). Demikian pula sebaliknya, apabila dimulai dari jenjang paling bawah, maka data anggaran dapat dilacak akumulasinya ke jenjang-jenjang di atasnya, sampai ke jejang paling atas, yaitu jenjang PT X.

Modul *Standard Rate Budget* digunakan untuk memasukkan data anggaran tipe *Standard Rate*, yaitu tipe anggaran di mana *rate* atau biaya per unit aktivitasnya sudah baku, ditentukan oleh perusahaan dan berlaku sama untuk semua pemilik pusat-biaya. Yang termasuk tipe anggaran Standard Rate adalah jenis-jenis anggaran untuk membiayai *Salary, Overtime, Light Cars, PC & Printers, Fuel & Lubricant*, dan sebagainya.

Modul *User Defined Rate* digunakan untuk memasukkan data anggaran tipe *User Defined*, yaitu tipe-tipe anggaran di mana *rate* atau biaya per unit aktivitasnya sebagian ditentukan sendiri oleh pemilik pusat-biaya yang memasukkan data anggaran tersebut berdasarkan data histori yang dimilikinya. Yang termasuk dalam tipe anggaran User Defined Rate adalah jenis-jenis anggaran untuk membiayai

Training, Business Travel, Contract Services, Rental/Lease, Medical Supplies, Materials & Supplies, Utilities, Insurance, dan sebagainya.

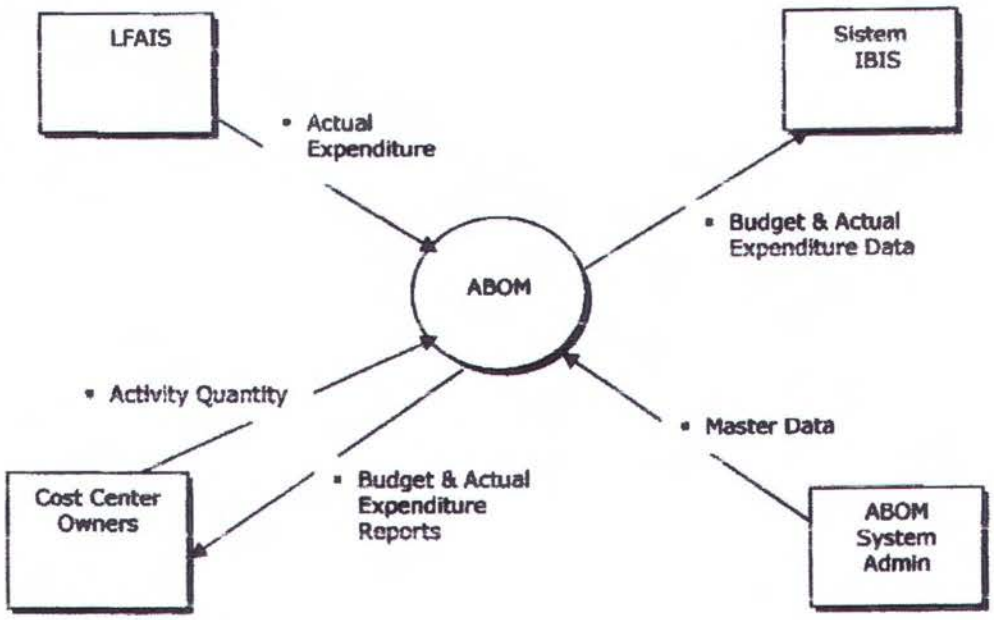
Modul *Project Expense Budget* digunakan untuk memasukkan data anggaran tipe proyek, yaitu tipe-tipe anggaran yang dipakai untuk membiayai kegiatan suatu proyek.

Modul *Redistributed Budget* digunakan untuk memasukan data anggaran tipe *Redistributed* yaitu tipe anggaran yang berupa hasil redistribusi anggaran secara otomatis dari tim-tim tertentu misalnya tim penyedia perumahan (*housing*), tim penyedia listrik, yang tidak lain merupakan tim penyedia layanan kepada tim-tim penerima layanan.

Modul *Expenditure Monitoring & Controlling* digunakan untuk menyusun rencana (*forecast*) pengeluaran bulanan dan untuk membuat laporan-laporan yang berisi perbandingan antara anggaran dengan pengeluaran.

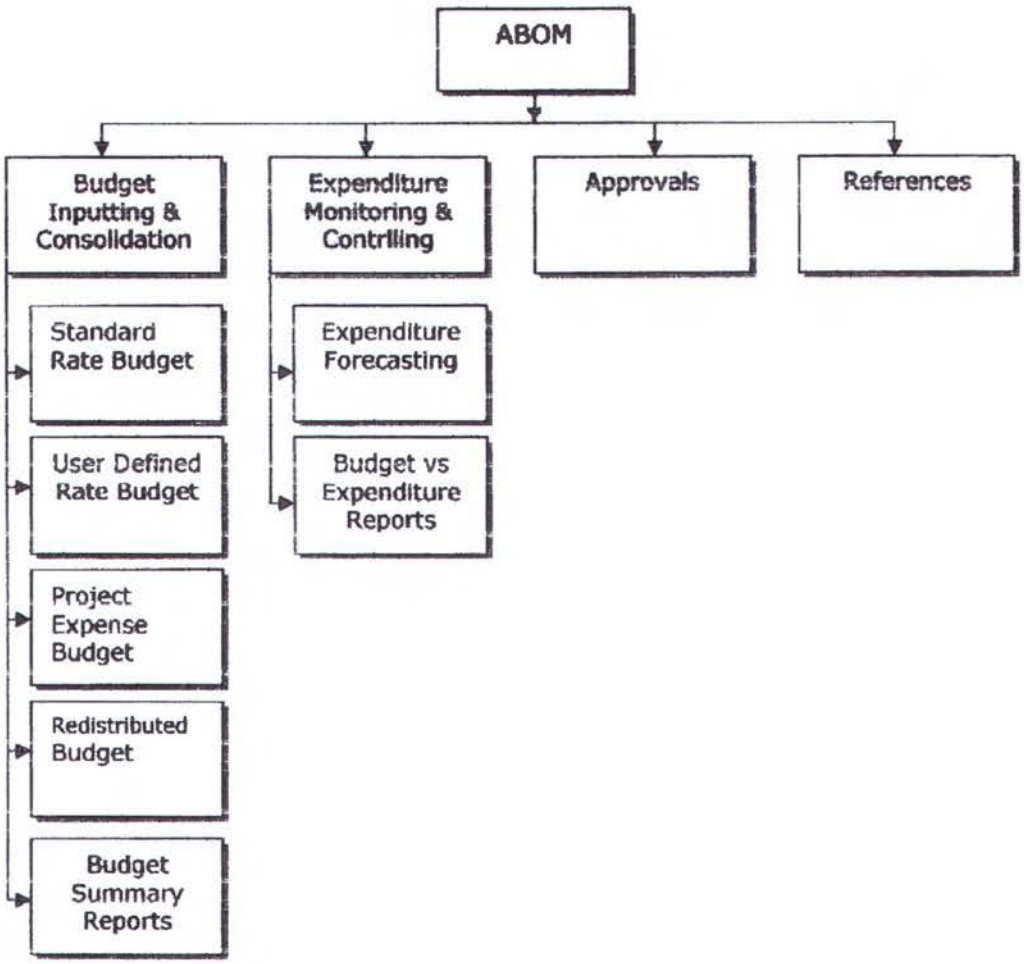
Modul *Approvals* digunakan untuk mendapatkan pengesahan (*approval*) anggaran dari jenjang-jenjang manajemen tertentu, yaitu *Team Manager, Manager*, dan seterusnya ke atas sampai ke jenjang *Vice President*.

Modul *References* digunakan untuk menyimpan tabel-tabel tertentu misalnya tabel *Standard Rate, Cost-Center, Account, Cost-Center Hierarchy*, dan sebagainya.



Gambar 2.4 Diagram Konteks Sistem ABOM

Agar terlihat jelas proses-proses apa saja yang ada di dalam sistem ABOM dan data apa saja yang melalui proses-proses tersebut dan mengalami tranformasi, maka sistem ABOM perlu digambarkan secara Diagram Aliran Data (*Data Flow Diagram*). Diagram Aliran Data akan dibuat dalam dua level yaitu, Level 0 atau sering disebut dengan *Context Diagram* dan Level 1. Bilamana diperlukan untuk lebih menjelaskan aliran data dan proses yang ada dalam sistem maka akan dibuat juga Diagram Aliran Data Level 2.

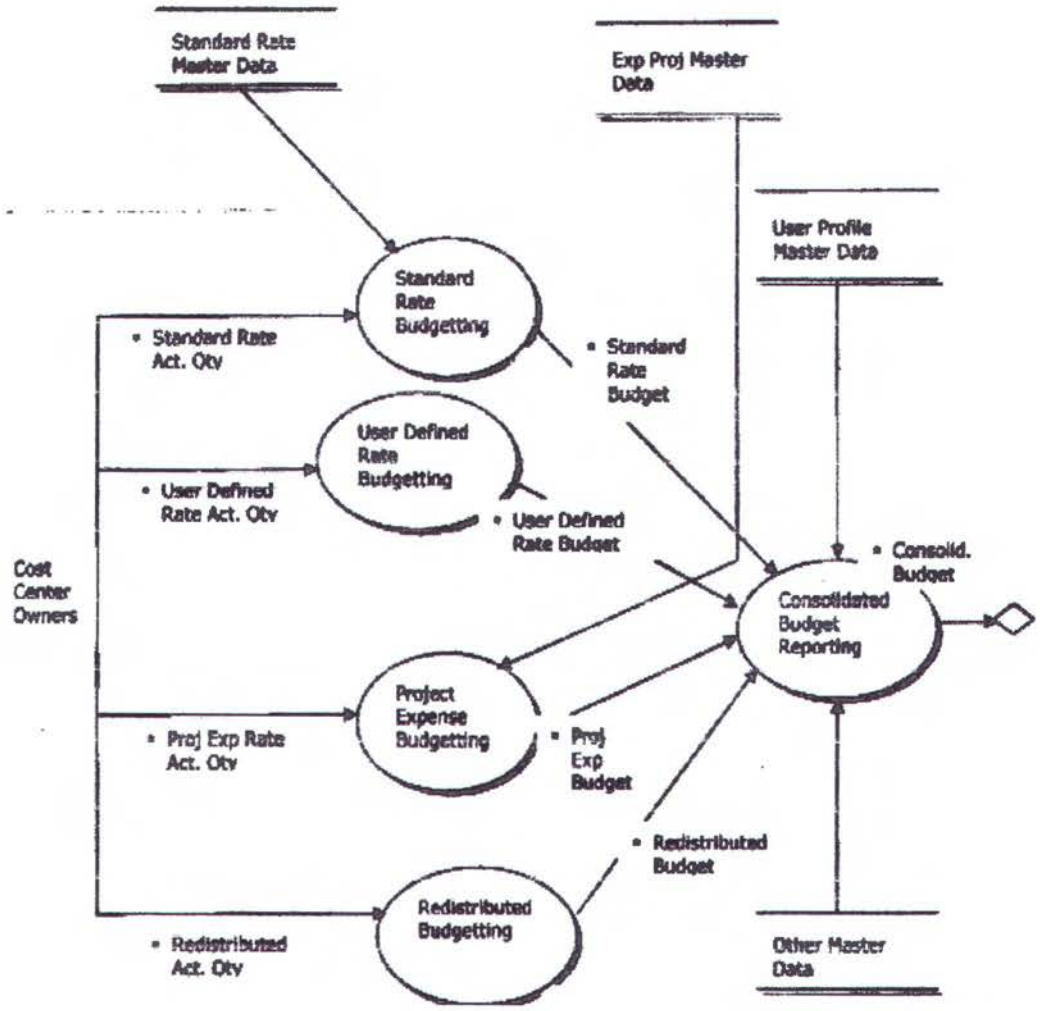


Gambar 2.5 Dekomposisi Fungsi ABOM

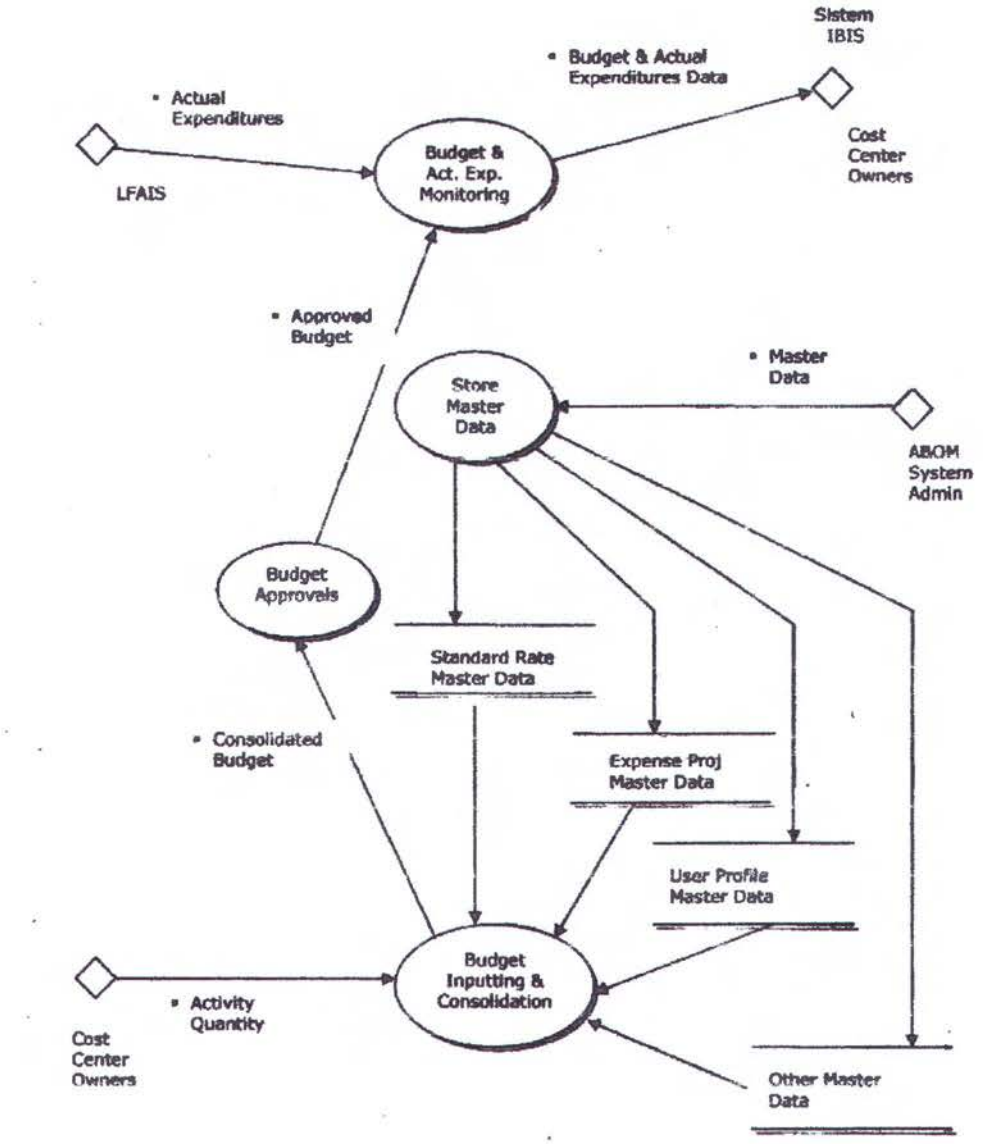
Dari Data Flow Diagram Level 0 (Context Diagram) dapat dilihat bahwa sistem ABOM menerima data *Actual Expenditures* dari sistem ERP (*Enterprise Resource Planning*) LFAIS (*Logistics, Financial and Accounting Information System*), data *Budget Activity Quantity* dari *Cost Center Owner* (yaitu tim-tim dalam struktur organisasi PT X yang berfungsi sebagai pemilik *Cost Center* atau Pusat Biaya), dan Master Data dari ABOM System Administrator. Di sisi lain sistem ABOM memberikan data *Budget & Actual Expenditures* ke *Cost Center Owner* (dalam

bentuk laporan) dan ke sistem IBIS (dalam bentuk data mentah untuk diolah lebih lanjut di dalam sistem IBIS). Bila sistem ABOM didekomposisi lebih lanjut akan menjadi DFD level 1 sistem ABOM.

Untuk memberikan gambaran lebih detail mengenai sub-proses yang ada dalam proses *Budget Inputting & Consolidation*, maka proses ini dikomposisi lebih lanjut menjadi DFD level 2.



Gambar 2.6 DFD ABOM Level 1

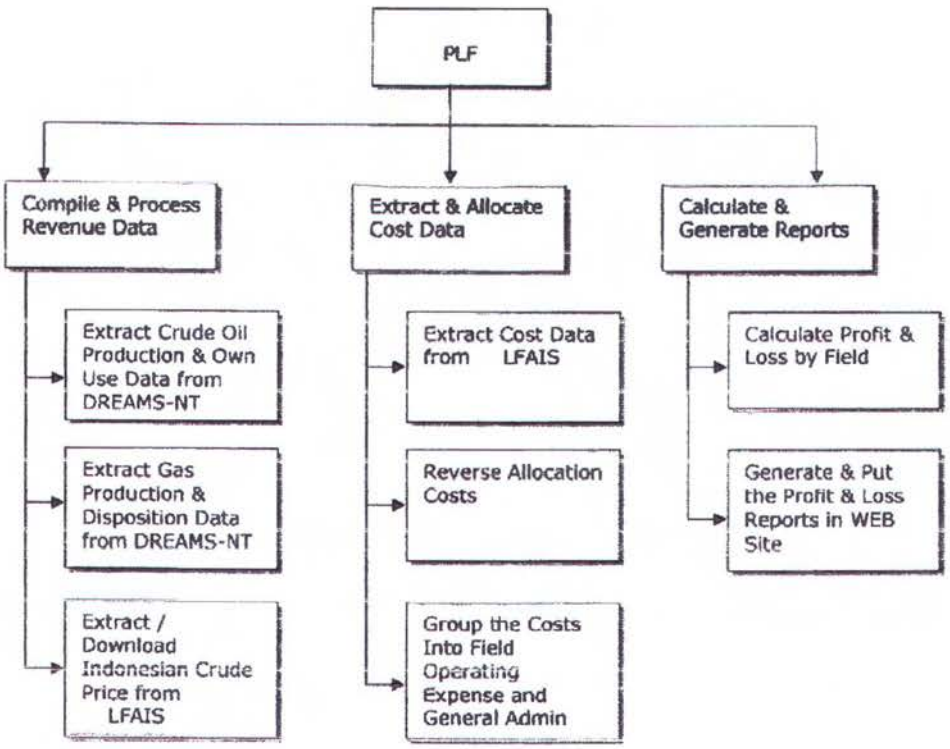


Gambar 2.7 DFD ABOM Level 2 (*Budget Inputting & Consolidation*)

2.5.2 PLF (Profit & Loss by Field)

PLF (*Profit & Loss by Field*) adalah sistem perangkat lunak yang digunakan untuk mengelola kinerja (dalam bentuk Keuntungan dan Kerugian) yang dihasilkan oleh setiap lapangan minyak milik PT X. Selain itu sistem PLF juga digunakan sebagai alat bantu untuk pembuatan keputusan oleh pihak manajemen puncak PT X

guna menentukan lapangan minyak mana yang akan ditutup sementara apabila tejadi kekurangan pasokan tenaga listrik yang dibutuhkan oleh lapangan minyak dan juga apabila tejadi pemotongan kuota produksi minyak oleh organisasi negara-negara pengekspor minyak OPEC (*Organization of Petroleum Exporting Countries*). Sistem PLF juga membantu semua tingkatan manajemen dalam hal pembuatan keputusan untuk melanjutkan investasi atau menutup secara permanen suatu lapangan minyak tertentu, mengidentifikasi area untuk memperbaiki kinerja lapangan minyak, dan juga untuk mengoptimalkan alokasi sumber daya untuk pengelolaan setiap lapangan minyak.

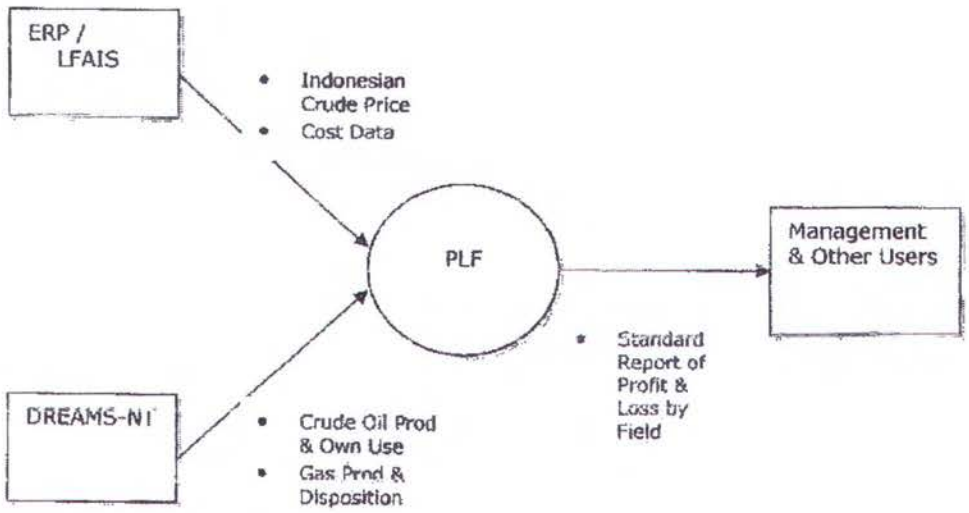


Gambar 2.8 Dekomposisi Fungsi Sistem PLF

Sistem PLF menerima data dari dua sistem aplikasi lainnya yaitu: dari sistem ERP/LFAIS (untuk data biaya operasional dari setiap lapangan minyak) dan dari

sistem produksi minyak DREAMS-NT (untuk data produksi minyak dari setiap lapangan minyak). Selain itu sistem PLF juga menerima data biaya dari tim HCT (*Hydro Carbon Transportation*) yaitu biaya pengangkutan minyak dari lapangan ke tempat pengapalan minyak di kota Dumai, dari tim PG&T (*Power Gas & Turbine*) yaitu biaya pemakaian listrik oleh lapangan minyak, dari tim FMT (*Field Maintenance Team*) yaitu biaya pemeliharaan lapangan minyak, dan dari tim AMT (*Asset Maintenance Team*) yaitu untuk biaya pemeliharaan asset.

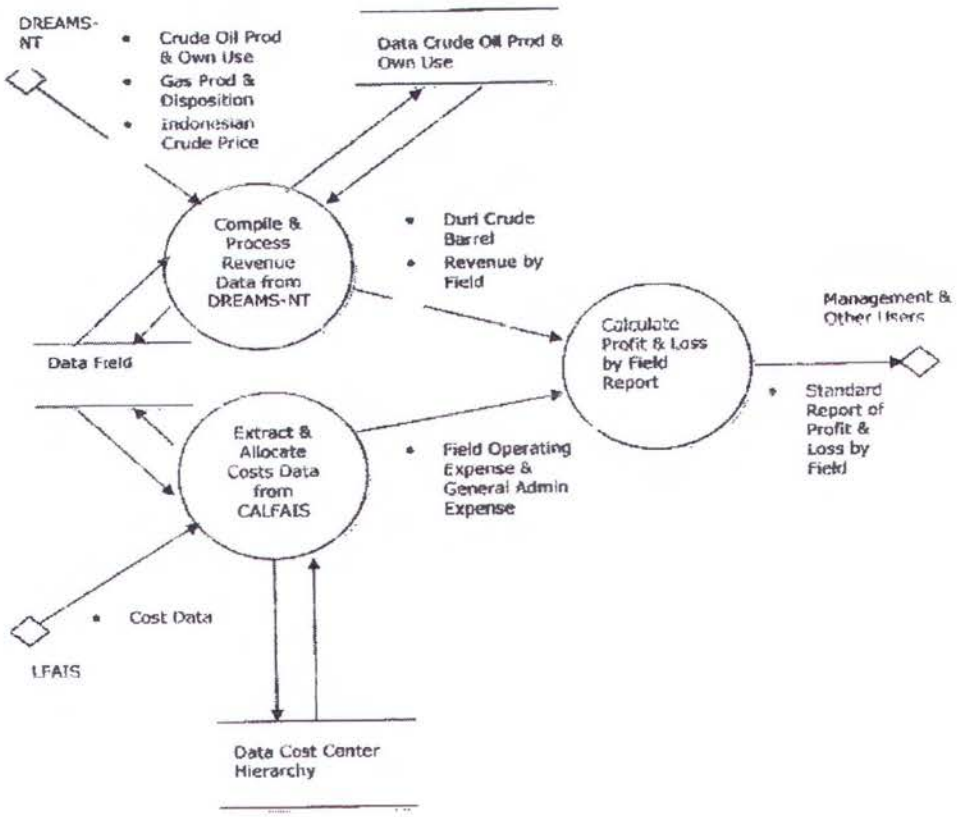
Untuk memahami proses-proses yang ada dalam sistem PLF dan data yang mengalir melalui proses-proses tersebut maka perlu dibuat Diagram Aliran Data. Diagram Aliran Data mengacu pada proses-proses pada Dekomposisi Fungsi sistem PLF.



Gambar 2.9 Context Diagram Sistem PLF

Dari Context Diagram pada Gambar 2.9 terlihat bahwa sistem PLF menerima masukan data mengenai harga minyak mentah Indonesia (*Indonesian Crude Price*) dan berbagai data biaya (*Costs*) per lapangan minyak dari sistem ERP / LFAIS dan

menerima masukan data mengenai produksi minyak mentah (*Crude Oil Production*) per lapangan minyak, jumlah minyak mentah yang dipakai sendiri oleh PT X sebagai bahan bakar (*Own Use*) per lapangan minyak dan produksi Gas dari sistem DREAMS-NT. Setelah dilakukan pemrosesan maka sistem PLF memberikan keluaran (output) berupa berbagai laporan Keuntungan & Biaya per lapangan minyak (*Profit & Loss by Field*) kepada pihak Manajemen dan Pengguna lainnya di PT X.



Gambar 2.10 Data Flow Diagram Level 1 Sistem PLF

Dari Data Flow Diagram Level 1 dapat dilihat bahwa ada tiga sub proses dalam sistem PLF, yaitu *Compile & Process Revenue Data from DREAMS-NT, Extract & Allocate Cost Data from LFAIS* dan *Calculate Profit & Loss by Field Reports.*

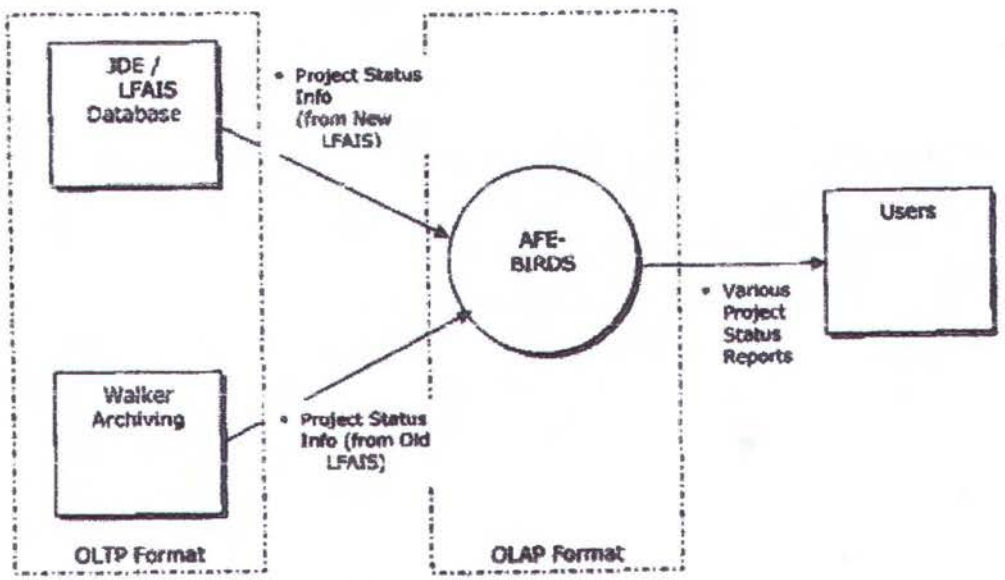
2.5.3 AFE BIRDS (Appropriation For Expenditures Business Information & Report Distribution System)

Sistem AFE BIRDS adalah suatu aplikasi Web yang dikembangkan secara *in-house* oleh tim TI yang ada di PT X yang digunakan sebagai alat untuk mengawasi pengeluaran biaya pekerjaan proyek. Sebagian besar data diekstrak melalui proses ETL (*Extract, Tranform, and Loading*) secara bulanan dari sistem aplikasi ERP (*Enterprise Resource Planning*) JD Edward/LFAIS dalam format OLTP (*On Line Transaction Processing*), untuk kemudian diubah ke dalam format OLAP (*On Line Analytical Processing*) dan disimpan di suatu warehouse database. Halaman-halaman ASP (*Active Server Page*) kemudian akan mengambil data dari database tersebut untuk menghasilkan berbagai laporan berisi informasi untuk pengguna.

Halaman-halaman ASP di dalam siste AFE BIRDS dibagi beberapa bagian, yaitu:

- *Reports*, adalah kumpulan dari halaman ASP yang memberikan berbagai informasi berkaitan dengan pengeluaran biaya proyek dalam format yang sudah ditentukan (*predefined format*) dan dalam format yang interaktif atau bisa dipilih oleh pengguna. Pengguna akan diberi informasi dalam format yang didesain untuk mengawasi pengeluaran proyek dalam berbagai sudut pandang (*views*) dan dapat melakukan query interaktif (*interactive query*) untuk mendapatkan lebih banyak informasi mengenai status pengeluaran proyek, bahkan sampai ke tingkat rinci pengeluaran. Informasi yang dihasilkan bisa didownload ke dalam format Excel sehingga pengguna dapat melakukan analisis lebih lanjut terhadap data pengeluaran proyek.

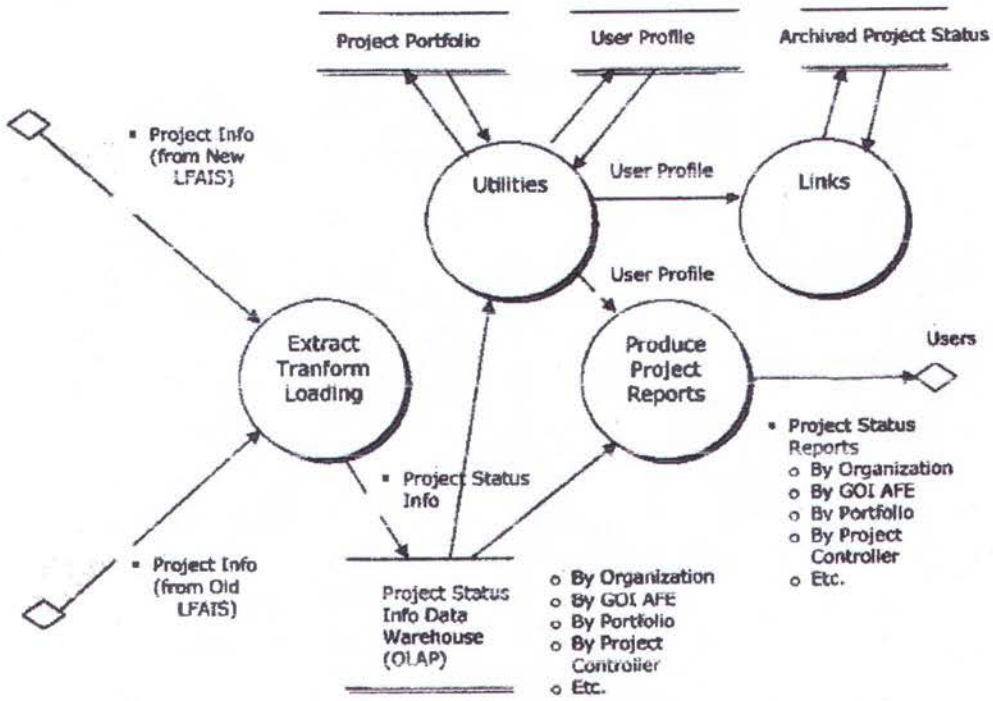
- *Inquiries*, adalah kumpulan dari halaman ASP yang membantu pengguna untuk melakukan pencarian informasi mengenai pengeluaran proyek dalam bentuk jenjang-jenjang hirarki yang sudah ditentukan.
- *Utilities*, adalah kumpulan halaman ASP yang digunakan untuk memelihara beberapa bentuk data di AFE BIRDS, seperti misalnya portofolio PM (Performance Measure), hubungan antara portofolio dan nomor proyek, menyalin portofolio, dan mengirim email berisi status proyek kepada penganggung jawab proyek (project controllers).
- *Links*, berisi links ke berbagai website yang berhubungan dengan pengelolaan proyek yaitu website LFAIS Archiving untuk mengambil informasi dalam format LFAIS Walker (format sistem ERP sebelum menggunakan JD Edward)



Gambar 2.11 Context Diagram Sistem AFE BIRDS

Dari Context Diagram pada, Gambar 2.11 dapat dilihat bahwa sistem AFE-BIRDS menerima masukan data dari sistem JDE/LFAIS dalam bentuk project info status (dalam format LFAIS baru) dan menerima masukan data dari sistem Walker

Archiving dalam bentuk project info status (dalam format LFAIS lama). Perlu diketahui bahwa baik sistem JDE/LFAIS database maupun sistem Walker Archiving keduanya adalah sistem OLTP (*On Line Transaction Processing*). Project info status tersebut di dalam sistem AFE-BIRDS akan transformasikan menjadi data dalam format OLAP (*On Line Analytical Processing*). Dapat disimpulkan bahwa sistem AFE-BIRDS adalah mengubah berbagai informasi mengenai proyek yang semula berbentuk OLTP menjadi informasi proyek yang disimpan dalam bentuk OLAP sehingga memudahkan pengguna untuk melakukan berbagai analisis.



Gambar 2.12 DFD Level 1 Sistem AFE BIRDS

2.5.4 FAM (Fixed Asset Management)

Sistem FAM adalah perangkat lunak yang digunakan untuk :

a. Mengontrol properti, bangunan dan peralatan milik perusahaan, yang mencakup transaksi-transaksi berikut:

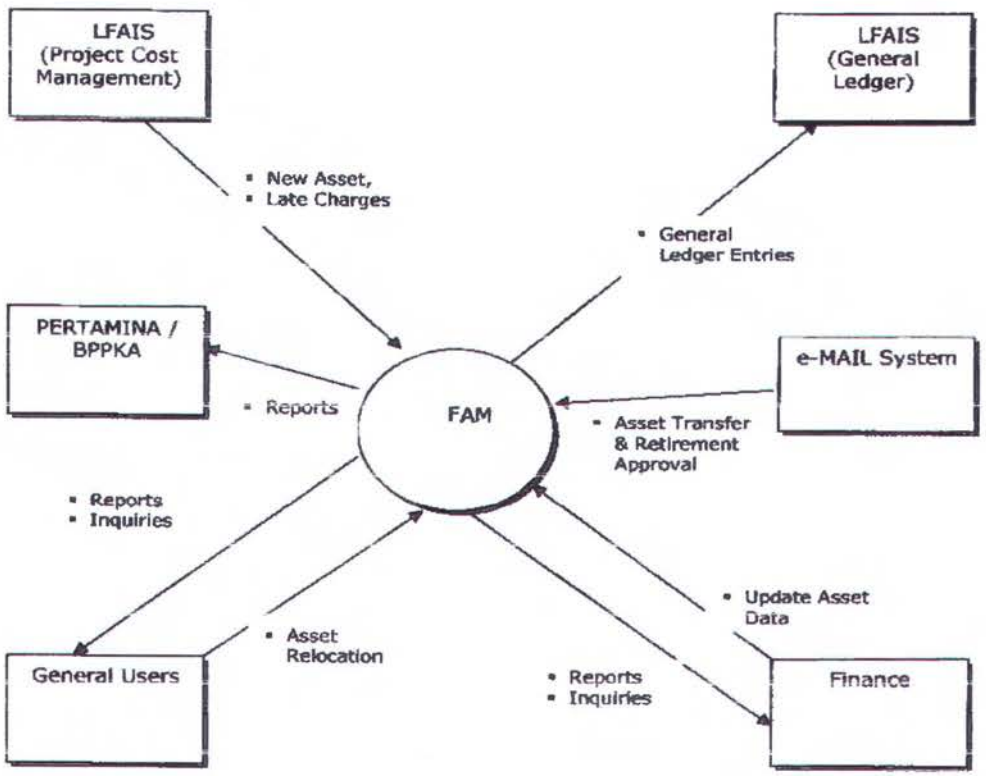
- Penambahan dan penyesuaian (*adjustment*) harta perusahaan (*asset*)
- Pemindahan (*transfer*) dan pemensiunan (*retirement*) harta perusahaan

b. Melakukan penghitungan secara otomatis depresiasi harta perusaahaan dari seluruh area kontrak PT X

c. Menghasilkan data masukan ke sistem akunting secara on-line untuk mencatat transaksi properti, bangunan dan peralatan

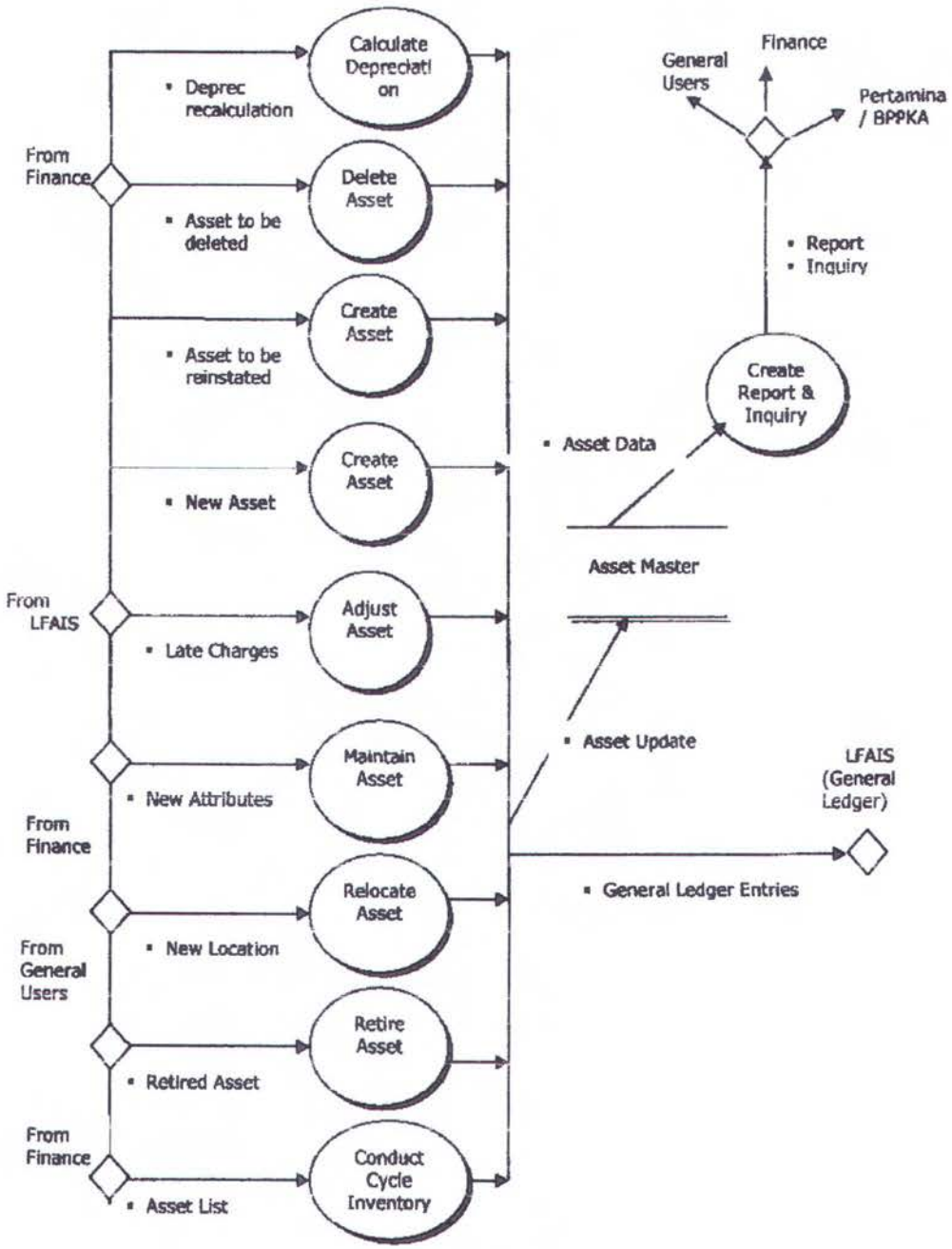
d. Menghasilkan laporan-laporan

e. Memberikan fasilitas untuk pencarian dan penelusuran data (*data queries and tracking*)



Gambar 2.13 Diagram Konteks Sistem FAM

Dari Context Diagram pada Gambar 2.13 dapat dilihat bahwa sistem FAM masukan data dari sistem LFAIS (modul *Project Cost Management*) berupa Aset Baru (*New Asset*) dan Tagihan Terlambat atas biaya pengerjaan proyek (*Late Charges*). Selain itu sistem FAM juga menerima masukan data dari Pengguna Umum (*General Users*) dalam bentuk Pemidahan Lokasi Aset (*Asset Relocation*), dari bagian Keuangan (*Finance*) dalam bentuk update informasi aset (*Update Asset Data*), dan dari sistem Email berupa pengesahan pemindahan dan pemensiunan aset (*Asset Transfer and Retirement Approval*). Di sisi lain sistem FAM juga memberikan keluaran (*output*) berupa berbagai laporan aset kepada Pertamina / BPPKA, Pengguna Umum dan bagian Finance. Selain itu juga memberikan keluaran berupa data Buku Besar (*General Ledger Entries*) ke sistem LFAIS (modul *General Ledger*).



Gambar 2.14 DFD Level 1 Sistem FAM

BAB III

ANALISA DATA DAN DESAIN PERANGKAT LUNAK

Pada bab ini data yang diperoleh akan dianalisa dan perangkat lunak akan dirancang. Untuk data profil yang diperoleh dari PT X akan dihitung level CMM, sedang data dari software yang telah dibuat dihitung jumlah UFP, dan ditentukan tingkat scale factor dan cost driver. Sedangkan perangkat lunak akan dirancang dengan Diagram Aliran Data, yaitu terdiri dari bagan berjenjang, contex diagram, level 0, dan level 1. Penggunaan Diagram Arus Data sangat membantu untuk memahami suatu sistem pada semua tingkat kompleksitas. Diagram Aliran Data merupakan alat yang cukup populer sekarang ini, karena dapat menggambarkan arus data dalam sistem dengan terstruktur dan jelas. Diagram Aliran Data juga merupakan dokumentasi dari sistem yang baik. Penggambaran Diagram Aliran Data ini menggunakan software Power Design.

3.1 Penghitungan Level CMM (*Capability Maturity Model*)

Penerapan model estimasi biaya COCOMO mensyaratkan adanya pengukuran *Scale Factors* yang memiliki dampak pada tingkat *economies* atau *diseconomies* suatu proyek pengembangan perangkat lunak. Salah satu Scale Factor tersebut adalah PMAT(*Process Maturity*) yang menggambarkan tingkat kematangan organisasi dalam melaksanakan proses pengembangan perangkat lunak. Tingkat kematangan tersebut diukut dengan menggunakan kriteria tingkat kematangan (*Maturity Level*) yang dikenal dengan nama CMM (*Capability Maturity Model*) yang

dikeluarkan oleh SEI (*Software Engineering Institute*).Kerangka tingkat kematangan proses pengembangan software yang diberikan oleh CMM meliputi :

- Praktek-praktek yang dapat diulang, artinya dalam organisasi terdapat kebijakan, prosedur, dan praktek-praktek yang digunakan secara konsisten
- Praktek-praktek terbaik dapat disebarkan dengan cepat atau dipindahkan ke kelompok lain
- Tujuan pekerjaan ditentukan kuantitatif dan terdapat ukuran-ukuran jelas dan dipelihara sebagaik dasar penilaian
- Praktek-praktek diperbaiki secara terus menerus untuk meningkatkan kemampuan organisasi.

Tabel 3.1 KPA CMM PT X

Key Process Areas (KPA)	Almost Always[1]	Frequently[2]	About Half[3]	Occasionally[4]	Rarely If Ever[5]	Does Not Apply[6]	Don't Know[7]
Requirements Management • System requirements allocated to software are controlled to establish a baseline for software engineering and management use. • Software plans, products, and activities are kept consistent with the system requirements allocated to software.	-	√	-	-	-	-	-
Software Project Planning • Software estimates are documented for use in planning and tracking the software project. • Software project activities and commitments are planned and documented. • Affected groups and individuals agree to their commitments related to the software project.	-	√	-	-	-	-	-

Tabel 3.1a Lanjutan Tabel 3.1 KPA CMM PT X

Key Process Areas (KPA)	Almost Always'	Frequently'	About Half'	Occasionally'	Rarely if Ever'	Does Not Apply'	Don't Know'
Software Project Tracking and Oversight • Actual results and performances are tracked against the software plans • Corrective actions are taken and managed to closure when actual results and performance deviate significantly from the software plans. • Changes to software commitments are agreed to by the affected groups and individuals.	-	-	√	-	-	-	-
Software Subcontract Management • The prime contractor selects qualified software subcontractors. • The prime contractor and the subcontractor agree to their commitments to each other. • The prime contractor and the subcontractor maintain ongoing communications. • The prime contractor tracks the subcontractor's actual results and performance against its commitments.	-	-	-	√	-	-	-
Software Quality Assurance (SQA) • SQA activities are planned. • Adherence of software products and activities to the applicable standards, procedures, and requirements is verified objectively. • Affected groups and individuals are informed of software quality assurance activities and results. • Noncompliance issues that cannot be resolved within the software project are addressed by senior management.	-	√	-	-	-	-	-
Software Configuration Management (SCM) • SCM activites are planned. • Selected workproducts are identified, controlled, and available. • Changes to identified work products are controlled • Affected groups and individuals are informed of the status and content of software baselines.	-	√	-	-	-	-	-
Organization Process Focus • Software process development and improvement activities are coordinated across the organization. • The strengths and weaknesses of the software processes used are identified relative to a process standard. • Organization-level process development and improvement activities are planned.	-	√	-	-	-	-	-
Organization Process Definition • A standard software process for the organiation is developed and maintained. • Information related to the use of the organization's standard software process by the software projects is collected, reviewed, and made available.	√	-	-	-	-	-	-

Tabel 3.1b Lanjutan Tabel 3.1a KPA CMM PT X

Key Process Areas (KPA)	Almost Always'	Frequently'	About Half'	Occasionally'	Rarely if Ever'	Does Not Apply'	Don't Know'
Training Program • Training activities are planned. • Training for developing the skills and knowledge needed to perform software management and technical roles is provided. • Individuals in the software engineering group and software-related groups receive the training necessary to perform their roles.	-	√	-	-	-	-	-
Integrated Software Management • The project's defined software process is a tailored version of the organization's standard software process. • The project is planned and managed according to the project's defined software process.	-	√	-	-	-	-	-
Software Product Engineering • The software engineering tasks are defined, integrated, and consistently performed to produce the software • Software work products are kept consistent with each other.	-	-	√	-	-	-	-
Intergroup Coordination • The customer's requirements are agreed to by all affected groups. • The commitments between the engineering groups are agreed to by the affected groups. • The engineering groups identify, track, and resolve intergroup issues.	√	-	-	-	-	-	-
Peer Reviews • Peer review activities are planned. • Defects in the software work products are identified and removed.	-	-	√	-	-	-	-
Quantitative Process Management • The quantitative process management activities are planned. • The process performance of the project's defined software process is controlled quantitatively. • The process capability of the organization's standard software process is known in quantitative terms.	-	-	-	√	-	-	-
Software Quality Management • The project's software quality management activities are planned. • Measurable goals of software product quality and their priorities are defined. • Actual progress toward achieving the quality goals for the software products is quantified and managed.	-	-	-	√	-	-	-
Defect Prevention • Defect prevention activities are planned. • Common causes of defects are sought out and identified. • Common causes of defects are prioritized and systematically eliminated.	-	√	-	-	-	-	-
Technology Change Management • Incorporation of technology changes are planned. • New technologies are evaluated to determine their effect on quality and productivity. • Appropriate new technologies are transferred into normal practice across the organization.	-	-	√	-	-	-	-
Process Change Management • Continuous process improvement is planned. • Participation in the organization's software process improvement activities is organization wide. • The organization's standard software process and the project's defined software processes are improved continuously.	-	-	-	-	√	-	-

Tabel 3.2 Perhitungan KPA CMM

Level	KPA	Total Score
Almost Always	2	2 x 100% = 200 %
Frequently	4	4 x 75% = 300 %
About Half	7	7 x 50% = 350 %
Occasionaly	3	3 x 25% = 75 %
Rare if Ever	2	2 x 1% = 2 %
Total :		**927%**

$$EPML = 5 \times \left(\sum_{i=1}^{n} \frac{KPA\%_i}{100} \right) \times \frac{1}{n} \qquad \text{.....................................} (15)$$

EPML = 5 x 927 / 100 x 1/18

= 2.575 ≈ **(level 2)**

Jadi dari perhitungan ini dapat disimpulkan bahwa PT X pada CMM mempunyai **level 2**.

3.2 Penentuan UFP pada sistem ABOM

Tabel 3.3 UFP pada Sistem ABOM

Elemen		Jumlah Data	Tingkat
External Input Low = 25 ; Avg = 17 ; High = 0			
1	Standard Rate-	7	Low
2	Salary(National,Expatriate,Agreement)	6	Low
3	Standard Rate-Overtime National	6	Low
4	Standard Rate-Expatriate Expenses	6	Low
5	Standard Rate-Rest & Relaxation	6	Low
6	Standard Rate-Singapore Leave	9	Low
7	Standard Rate-Light Cars	10	Low
8	Standard Rate-PC & Printer	9	Low
9	Standard Rate-Fuel & Lubicant	7	Low
10	Standard Rate-Get Together	24	Avg
11	User Define Rate-Traning	22	Avg
12	User Define Rate-Business Travel	10	Low
13	User Define Rate-Miscellaneous Personel Expense	10	Low

Tabel 3.3a Lanjutan Tabel 3.3 UFP pada Sistem ABOM

14	User Define Rate-Contract Service	10	Low
15	User Define Rate-Rental/Lease	10	Low
16	User Define Rate-Chemical	10	Low
17	User Define Rate-Medical Supplies	11	Low
18	User Define Rate-Material & Supplies	11	Low
19	User Define Rate-Utilities	11	Low
20	User Define Rate-Technology Support	7	Low
21	User Define Rate-Safety,Health, & Environment	11	Low
22	User Define Rate-Service Charges	14	Low
23	User Define Rate-Food & Beverages	10	Low
24	User Define Rate-Assistance & Contribution	13	Low
25	User Define Rate-Insurance	8	Low
26	User Define Rate-Miscellaneous&Prior Year	8	Low
27	Expenses	25	Avg
28	User Define Rate-Transferred Recovered	16	Avg
29	Expense Project Budget	16	Avg
30	Redistributed-Salaries/Wages/Benefit	16	Avg
31	Redistributed-Rigs Usage	16	Avg
32	Redistributed-Helicopter Usage	16	Avg
33	Redistributed-Power/Electricity	16	Avg
34	Redistributed-Natural Gas	16	Avg
35	Redistributed-Camp & Community	16	Avg
36	Redistributed-Central Maintence Services	16	Avg
37	Redistributed-Equipment Usage	16	Avg
38	Redistributed-Flight Services	16	Avg
39	Redistributed-E&T Support Laboratory	16	Avg
40	Redistributed-Security	16	Avg
41	Redistributed-Chartered Hire	16	Avg
42	Redistributed-Expense Project Allocation-In	1	Low
	Redistributed-Allocation to Field Level Approval-by Account		
	External Output		
Low = 4 ; Avg = 0 ; High = 0			
1	Report-Budget Summary by Activity Grouping	8	Low
2	Report- Budget Summary by Expense Account	10	Low
3	Report- Budget Summary by Budget Account	11	Low
4	Report-Approved Budget	14	Low
	External Inquiries		
Low = 1 ; Avg = 0 ; High = 0			
1	Select-Year-Cost Center-PSC	4	Low
	Internal Logical Files		
Low = 31 ; Avg = 0 ; High = 0			
1	M Approval Acct	6	Low
2	M Program Based Detail	4	Low
3	M Projccalloc	7	Low

Tabel 3.3b Lanjutan Tabel 3.3a UFP pada Sistem ABOM

4	M Project	9	Low
5	M Redistributed	8	Low
6	M SR Fuel Lubricant	7	Low
7	M SR Overtime	7	Low
8	M SR Exptexp	7	Low
9	R SR Salary Percentage	5	Low
10	R SR Vihicle	3	Low
11	R System	2	Low
12	R UDR Code	2	Low
13	R UDR General	4	Low
14	R UDR Insurance	6	Low
15	R UDR Traning Combus	6	Low
16	Account List	6	Low
17	Audit	3	Low
18	PIK Descr	2	Low
19	T CMS Master	4	Low
20	Costcenter	6	Low
21	Email ID	5	Low
22	Hierarchy CC	9	Low
23	Position	6	Low
24	M Program Based	14	Low
25	M Program Pcand Printers	10	Low
26	R SR Salary Overtime	10	Low
27	Project Plan	19	Low
28	T CMS Detail	10	Low
29	CMS Historical	11	Low
30	THABOM	18	Low
31	Project Ctrl	21	Low
External Interface Files			
Low = 3 ; Avg = 0 ; High = 0			
1	Interface-from LFAIS	9	Low
2	Interface-to LFAIS	15	Low
3	Interface-to IBIS	13	Low

3.3 Penentuan UFP pada sistem PLF

Tabel 3.4 UFP pada Sistem PLF

Elemen		Jumlah Data	Tingkat
External Input			
Low = 0 ; Avg = 0 ; High = 0			
1	-		
External Output			
Low = 0 ; Avg = 0 ; High = 0			
1	-		
External Inquiries			
Low = 0 ; Avg = 0 ; High = 0			
1	-		
Internal Logical Files			
Low = 28 ; Avg = 0 ; High = 0			
1	Cost Center Group	5	Low
2	Account Major	3	Low
3	Other Use	3	Low
4	Own Use	7	Low
5	Lifting Cost Summary	10	Low
6	Area LC	6	Low
7	SBU LC	5	Low
8	Corp LC	5	Low
9	Oil Price	5	Low
10	GL 4000	10	Low
11	Pipeline	7	Low
12	Rule Field	6	Low
13	Rules	3	Low
14	Account Rule	2	Low
15	Error Alloc	7	Low
16	HCT Cost	6	Low
17	Feeder KWH	4	Low
18	Account List	6	Low
19	FOE LC	9	Low
20	Pipe Field	2	Low
21	PGNT Cost	6	Low
22	Field Team	2	Low
23	Feeder Field	2	Low
24	Field List	5	Low
25	Conversion	5	Low
26	Field Prod	8	Low
27	Field Gas	8	Low
28	Data Status	9	Low

3.4 Penentuan UFP pada sistem AFE-BIRDS

Tabel 3.5 UFP pada Sistem AFE-BIRDS

Elemen		Jumlah Data	Tingkat
External Input			
Low = 6 ; Avg = 0 ; High = 0			
1	Utilities-Update Portofolio PM	9	Low
2	Utilities-Portofolio AFE Xref	7	Low
3	Utilities-Copy Portofolio	4	Low
4	Utilities-AFE Profile Definition	3	Low
5	Utilities-Email AFE Status	8	Low
External Output			
Low = 14 ; Avg = 6 ; High = 0			
1	Report-AFE Status by Organization	26	Avg
2	Report-AFE status by GOI AFE	23	Avg
3	Report-AFE Status by Project Controler	19	Low
4	Report-AFE Status by Hierarchy	12	Low
5	Report-AFE Status by Portofolio	19	Low
6	Report-Preproductive Capital	20	Avg
7	Report-GOI AFE Overrun Summary	10	Low
8	Report-GOI AFE Overrun Anaysis	12	Low
9	Report-AFE Inventory List	11	Low
10	Report-GOI AFE Inventory List	12	Low
11	Report-CEB Reporting	30	Avg
12	Report-Detail AFE List	14	Low
13	Report-AFE Comments	4	Low
14	Report-AFE Detail Activities	14	Low
15	Report-AFE Detail Activities (2003)	14	Low
16	Report-AFE Monthly Summary Analysis	10	Low
17	Report-User Access Statistics	4	Low
18	Report-Detail Log Activity	5	Low
19	Links-AFE Summary Download	20	Avg
20	Links-Archived Project Status	23	Avg
External Inquiries			
Low = 1 ; Avg = 1 ; High = 0			
1	Inquiry-AFE Hierarchy	30	Avg
2	Inquiry-Portofolio	18	Low
Internal Logical Files			
Low = 32 ; Avg = 2 ; High = 0			
1	AFEBIRDS Comments	12	Low
2	DIM Addressbook	4	Low
3	DIM AFE Costcode	1	Low
4	DIM AFE Costtype	3	Low

Tabel 3.5a Lanjutan Tabel 3.5 UFP pada sistem AFE-BIRDS

5	DIM AFE Costtype Costcode	4	Low
6	DIM AFE Government	2	Low
7	DIM AFE Organization	43	Low
8	DIM AFE Status	2	Low
9	DIM BU Catcd04	3	Low
10	DIM BU Detl	22	Low
11	DIM Company	2	Low
12	DIM Costcode	2	Low
13	DIM GOIAFE	20	Low
14	Employee Profile	26	Low
15	Fact F0902	44	Low
16	Fact F0911	116	Avg
17	Fact GOIAFE Overrun	10	Low
18	Fact GOIAFE Overrun Monthly	78	Avg
19	Log Activity	4	Low
20	Select Date	27	Low
21	User Profile	3	Low
22	WlkrProjExpnd	31	Low
23	Work AFEBALAA	17	Low
24	Work AFEBALAAI	17	Low
25	Work AFEBALAAT	17	Low
26	Work AFEBALCRI	17	Low
27	Work AFEBALCRT	17	Low
28	Work AFEBALDA	18	Low
29	Work AFEBALJA	19	Low
30	Work AFEHEMAT4	22	Low
31	Work AFEMailDef	9	Low
32	Work AFEMailDtl	11	Low
33	Work AFEMailDtl Temp	11	Low
34	Work Xref PortAFE	2	Low
External Interface Files			
Low = 0 ; Avg = 1 ; High =0			
1	Extracted Data from LFAIS/JDE	127	Avg

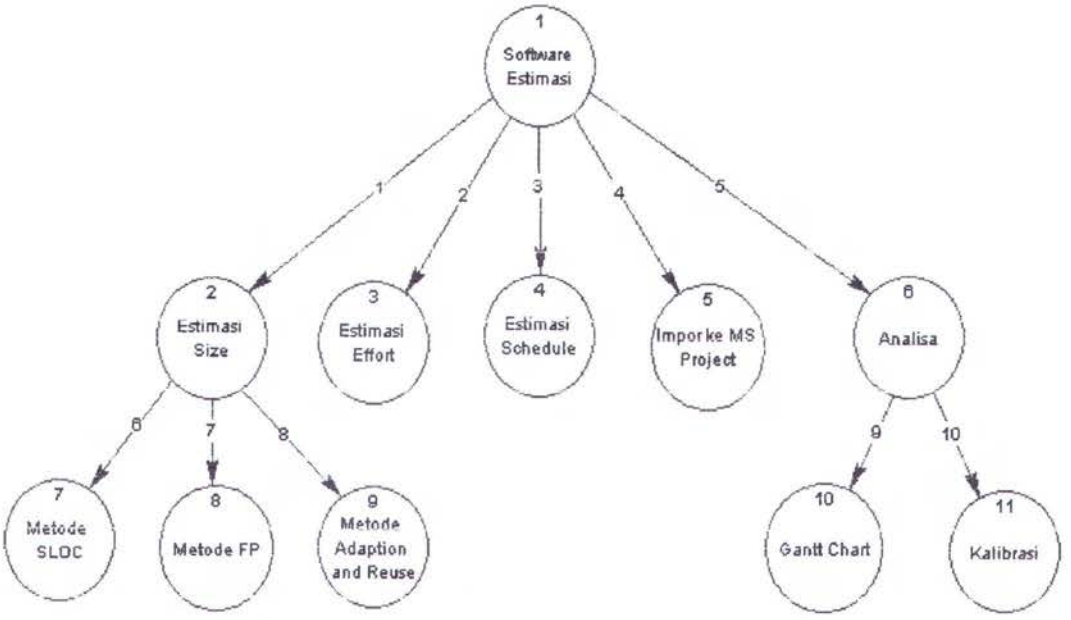
3.5 Penentuan UFP pada sistem FAM

Tabel 3.6 UFP pada Sistem FAM

Elemen		Jumlah Data	Tingkat
External Input			
Low = 0 ; Avg = 6 ; High =2			
1	New Asset Attribute	18	High
2	Asset Key	1	Avg
3	Asset Adjustment Factor	4	Avg
4	Reason for Asset Change	1	Avg
5	Asset Location	2	Avg
6	Asset Owner	3	Avg
7	User Approval	2	Avg
8	Asset Attribute Changes	13	High
External Output			
Low = 0 ; Avg = 0 ; High =3			
1	Display To be Asset	9	High
2	Display Asset Information	18	High
3	Display Error Message	8	High
External Inquiries			
Low = 0 ; Avg = 0 ; High =6			
1	General Inquiry	8	High
2	Financial Book Inquiry	15	High
3	Journal Entry Inquiry	7	High
4	Historical Inquiry	7	High
5	Asset Listing Inquiry	20	High
6	Inventory Inquiry	6	High
Internal Logical Files			
Low = 0 ; Avg = 0 ; High =9			
1	Location		High
2	Account		High
3	GL Entry		High
4	Class		High
5	Asset Master		High
6	Organization		High
7	Tax Category		High
8	Historical		High
9	Depreciation Rate		High
External Interface Files			
Low = 0 ; Avg = 3 ; High =1			
1	New Asset from Project Close Out	6	Avg
2	Asset Late Charges from Project Close Out	3	Avg
3	Journal Entry to General Ledger	32	High
4	Feeder Harmoni II System	16	Avg

3.6 Bagan Berjenjang Software Estimasi

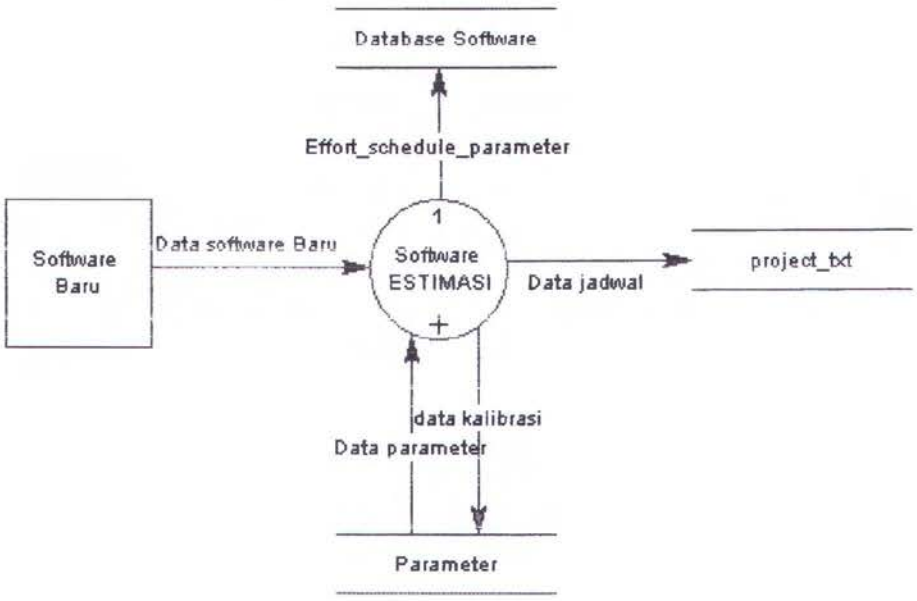
Bagan berjenjang (gambar 3.1) mendeskripsikan level - level diagram aliran secara keseluruhan. Pada bagan ini terlihat pembagian dari level - level yang nantinya dapat dipakai dalam penyusunan menu. Terlihat disini fungsi software ini adalah untuk mengestimasi *size*, *effort*, dan *schedule*. Software ini juga dapat digunakan untuk analisa dengan gantt chart dan dapat dikalibrasi, serta dapat diimpor ke bentuk data MS Project.



Gambar 3.1 Bagan Berjenjang Software Estimasi

3.7 Context Diagram Software Estimasi

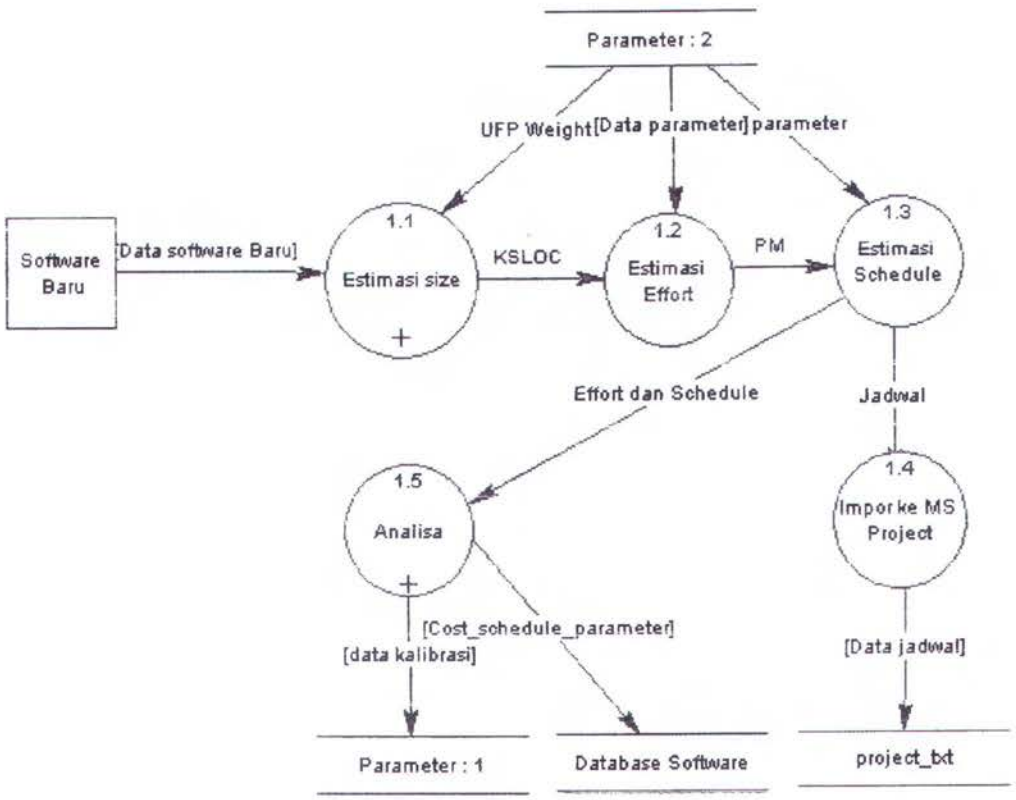
Context Diagram (gambar 3.2) menggambarkan perangkat lunak secara global. Dari sini terlihat bahwa input yang dibutuhkan adalah data dari software baru yang akan dibuat dan parameter dasar yang telah ada. Sedangkan outputnya adalah berupa database software, file text yang dapat dibuka dengan MS Project dan data kalibrasi ulang untuk mengupdate parameter.



Gambar 3.2 Context Diagram Software Estimasi

3.8 Diagram Arus Data level 0 Software Estimasi

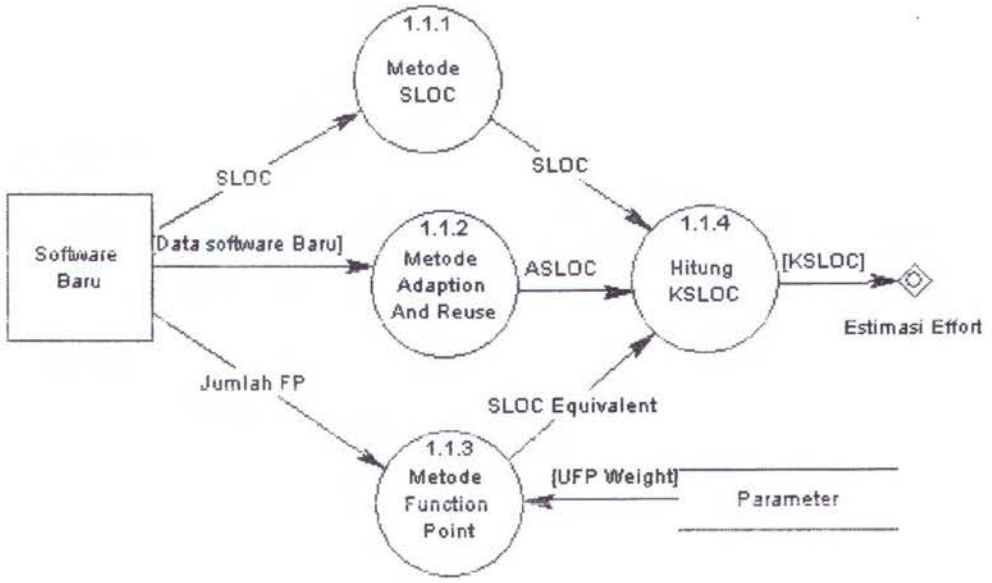
Pada level 0 (gambar 3.3) Software Estimasi diuraikan menjadi Estimasi Size, Estimasi Effort, Estimasi Schedule, Impor ke MS Project, dan Analisa. Estimasi Size adalah langkah pertama yaitu menghitung ukuran baris kode program yang akan dibuat dalam satuan KSLOC.



Gambar 3.3 DAD Level 0 Software Estimasi

3.8.1 Diagram Arus Data Level 1 Estimasi *Size*

Pada level 1 (gambar 3.4) Estimasi Size diuraikan menjadi Metode SLOC, Metode *Adaption and Reuse*, Metode *Function Point* dan Hitung KSLOC. Metode SLOC digunakan bila input sudah berupa jumlah baris kode program (SLOC). Metode *Function Point* digunakan bila input berupa jumlah dari Function Point yang dikalikan dengan parameter *Function Point Weight* menjadi SLOC *equivalent*. Metode *Adaption dan Reuse* digunakan bila input berupa data program Reuse dan persentasenya yang akan dikonversikan menjadi *Adaption* SLOC (ASLOC). Hasil ketiga metode tersebut oleh Hitung KSLOC dikalikan dengan REVL dan dibagi dengan 1000 dan menjadi KSLOC.



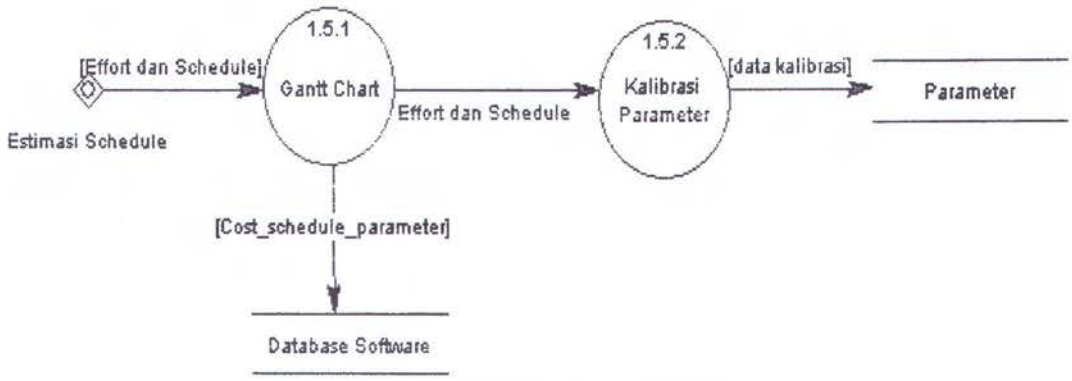
Gambar 3.4 DAD Level 1 Estimasi *Size*

Estimasi Effort adalah penghitungan jumlah effort yang dibutuhkan dalam PM (*Person Months*). Perhitungan effort menggunakan input KSLOC dan parameter berupa konstanta, scale factor, dan cost driver. Estimasi *Schedule* adalah penentuan lama waktu software akan dikembangkan dalam hitungan bulan. Penentuan jadwal ini membutuhkan input PM dan parameter berupa konstanta dan cost driver. Impor ke MS Project adalah fungsi tambahan untuk mengimpor data jadwal menjadi file teks yang dapat digunakan oleh MS Project. Analisa adalah modul untuk menganalisa jadwal yang dihasilkan berupa gantt chart dan untuk mengkalibrasi parameter bila diperlukan.

3.8.2 Diagram Arus Data Level 1 Analisa

Pada level 1 (gambar 3.5) Analisa diuraikan menjadi Gantt Chart dan Kalibrasi Parameter. Gantt Chart adalah grafik dari jadwal yang dihasilkan agar mudah dianalisa. Kalibrasi Parameter digunakan bila akan melakukan kalibrasi ulang parameter yang ada bila diperlukan. Metode kalibrasi yang digunakan adalah metode

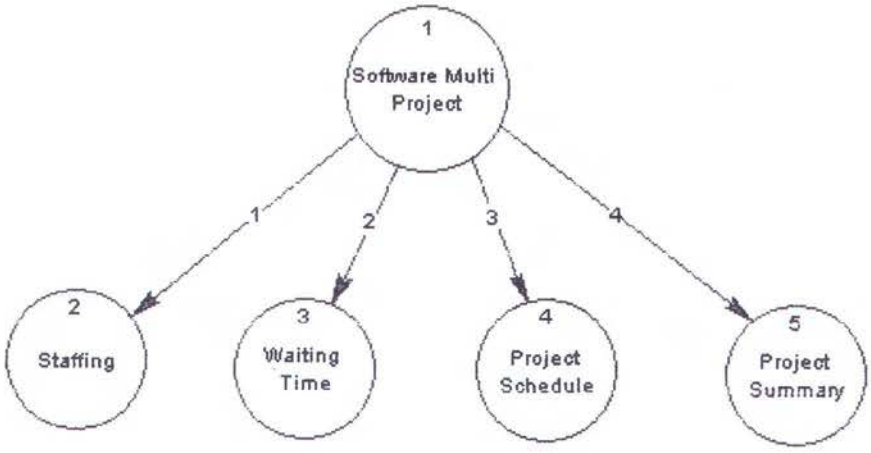
Log Natural (Ln) dan Bayesian. Konstanta yang dikalibrasi dengan metode tersebut adalah konstanta A dan konstanta C. Konstanta A berpengaruh pada estimasi effort, sedangkan konstanta C berpengaruh pada estimasi schedule.



Gambar 3.5 DAD Level 1 Analisa

3.9 Bagan Berjenjang Software *Multi Schedule*

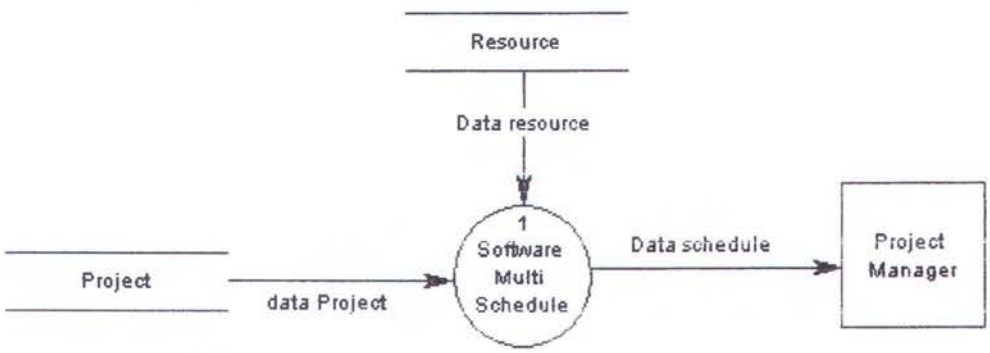
Bagan berjenjang (gambar 3.6) mendeskripsikan level - level diagram aliran secara keseluruhan. Pada bagan ini terlihat pembagian dari level - level yang nantinya dapat dipakai dalam penyusunan menu. Terlihat disini bahwa software ini digunakan untuk membuat jadwal multi proyek dengan menghitung staf dan waktu tunggu (*waiting time*) untuk tiap proyek.



Gambar 3.6 Bagan Berjenjang Software *Multi Schedule*

3.10 Context Diagram Software Multi Schedule

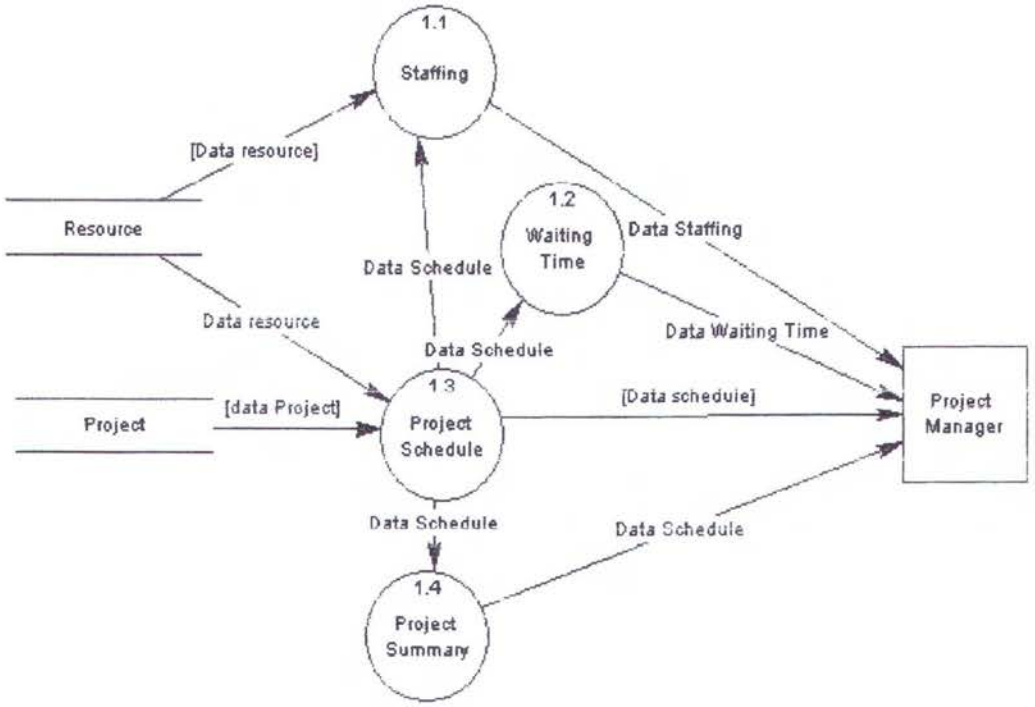
Context Diagram (gambar 3.7) menggambarkan perangkat lunak secara global. Dari sini terlihat bahwa input yang dibutuhkan adalah data dari database project yang telah dibuat dengan software Estimasi. Sedangkan outputnya adalah berupa tampilan jadwal yang digunakan manajer proyek untuk menganalisa jadwal yang telah dibuat dan kebutuhan biaya dari sumber dayanya.



Gambar 3.7 Context Diagram Software *Multi Schedule*

3.11 Diagram Arus Data level 0 Software *Multi Schedule*

Pada level 0 (gambar 3.8) Software *Multi Schedule* diuraikan menjadi *staffing*, *Waiting Time*, *Project Schedule* dan *Project Summary*. *Staffing* digunakan untuk menganalisa kebutuhan sumber daya dan biayanya. *Waiting Time* digunakan untuk menganalisa waktu tunggu masing-masing proyek. *Project Schedule* digunakan untuk membuat jadwal dari data project dan data resource, dimana sumber daya yang digunakan untuk membuat jadwal multi proyek terbatas. Sedang *Project Summary* berisi ringkasan dari semua proyek yang ada.



Gambar 3.8 DAD Level 0 Software *Multi Schedule*

BAB IV

HASIL DAN PEMBAHASAN

Pada bab ini akan diimplementasikan dua buah software, yang pertama digunakan untuk mengestimasi effort dan schedule dengan metode COCOMO, beserta kalibrasinya, dan yang kedua digunakan untuk menjadwal secara multi proyek dengan metode heuristik. Sebagai ujicoba dimasukkan data yang terdapat pada empat software yang dikembangkan oleh PT X yang telah dianalisa pada bab sebelumnya.

4.1 Estimasi Software ABOM

Dari hasil kuisioner sistem ABOM didapatkan data PM aktual (24.5 PM), TDEV aktual (6 bulan), Scale Factor dan cost driver seperti pada tabel 4.1. Scale Factor (SF) yang didapat kemudian dimasukkan ke software estimasi seperti pada gambar 4.1, sedangkan cost drivernya (EAF) seperti pada gambar 4.2. Dari sini didapatkan nilai SF : **16.61**, sedangkan nilai EAF adalah : **0.249.**

Tabel 4.1 Scale Factor dan Cost Driver Software ABOM

Form SPD-5a COCOMO II Project Actuals: Simple Completed Project

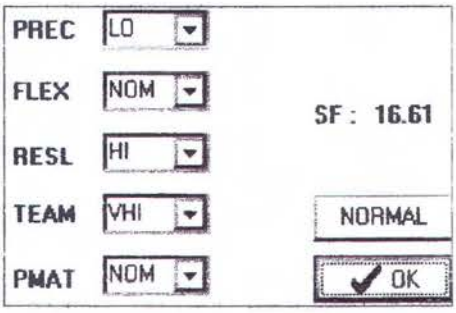
1. Project Title: ABOM 2. Project ID No.: 01 3. Rev No.:
4. Date Prepared: 18 Mei 05 5. Originator: Ade D. Hardiana 6. Organization: PT CPI

7. Starting Milestone: End of Req Def 8. Ending Milestone: End of User Acceptance
9. Total no. of person-months: 24, 5 PM 10. Total no. of calendar months: 6 Months
11. Equivalent SLOC: 12. Total no. of SLOC reused:
13. Non-trivial defects detected: 14. Defect detection starting milestone:

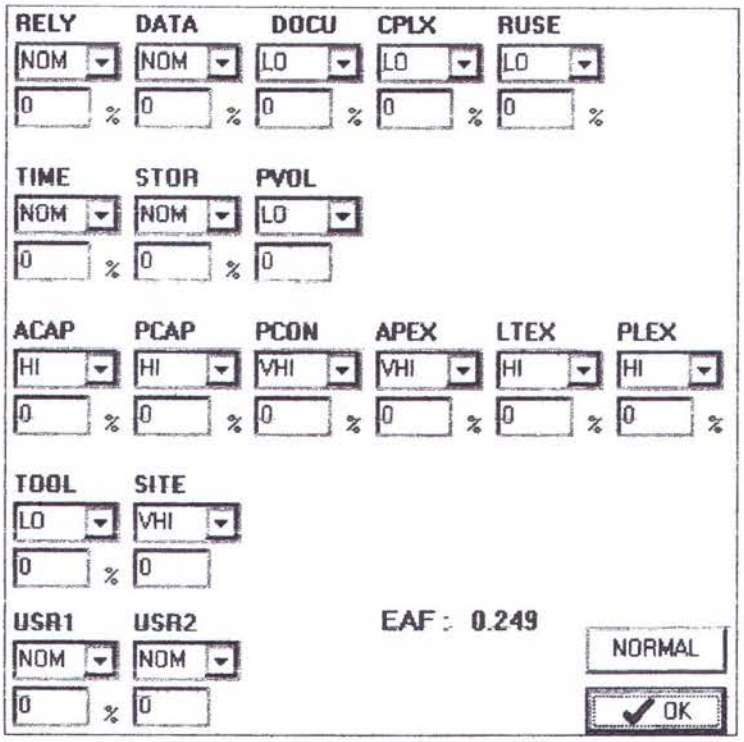
15. Project attribute ratings

	VL	L	N	H	VH	XH	Comments (Including Don't Kn
Precedentedness (PREC)		●					
Development flexibility (FLEX)			●				
Architecture/risk resolution (RESL)				●			
Team cohesion (TEAM)					●		
Process maturity (PMAT)			●				
Required reliability (RELY)			●				
Data base size (DATA)			●				
Product complexity (CPLX)		●					
Develop for reuse (RUSE)		●					
Documentation match to life-cycle needs (DOCU)		●					
Execution time constraint (TIME)			●				
Main storage constraint (STOR)			●				
Platform volatility (PVOL)		●					
Analyst capability (ACAP)				●			
Programmer capability (PCAP)				●			
Personnel continuity (PCON)					●		
Applications experience (APEX)					●		
Platform experience (PLEX)				●			
Language & tool experience (LTEX)				●			
Use of software tools (TOOL)		●					
Multi-site development (SITE)					●		
Required development schedule (SCED)			●				
Other							

16. Special project characteristics or lessons learned:

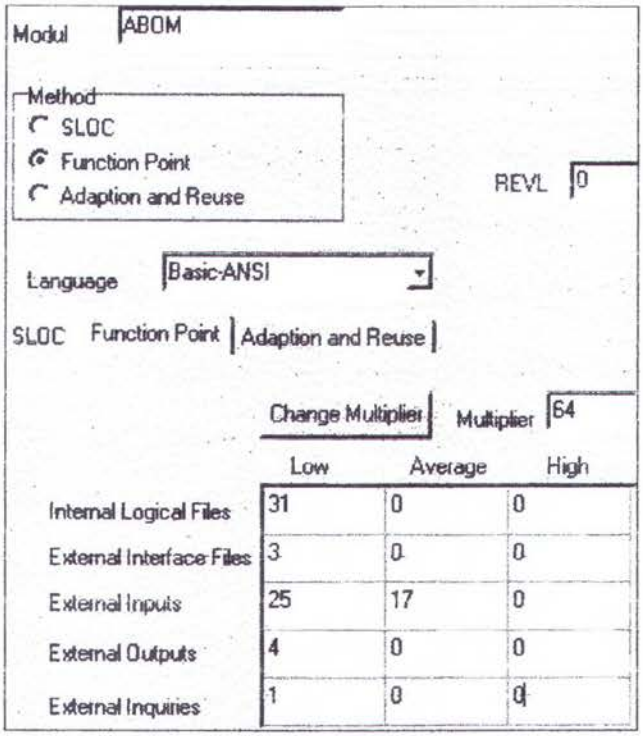


Gambar 4.1 Scale Factor Software ABOM



Gambar 4.2 Cost Driver Software ABOM

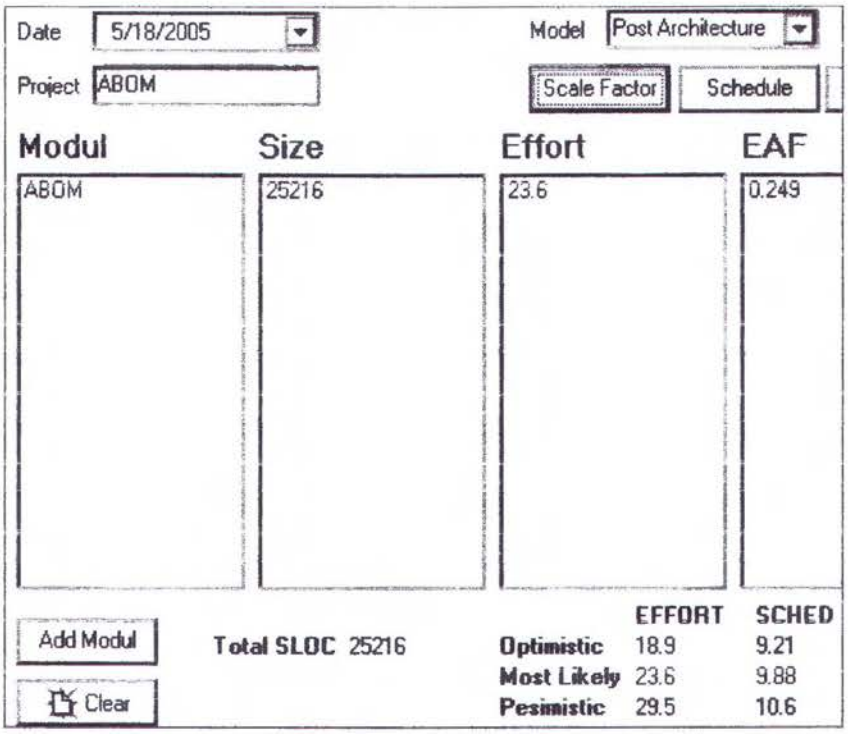
Langkah selanjutkan dimasukkan data size dengan metode Function Point seperti pada gambar 4.3. Nilai Function Point didapat dari analisa di bab sebelumnya (bab III). Sistem ini ditulis dengan bahasa CFM (*ColdFusion*) yang secara sintaks mirip dengan **Basic-ANSI**, dimana Basic-ANSI memiliki nilai pengali (*multiplier)* **64**. Nilai pengali ini dikalikan dengan semua nilai Function Point, lalu dijumlahkan.



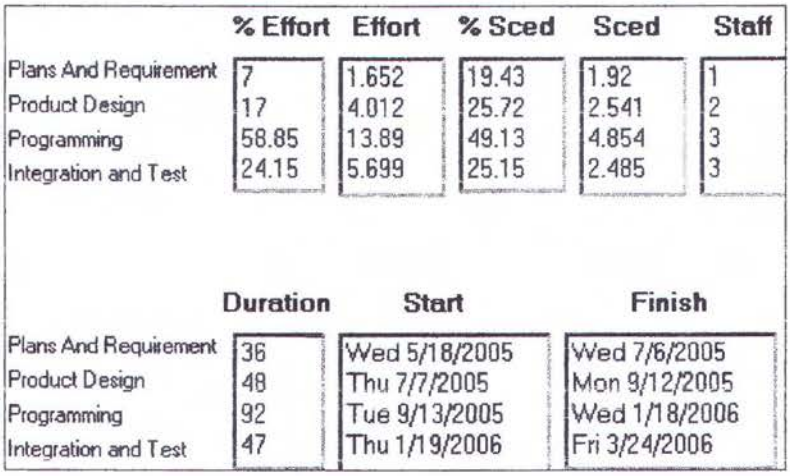
Gambar 4.3 UFP Software ABOM

Dari pemasukan function point ini dapat dihitung SLOC sistem, yaitu **25216**, seperti pada gambar 4.4. Setelah diketahui SLOC maka dapat dihitung nilai effort : **23.6** PM dan schedule : **9.88** bulan. Dihitung juga effort dan schedule dengan persentase optimistic (80%) dan pesimistic (125%).

Effort dan schedule dapat dirinci menjadi tahapan-tahapan *life-cycle waterfall*, dimana masing-masing tahapan mempunyai persentase, seperti pada gambar 4.5. Dari masing-masing tahapan dihitung menurut waktu kalender dalam satuan hari dan jumlah staf (dibulatkan) yang dibutuhkan.

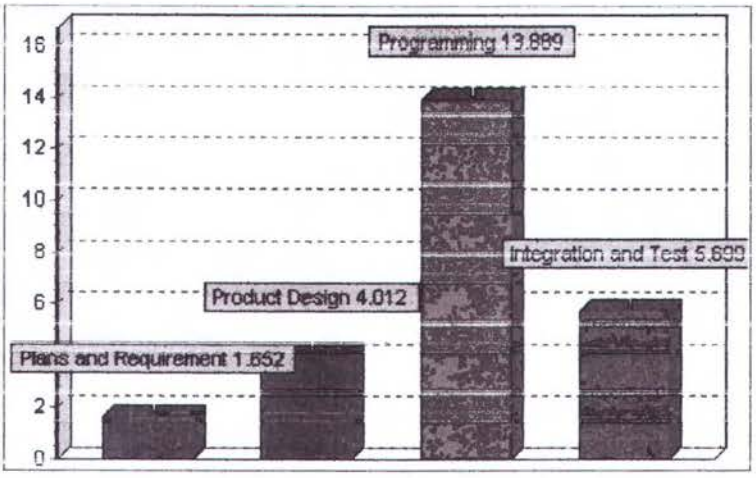


Gambar 4.4 Effort dan Jadwal Software ABOM

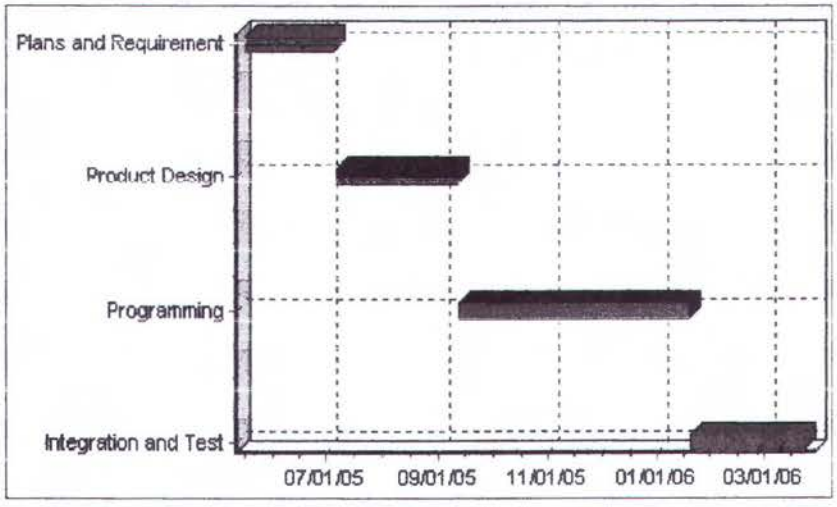


Gambar 4.5 Jadwal Tiap Tahap Software ABOM

Dibuat juga diagram batang dari effort untuk masing-masing tahap seperti pada gambar 4.6 dan gantt chart dari jadwal seperti pada gambar 4.7. Terakhir apabila dibutuhkan data dapat dikonversi dalam bentuk text yang dapat dibaca oleh MS Project, seperti pada gambar 4.8.



Gambar 4.6 Diagram Batang Effort Setiap Tahap Software ABOM



Gambar 4.7 Gantt Chart Software ABOM

Task Name	Duration	Start	Finish
Plans and Requirement	36 days	Wed 5/18/05	Wed 7/6/05
Product Design	48 days	Thu 7/7/05	Mon 9/12/05
Programming	92 days	Tue 9/13/05	Wed 1/18/06
Integration and Test	47 days	Thu 1/19/06	Fri 3/24/06

Gambar 4.8 Hasil Konversi Software ABOM ke MS Project

4.2 Estimasi software PLF

Tabel 4.2 Scale Factor dan Cost Driver Software PLF

Form SPD-5a COCOMO II Project Actuals: Simple Completed Project

1. Project Title: 2. Project ID No. 3. Rev No.
4. Date Prepared: 5. Originator: 6. Organization:

7. Starting Milestone: 8. Ending Milestone:
9. Total no. of person-months: 10. Total no. of calendar months:
11. Equivalent SLOC: 12. Total no. of SLOC reused:
13. Non-trivial defects detected: 14. Defect detection starting milestone:

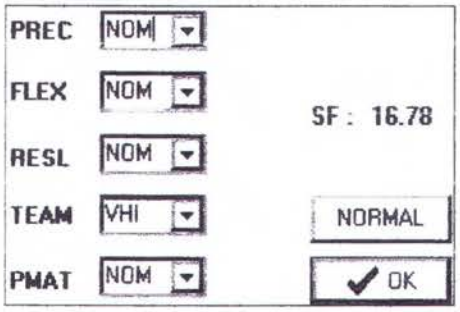
15. Project attribute ratings

	VL	L	N	H	VH	XH	Comments (including Don't Kn
Precedentedness (PREC)			●				
Development flexibility (FLEX)			●				
Architecture/risk resolution (RESL)			●				
Team cohesion (TEAM)					●		
Process maturity (PMAT)			●				
Required reliability (RELY)			●				
Data base size (DATA)			●				
Product complexity (CPLX)		●					
Develop for reuse (RUSE)		●					
Documentation match to life-cycle needs (DOCU)		●					
Execution time constraint (TIME)			●				
Main storage constraint (STOR)			●				
Platform volatility (PVOL)		●					
Analyst capability (ACAP)				●			
Programmer capability (PCAP)				●			
Personnel continuity (PCON)					●		
Applications experience (APEX)					●		
Platform experience (PLEX)				●			
Language & tool experience (LTEX)				●			
Use of software tools (TOOL)		●					
Multi-site development (SITE)					●		
Required development schedule (SCED)			●				
Other							

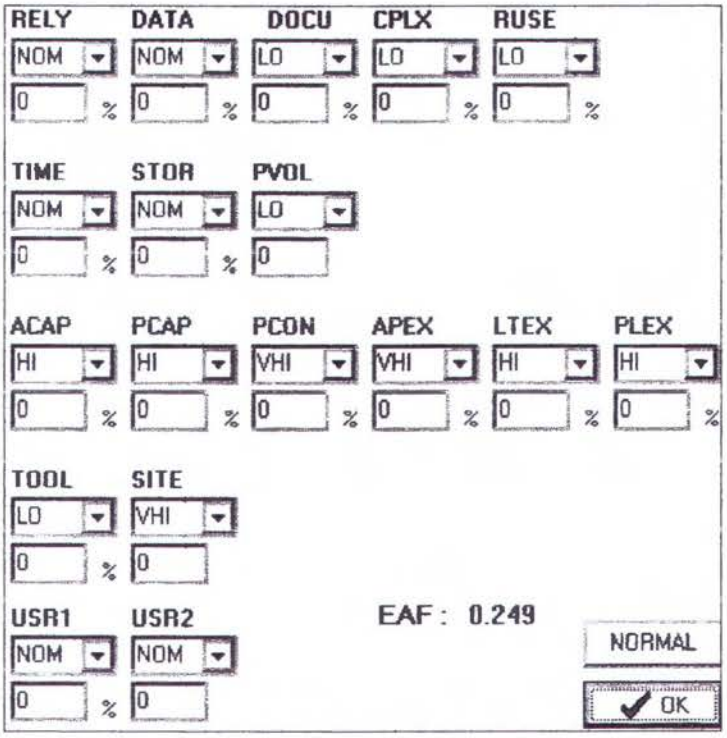
16. Special project characteristics or lessons learned:

Dari hasil kuisioner sistem PLF didapatkan PM aktual, TDEV aktual, Scale Factor dan cost driver seperti pada tabel 4.2. Scale Factor (SF) yang didapat kemudian dimasukkan ke software estimasi seperti pada gambar 4.9, sedangkan cost

drivernya (EAF) seperti pada gambar 4.10. Dari sini didapatkan nilai SF : **16.78**, sedangkan nilai EM adalah : **0.249.**



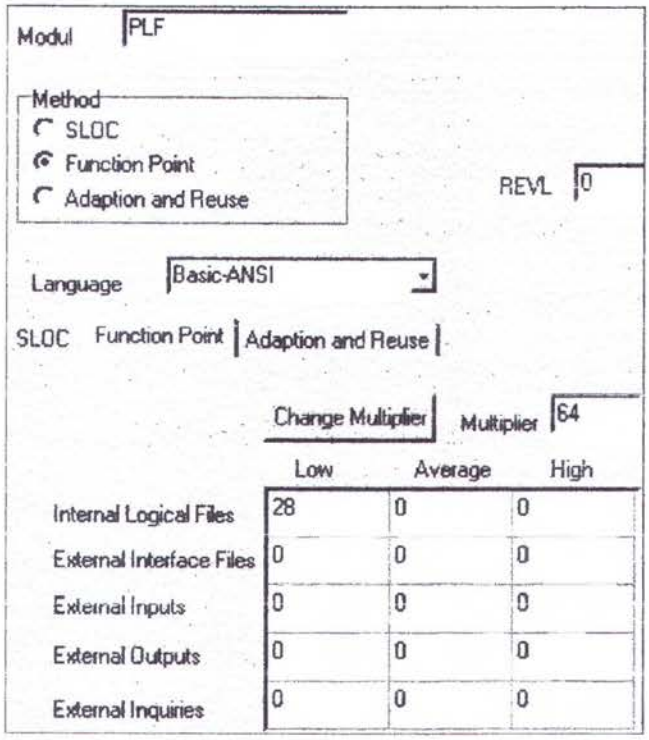
Gambar 4.9 Scale Factor Software PLF



Gambar 4.10 Cost Driver Software PLF

Langkah selanjutkan dimasukkan data size dengan metode Function Point seperti pada gambar 4.11. Nilai Function Point didapat dari analisa di bab sebelumnya (bab III). Sistem ini ditulis dengan bahasa CFM (*ColdFusion*) yang

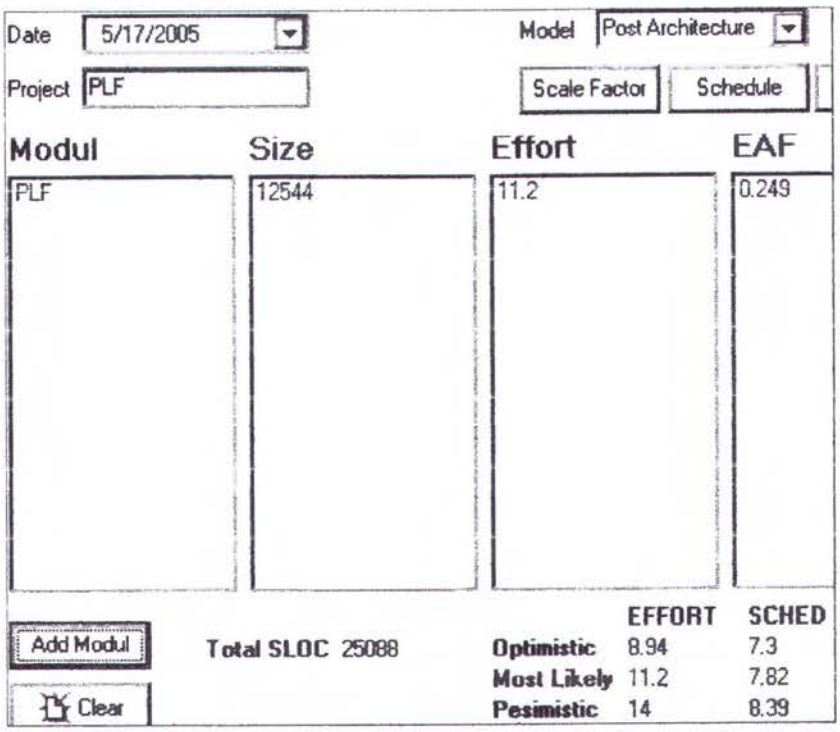
secara sintaks mirip dengan **Basic-ANSI**, dimana Basic-ANSI memiliki *multiplier* **64**. Semua nilai Function Point dikalikan dengan multiplier.



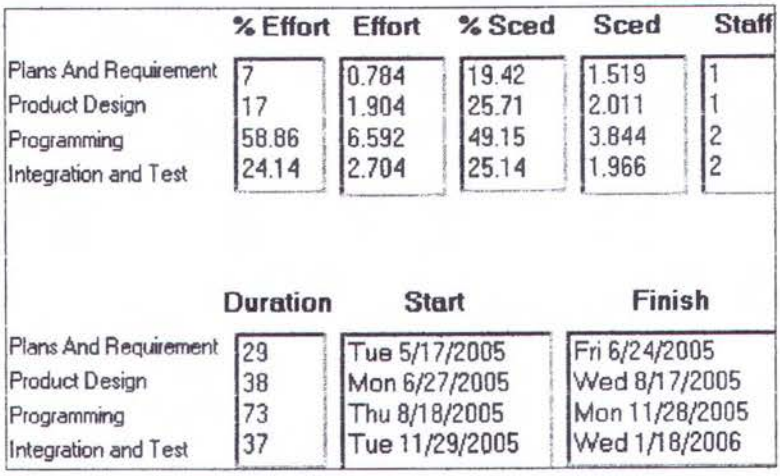
Gambar 4.11 UFP Software PLF

Dari pemasukan function point ini dapat dihitung SLOC sistem, yaitu **12544**, seperti pada gambar 4.12. Setelah diketahui SLOC maka dapat dihitung nilai effort : **11.2** PM dan schedule : **7.82** bulan. Dihitung juga effort dan schedule dengan persentase optimistic (80%) dan pesimistic (125%).

Effort dan schedule dapat dirinci menjadi tahapan-tahapan *life-cycle waterfall*, dimana masing-masing tahapan mempunyai persentase, seperti pada gambar 4.13. Dari masing-masing tahapan dihitung menurut waktu kalender dalam satuan hari dan jumlah staf (dibulatkan) yang dibutuhkan.

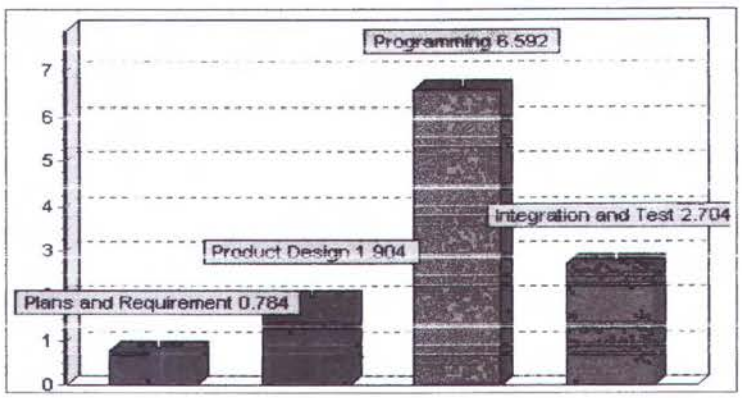


Gambar 4.12 Effort dan Jadwal Software PLF

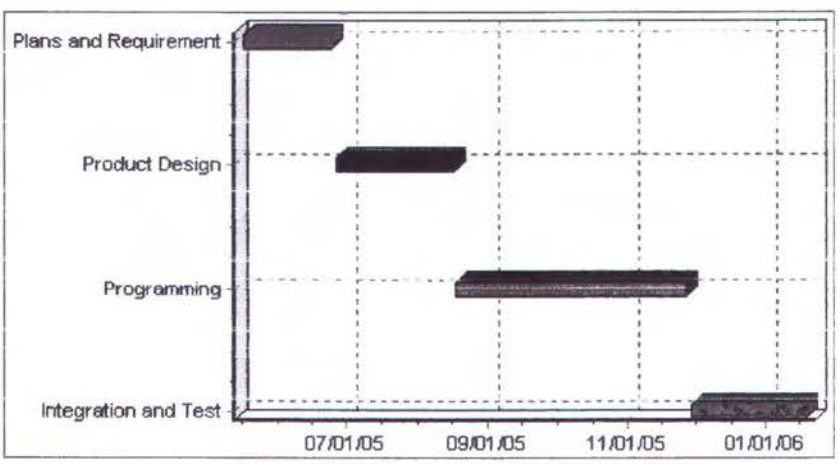


Gambar 4.13 Jadwal Tiap Tahap Software PLF

Dibuat juga diagram batang dari effort untuk masing-masing tahap seperti pada gambar 4.14 dan gantt chart dari jadwal seperti pada gambar 4.15. Terakhir apabila dibutuhkan data dapat dikonversi dalam bentuk text yang dapat dibaca oleh MS Project, seperti pada gambar 4.16.



Gambar 4.14 Diagram Batang Effort Setiap Tahap Software PLF



Gambar 4.15 Gantt Chart Software PLF

Task Name	Duration	Start	Finish	Predecessors
Plans and Requirement	27 days	Tue 5/17/05	Wed 6/22/05	
Product Design	37 days	Thu 6/23/05	Fri 8/12/05	1
Programming	76 days	Mon 8/15/05	Mon 11/28/05	2
Integration and Test	35 days	Tue 11/29/05	Mon 1/16/06	3

Gambar 4.16 Hasil Konversi Software PLF ke MS Project

4.3 Estimasi Software AFE-BIRDS

Tabel 4.3 Scale Factor dan Cost Driver Software **AFE-BIRDS**

Form SPD-5a COCOMO II Project Actuals: Simple Completed Project

1. Project Title: AFE-BIRDS 2. Project ID No.: 02 3. Rev No.:
4. Date Prepared: 21/05/08 Originator: Poltak P. Sanov 6. Organization: PT CPI
7. Starting Milestone: End of Rqm Def 8. Ending Milestone: End of User Acceptance
9. Total no. of person-months: 12 PM 10. Total no. of calendar months: 6 Months
11. Equivalent SLOC: 106K 12. Total no. of SLOC reused:
13. Non-trivial defects detected: 14. Defect detection starting milestone:

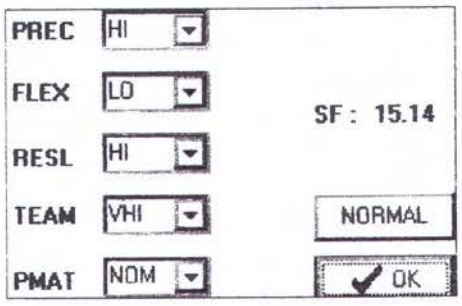
15. Project attribute ratings

	VL	L	N	H	VH	XH	Comments (Including Don't Kn
Precedentedness (PREC)				●			
Development flexibility (FLEX)		●					
Architecture/risk resolution (RESL)				●			
Team cohesion (TEAM)					●		
Process maturity (PMAT)			●				
Required reliability (RELY)				●			
Data base size (DATA)					●		
Product complexity (CPLX)			●				
Develop for reuse (RUSE)				●			
Documentation match to life-cycle needs (DOCU)				●			
Execution time constraint (TIME)			●				
Main storage constraint (STOR)			●				
Platform volatility (PVOL)		●					
Analyst capability (ACAP)				●			
Programmer capability (PCAP)				●			
Personnel continuity (PCON)		●					
Applications experience (APEX)				●			
Platform experience (PLEX)				●			
Language & tool experience (LTEX)				●			
Use of software tools (TOOL)		●					
Multi-site development (SITE)					●		
Required development schedule (SCED)			●				
Other							

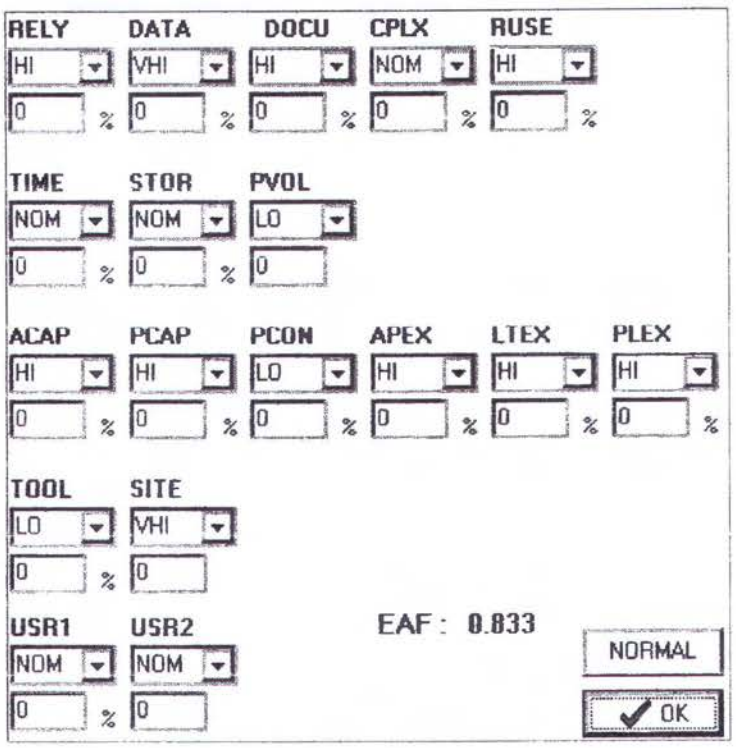
16. Special project characteristics or lessons learned:

Dari hasil kuisioner sistem AFE-BIRDS didapatkan PM aktual, TDEV aktual, Scale Factor dan cost driver seperti pada tabel 4.3. Scale Factor (SF) yang didapat kemudian dimasukkan ke software estimasi seperti pada gambar 4.17, sedangkan

cost drivernya (EAF) seperti pada gambar 4.18. Dari sini didapatkan nilai SF : **15.14**, sedangkan nilai EAF adalah : **0.833.**



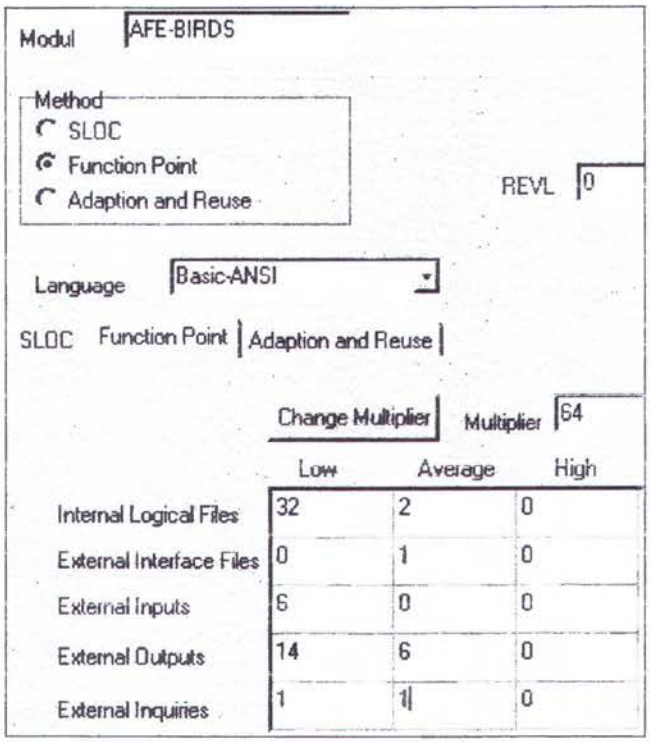
Gambar 4.17 Scale Factor Software AFE-BIRDS



Gambar 4.18 Cost Driver Software AFE-BIRDS

Langkah selanjutkan dimasukkan data size dengan metode Function Point seperti pada gambar 4.19. Nilai Function Point didapat dari analisa di bab sebelumnya. Sistem ini ditulis dengan bahasa CFM (*ColdFusion*) yang secara sintaks mirip

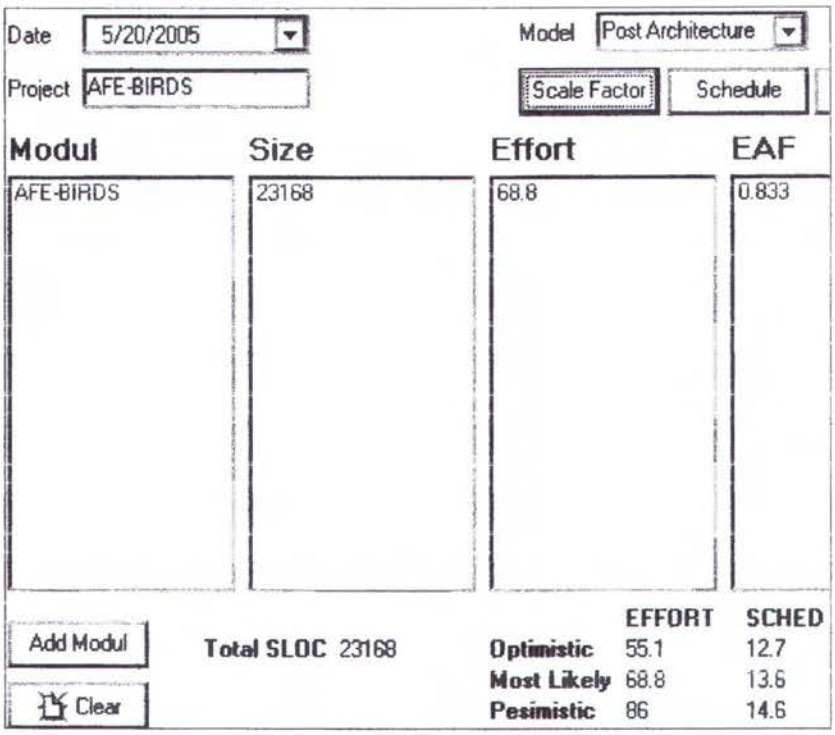
dengan **Basic-ANSI**, dimana Basic-ANSI memiliki nilai pengali (*multiplier)* **64**. Multiplier ini untuk mengalikan semua nilai Function Point.



Gambar 4.19 UFP Software AFE-BIRDS

Dari pemasukan function point ini dapat dihitung SLOC sistem, yaitu **23168**, seperti pada gambar 4.20. Setelah diketahui SLOC maka dapat dihitung nilai effort : **68.8** PM dan schedule : **13.6** bulan. Dihitung juga effort dan schedule dengan persentase optimistic (80%) dan pesimistic (125%)..

Effort dan schedule dapat dirinci menjadi tahapan-tahapan *life-cycle waterfall*, dimana masing-masing tahapan mempunyai persentase, seperti pada gambar 4.21. Dari masing-masing tahapan dihitung menurut waktu kalender dalam satuan hari dan jumlah staf (dibulatkan) yang dibutuhkan.



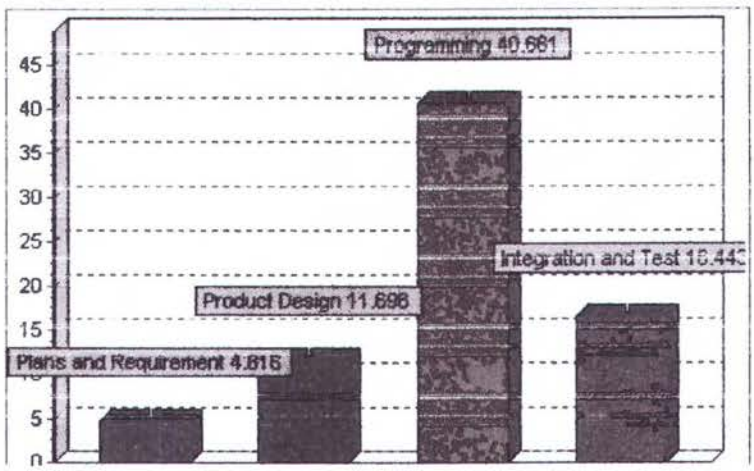
Gambar 4.20 Effort dan Jadwal Software AFE-BIRDS

	% Effort	Effort	% Sced	Sced	Staff
Plans And Requirement	7	4.816	19.26	2.619	2
Product Design	17	11.7	25.63	3.486	4
Programming	59.1	40.66	49.47	6.728	7
Integration and Test	23.9	16.44	24.9	3.386	5

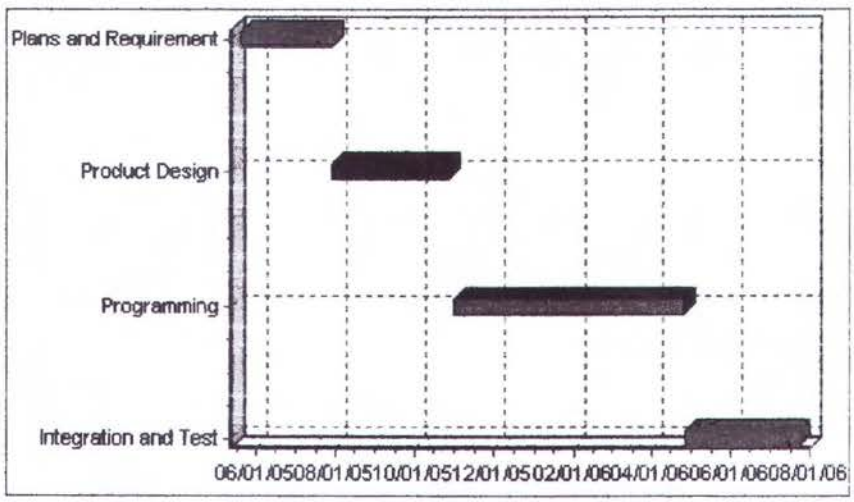
	Duration	Start	Finish
Plans And Requirement	50	Fri 5/20/2005	Thu 7/28/2005
Product Design	66	Fri 7/29/2005	Fri 10/28/2005
Programming	128	Mon 10/31/2005	Wed 4/26/2006
Integration and Test	64	Thu 4/27/2006	Tue 7/25/2006

Gambar 4.21 Jadwal Tiap Tahap Software AFE-BIRDS

Dibuat juga diagram batang dari effort untuk masing-masing tahap seperti pada gambar 4.22 dan gantt chart dari jadwal seperti pada gambar 4.23. Terakhir apabila dibutuhkan data dapat dikonversi dalam bentuk text yang dapat dibaca oleh MS Project, seperti pada gambar 4.24.



Gambar 4.22 Diagram Batang Effort Setiap Tahap Software AFE-BIRDS



Gambar 4.23 Gantt Chart Software AFE-BIRDS

Task Name	Duration	Start	Finish	Pred	3rd Quarter	4th Quarter
Plans and Requirement	50 days	Fri 5/20/05	Thu 7/28/05			
Product Design	66 days	Fri 7/29/05	Fri 10/28/05	1		
Programming	128 days	Mon 10/31/05	Wed 4/26/06	2		
Integration and Test	64 days	Thu 4/27/06	Tue 7/25/06	3		

Gambar 4.24 Hasil Konversi Software AFE-BIRDS ke MS Project

4.4 Estimasi software FAM

Tabel 4.4 Scale Factor dan Cost Driver Software FAM

Form SFD-5a COCOMO II Project Actuals: Simple Completed Project

1. Project Title: FAM 2. Project ID No. 03 3. Rev No.
4. Date Prepared: 21 May 05 Originator: Lutfi Shahab 6. Organization: PT CPT
7. Starting Milestone: End of Req Def 8. Ending Milestone: End of User Acceptance
9. Total no. of person-months: 20 PM 10. Total no. of calendar months: 10 Months
11. Equivalent SLOC: 78 K 12. Total no. of SLOC reused:
13. Non-trivial defects detected: 14. Defect detection starting milestone:

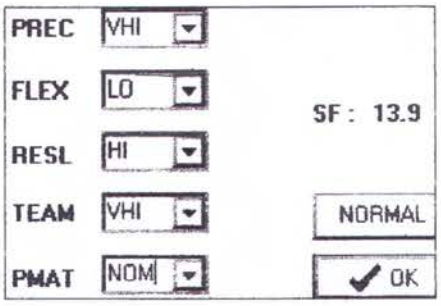
15. Project attribute ratings

	VL	L	N	H	VH	XH	Comments (Including Don't Kn
Precedentedness (PREC)					●		
Development flexibility (FLEX)		●					
Architecture/risk resolution (RESL)				●			
Team cohesion (TEAM)					●		
Process maturity (PMAT)			●				
Required reliability (RELY)				●			
Data base size (DATA)			●				
Product complexity (CPLX)		●					
Develop for reuse (RUSE)			●				
Documentation match to life-cycle needs (DOCU)			●				
Execution time constraint (TIME)			●				
Main storage constraint (STOR)			●				
Platform volatility (PVOL)		●					
Analyst capability (ACAP)				●			
Programmer capability (PCAP)				●			
Personnel continuity (PCON)					●		
Applications experience (APEX)					●		
Platform experience (PLEX)				●			
Language & tool experience (LTEX)				●			
Use of software tools (TOOL)	●						
Multi-site development (SITE)					●		
Required development schedule (SCED)			●				
Other							

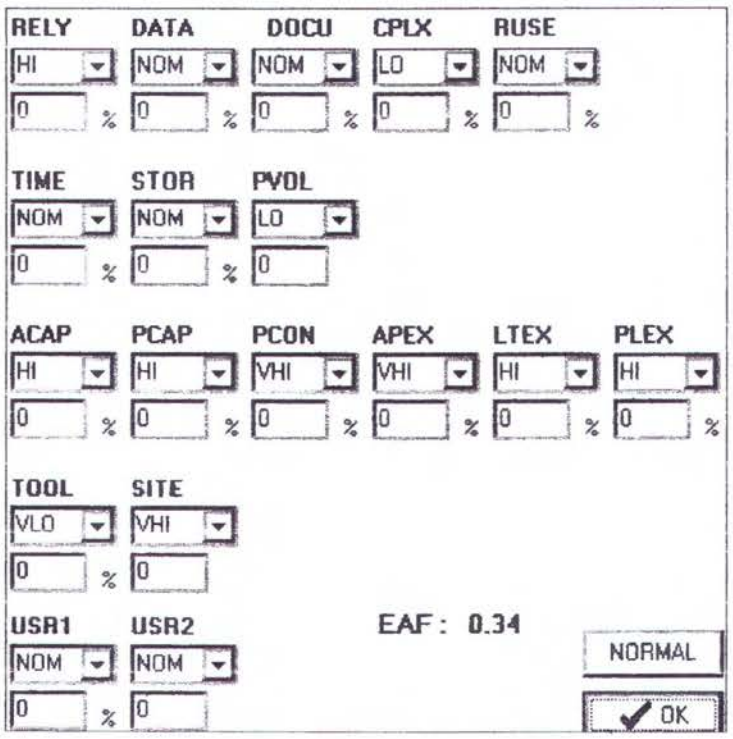
16. Special project characteristics or lessons learned:

Dari hasil kuisioner sistem FAM didapatkan PM aktual, TDEV aktual, Scale Factor dan cost driver seperti pada tabel 4.4. Scale Factor (SF) yang didapat kemudian dimasukkan ke software estimasi seperti pada gambar 4.25, sedangkan

cost drivernya (EAF) seperti pada gambar 4.26. Dari sini didapatkan nilai SF : **13.9**, sedangkan nilai EAF adalah : **0.34.**



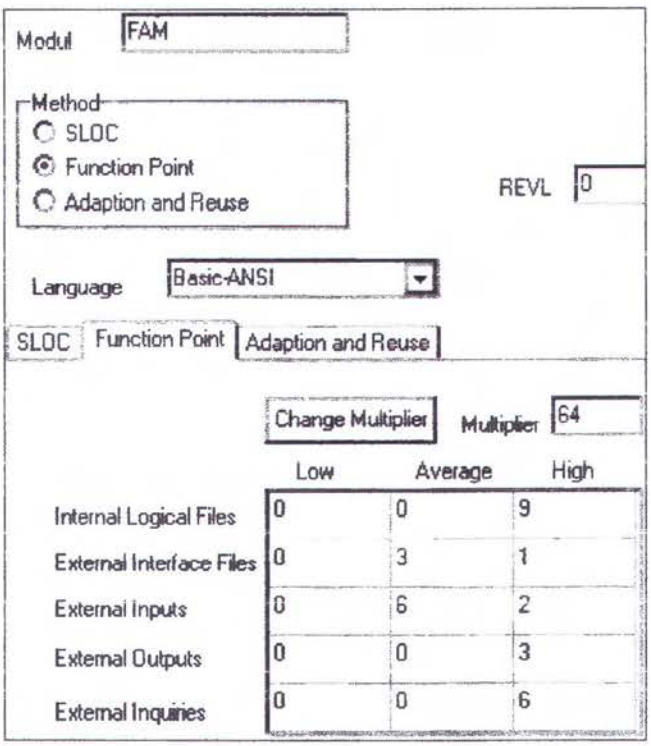
Gambar 4.25 Scale Factor Software FAM



Gambar 4.26 Cost Driver Software FAM

Langkah selanjutkan dimasukkan data size dengan metode Function Point seperti pada gambar 4.27. Nilai Function Point didapat dari analisa di bab sebelumnya. Sistem ini ditulis dengan bahasa CFM (*ColdFusion*) yang secara sintaks mirip

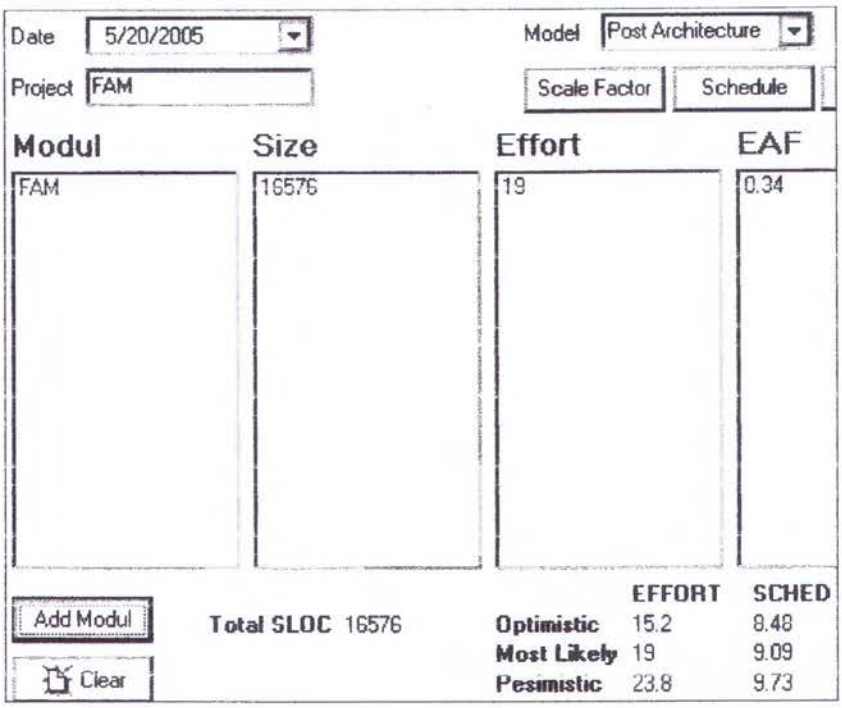
dengan **Basic-ANSI**, dimana Basic-ANSI memiliki nilai pengali *(multiplier)* **64**. Multiplier ini dikalikan dengan semua nilai Function Point.



Gambar 4.27 UFP Software FAM

Dari pemasukan function point ini dapat dihitung SLOC sistem, yaitu **16576**, seperti pada gambar 4.28. Setelah diketahui SLOC maka dapat dihitung nilai effort : **19** PM dan schedule : **9.09** bulan. Dihitung juga effort dan schedule dengan persentase *optimistic* (80%) dan *pesimistic* (125%).

Effort dan schedule dapat dirinci menjadi tahapan-tahapan *life-cycle waterfall*, dimana masing-masing tahapan mempunyai persentase, seperti pada gambar 4.29. Dari masing-masing tahapan dihitung menurut waktu kalender dalam satuan hari dan jumlah staf (dibulatkan) yang dibutuhkan.



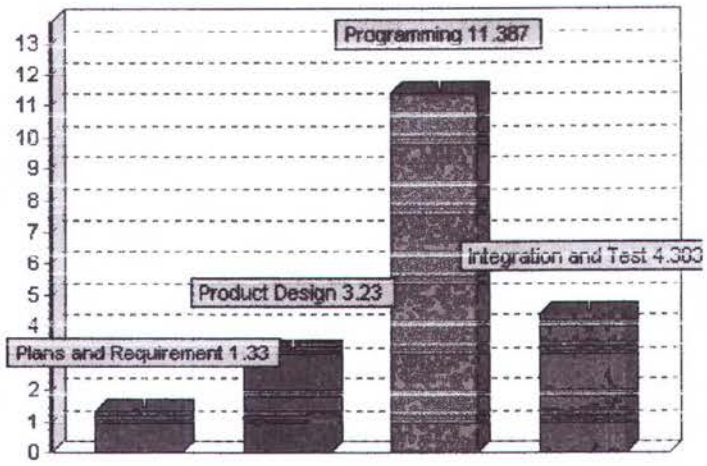
Gambar 4.28 Effort dan Jadwal Software FAM

	% Effort	Effort	% Sced	Sced	Staff
Plans And Requirement	7	1.33	18.71	1.701	1
Product Design	17	3.23	25.36	2.305	2
Programming	59.93	11.39	50.57	4.597	3
Integration and Test	23.07	4.383	24.07	2.188	3

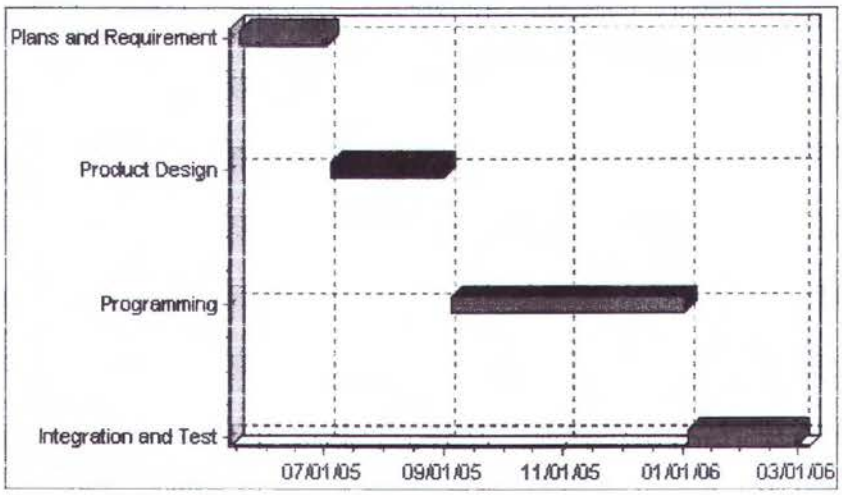
	Duration	Start	Finish
Plans And Requirement	32	Fri 5/20/2005	Mon 7/4/2005
Product Design	44	Tue 7/5/2005	Fri 9/2/2005
Programming	87	Mon 9/5/2005	Tue 1/3/2006
Integration and Test	42	Wed 1/4/2006	Thu 3/2/2006

Gambar 4.29 Jadwal Tiap Tahap Software FAM

Dibuat juga diagram batang dari effort untuk masing-masing tahap seperti pada gambar 4.30 dan gantt chart dari jadwal seperti pada gambar 4.31. Terakhir apabila dibutuhkan data dapat dikonversi dalam bentuk text yang dapat dibaca oleh MS Project, seperti pada gambar 4.32.



Gambar 4.30 Diagram Batang Effort Setiap Tahap



Gambar 4.31 Gantt Chart Software FAM

Task Name	Duration	Start	Finish	Pred
Plans and Requirement	32 days	Fri 5/20/05	Mon 7/4/05	
Product Design	44 days	Tue 7/5/05	Fri 9/2/05	1
Programming	87 days	Mon 9/5/05	Tue 1/3/06	2
Integration and Test	42 days	Wed 1/4/06	Thu 3/2/06	3

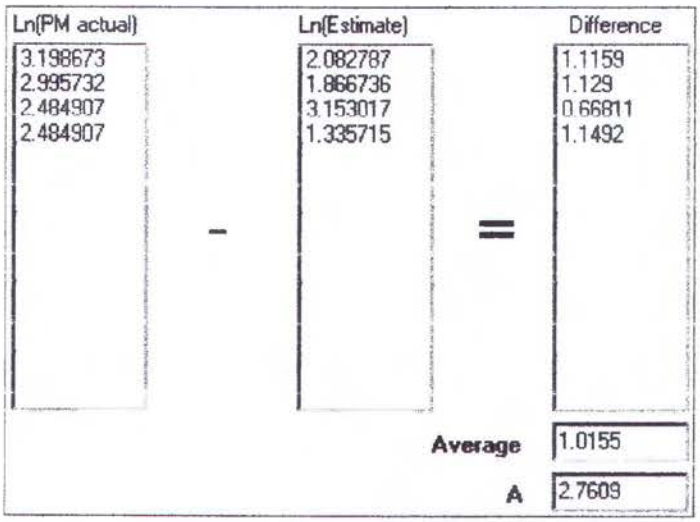
Gambar 4.32 Hasil Konversi Software FAM ke MS Project

4.5 Kalibrasi Lokal A

Setelah diperoleh data SLOC, EM, dan SF keempat software tersebut, maka nilai data ini dimasukkan ke analisa Kalibrasi, dimasukkan juga nilai PM aktual, seperti pada gambar 4.33. Kemudian dihitung nilai Log natural (Ln) untuk PM aktual dan PM estimasi dari COCOMO II. Kemudian dicari selisihnya, dan dirata-rata. Dengan anti Log didapatkan nilai konstanta A lokal yaitu : **2.7609**, seperti pada gambar 4.34.

	Nama	PM	SLOC	EM	SF
	ABOM	24.5	25.216	0.249	16.61
	FAM	20	16.576	0.34	13.9
	AFE-BIRDS	12	23.168	0.833	15.14
▶	PLF	12	12.544	0.249	16.78

Gambar 4.33 Pemasukan Data untuk Kalibrasi



Gambar 4.34 Penghitungan Konstanta A di Lingkungan Lokal

Kemudian kalibrasi yang telah dilakukan dianalisa, dibandingkan persentase errornya bila menggunakan konstanta A dari COCOMO dan konstanta A hasil dari kalibrasi lokal, seperti pada gambar 4.35. Terlihat disini terjadi penurunan error dari **122.23%** (COCOMO II) menjadi **117.82%** (lokal). Tapi persentase ini masih terlalu

tinggi, lebih dari persentase penghitungan effort optimistic (-20%) atau persentase effort pesimistic (+25%). Hal ini disebabkan adanya error yang tinggi pada penghitungan effort AFE-BIRDS yaitu 473,46% (COCOMO).

PM actual	PM COCOMO	PM local A	Error COCOMO	Error Local A	% Error COCOMO	% Error Local A
24.5	23.59883	22.161	0.90117	2.339	3.6782	9.5469
20	19.01343	17.855	0.98657	2.145	4.9329	10.725
12	68.8153	64.622	56.815	52.622	473.46	438.52
12	11.17998	10.499	0.82002	1.501	6.8335	12.508
				Average	122.23	117.82

Gambar 4.35 Analisa Kalibrasi

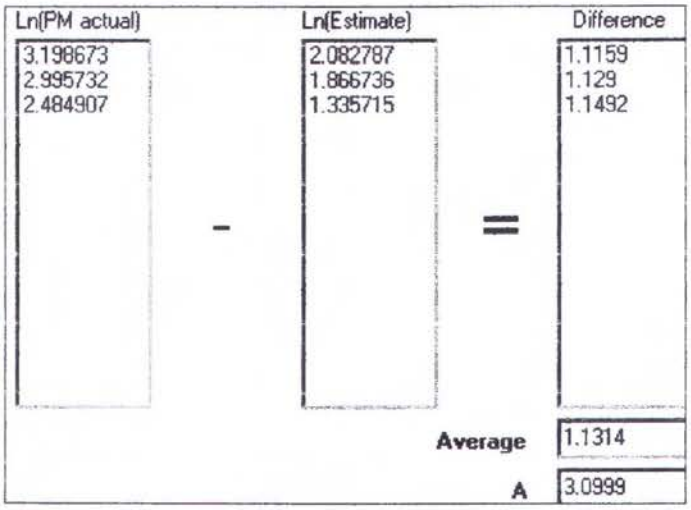
Karena AFE-BIRDS mempunyai error yang terlalu tinggi, penghitungan diulang hanya dengan tiga software saja, tanpa AFE-BIRDS seperti pada gambar 4.36. Kemudian dihitung nilai Log natural (Ln) untuk PM aktual dan PM estimasi dari COCOMO II. Kemudian dicari selisihnya, dan dirata-rata. Dengan anti Log didapatkan nilai konstanta A lokal yaitu : **3.0999**, seperti pada gambar 4.37.

	Nama	PM	SLOC	EM	SF
	ABOM	24.5	25.216	0.249	16.61
	FAM	20	16.576	0.34	13.9
▶	PLF	12	12.544	0.249	16.78

Gambar 4.36 Pemasukan Data untuk Kalibrasi tanpa AFE-BIRDS

Kemudian kalibrasi yang telah dilakukan dianalisa, dibandingkan persentase errornya bila menggunakan konstanta A dari COCOMO II dan konstanta A hasil dari kalibrasi lokal, seperti pada gambar 4.38. Terlihat disini terjadi penurunan persentase error dari **5.1482%** (COCOMO) menjadi **1.187%** (lokal). Persentase error dari

perhitungan ini lebih kecil bila dibandingkan dengan persentase penghitungan effort optimistic (-20%) atau persentase effort pesimistic (+25%), sehingga dapat dikatakan bahwa metode COCOMO II dan kalibrasinya ini menghasilkan estimasi yang cukup akurat.



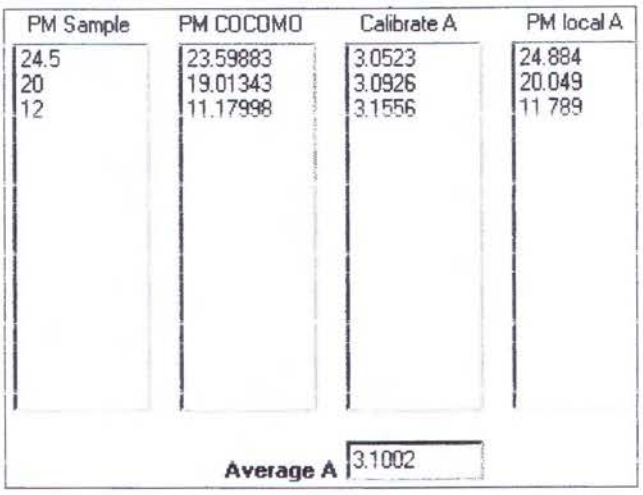
Gambar 4.37 Penghitungan Konstanta A di Lingkungan Lokal tanpa AFE-BIRDS

PM actual	PM COCOMO	PM local A	Error COCOMO	Error Local A	% Error COCOMO	% Error Local A
24.5	23.59883	24.882	0.90117	0.382	3.6782	1.5592
20	19.01343	20.047	0.98657	0.047	4.9329	0.235
12	11.17998	11.788	0.82002	0.212	6.8335	1.7667
				Average	5.1482	1.187

Gambar 4.38 Analisa Kalibrasi tanpa AFE-BIRDS

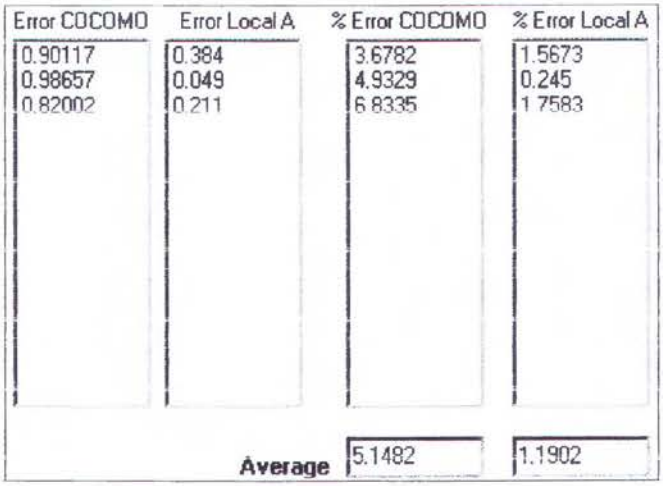
Kalibrasi dapat dihitung dengan metode lain yaitu Bayesian seperti pada gambar 4.39. Metode ini dilakukan dengan cara membandingkan antara PM aktual

dan PM estimasi dari COCOMO II. Kemudian dicari nilai lokal A, dan dirata-rata, sehingga ditemukan nilai A yaitu : **3.1002**.



PM Sample	PM COCOMO	Calibrate A	PM local A
24.5	23.59883	3.0523	24.884
20	19.01343	3.0926	20.049
12	11.17998	3.1556	11.789
		Average A	3.1002

Gambar 4.39 Penghitungan Konstanta A di Lingkungan Lokal Metode Bayesian



Error COCOMO	Error Local A	% Error COCOMO	% Error Local A
0.90117	0.384	3.6782	1.5673
0.98657	0.049	4.9329	0.245
0.82002	0.211	6.8335	1.7583
	Average	5.1482	1.1902

Gambar 4.40 Analisa Kalibrasi Metode Bayesian

Kemudian kalibrasi yang telah dilakukan dianalisa, dibandingkan persentase errornya bila menggunakan konstanta A dari COCOMO II dan konstanta A hasil dari kalibrasi lokal, seperti pada gambar 4.40. Terlihat disini persentase error dari perhitungan metode Bayesian lebih besar sedikit dari metode Log Natural, yaitu 1.1902% tapi tetap lebih kecil bila dibandingkan dengan persentase penghitungan

effort optimistic (-20%) atau persentase effort pesimistic (+25%), sehingga dapat dikatakan bahwa kalibrasi dengan metode Bayesian ini menghasilkan estimasi yang cukup akurat.

4.6 Kalibrasi Lokal C

Untuk menghitung estimasi TDEV dengan lebih akurat, konstanta C dapat dikalibrasi. Kalibrasi dilakukan dengan cara memasukkan data TDEV aktual, PM (*effort*) dan SF (*Scale Factor*), seperti pada gambar 4.41. Kemudian dihitung nilai Log natural (Ln) untuk TDEV aktual dan TDEV estimasi dari COCOMO II. Kemudian dicari selisihnya, dan dirata-rata. Dengan anti Log didapatkan nilai konstanta C lokal yaitu : **2.8892** seperti pada gambar 4.42.

	Name	TDEV	PM	SF
	CALAM	10	20.047	13.9
	PLF	6	11.788	16.78
▶	ABOM	6	24.882	16.61

Gambar 4.41 Pemasukan Data untuk Kalibrasi Lokal C

Ln(Actual)		Ln(Estimate)		Difference
2.302585		0.9228089		1.3798
1.791759		0.7736769		1.0181
1.791759		1.00667		0.78509
	-		=	
			Average	1.061
			C	2.8892

Gambar 4.42 Penghitungan Konstanta C di Lingkungan Lokal

Kemudian kalibrasi yang telah dilakukan dianalisa, dibandingkan persentase errornya bila menggunakan konstanta C dari COCOMO II dan konstanta C hasil dari

kalibrasi lokal, seperti pada gambar 4.43. Terlihat disini terjadi penurunan persentase error dari **35.874%** (COCOMO) menjadi **21.15%** (lokal).

Actual	COCOMO	Local	Error COCOMO	Error Local	% Error COCOMO	% Error Local
10	9.234999	7.2703	0.765	2.7297	7.65	27.297
6	7.95554	6.263	1.9555	0.263	32.592	4.3833
6	10.04286	7.9062	4.0429	1.9062	67.381	31.77
				Average	35.874	21.15

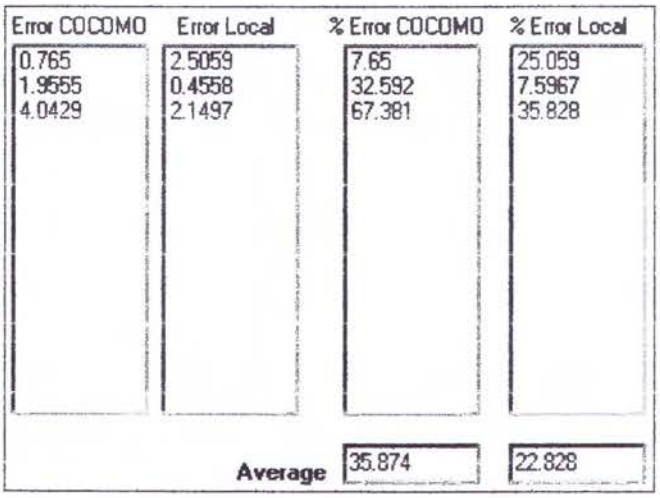
Gambar 4.43 Analisa Kalibrasi Lokal C

Kalibrasi dapat dihitung dengan metode lain yaitu Bayesian seperti pada gambar 4.44. Metode ini dilakukan dengan cara membandingkan antara TDEV aktual dan TDEV estimasi dari COCOMO II. Kemudian dicari nilai lokal C, dan dirata-rata, sehingga ditemukan nilai C yaitu : **2.9782**.

TDEV Sample	TDEV COCOMO	Calibrate	TDEV local
10	9.234999	3.974	7.4941
6	7.95554	2.7679	6.4558
6	10.04286	2.1926	8.1497
		Average	2.9782

Gambar 4.44 Penghitungan Konstanta C di Lingkungan Lokal dengan Metode Bayesian

Kemudian kalibrasi yang telah dilakukan dianalisa, dibandingkan persentase errornya bila menggunakan konstanta C dari COCOMO II dan konstanta C hasil dari kalibrasi lokal, seperti pada gambar 4.45. Terlihat disini terjadi penurunan persentase error dari **34.758%** (COCOMO) menjadi **22.835%** (lokal).



Gambar 4.45 Analisa Kalibrasi Lokal C dengan Metode Bayesian

4.7 Penjadwalan Multi Proyek

Input data dari software ini berasal dari estimasi effort dan schedule yang telah dilakukan sebelumnya. Masing-masing tahap dari proyek memiliki staf yang dibedakan keahliannya, yaitu :

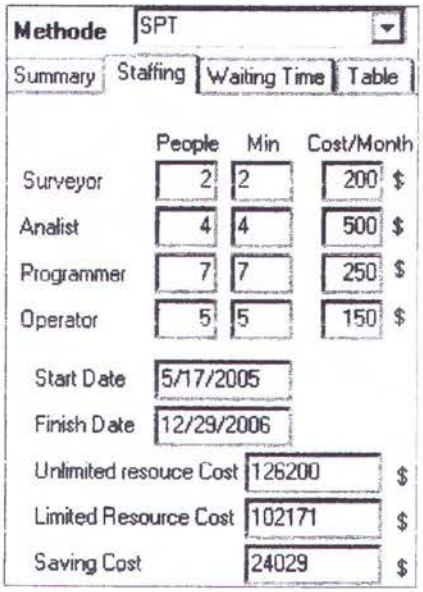
- Surveyor, pada tahap *Plans and Requirement*
- Analist, pada tahap *Product Design*
- Programmer, pada tahap *Programming*
- Operator, pada tahap *Integration and Test*

Sebelum dibuat jadwal multi proyek secara heuristik, staf ini diinisial jumlah dan biaya per bulannya seperti pada gambar 4.46.Untuk mendapatkan hasil maksimal

diinisial jumlah staf sama dengan jumlah minimal yang dibutuhkan yaitu jumlah yang dibutuhkan dari proyek yang terbesar.

Metode penjadwalan yang dapat digunakan pada software ini adalah :

- **SPT** : proyek yang lebih pendek durasinya didahulukan
- **Due Date** : proyek yang jatuh temponya lebih dekat didahulukan
- **NPV** : proyek yang harganya lebih tinggi didahulukan
- **Mixed** : dicoba SPT, Due Date, dan NPV lalu dicari yang biayanya terendah
- **Permutation**: dicoba semua kemungkinan urutan prioritas dengan permutasi, kemudian dicari yang terendah biayanya



Gambar 4.46 Project Staffing Metode SPT

Gambar 4.47 Project Staffing Metode Due Date

Gambar 4.48 Project Staffing Metode NPV

Gambar 4.49 Project Staffing Metode Mixed

Gambar 4.50 Project Staffing Metode Permutation

Start date menunjukkan waktu awal dari proyek yang pertama dimulai, sedangkan *finish date* adalah waktu akhir dari proyek yang terakhir. *Unlimited Cost* adalah perhitungan biaya apabila dianggap jumlah staf tidak terbatas, sedangkan *Limited Cost* adalah perhitungan biaya dengan staf sesuai dengan yang diinputkan

(terbatas). *Saving Cost* adalah selisih dari kedua biaya tersebut, yaitu biaya yang dapat dihemat bila menggunakan jumlah staf yang terbatas, meskipun nantinya terjadi keterlambatan. Dari sini dapat dilihat bahwa Limited Cost dengan pendekatan heuristik lebih baik karena Unlimited Cost selalu lebih besar dari Limited Cost.

Terlihat pada gambar gambar 4.46 sampai gambar 4.50 metode yang terbaik adalah Permutation, yang menghasilkan *Saving Cost* tertinggi. Ini dikarenakan semua kemungkinan dihitung, lalu dipilih yang terbaik. Tetapi metode ini memiliki kekurangan yaitu maksimum proyek hanya 6 saja. Ini dikarenakan banyaknya perhitungannya adalah 2x3x4x5x6 atau 720 kali perhitungan. Program Delphi 5.0 yang digunakan tidak mampu menghitung lebih dari itu (2x3x4x5x6x7=5040). Pada kasus lain dapat juga metode SPT, NPV, atau Due Date yang menjadi yang terbaik.

People adalah jumlah sumber daya yang tersedia. Sedangkan *Min* adalah jumlah minimum dari sumber daya. Min diambil nilainya dari kebutuhan staf pada proyek yang terbesar. Jumlah People tidak boleh kurang dari jumlah Min, karena akan terdapat suatu proyek yang tidak terpenuhi kebutuhannya. People boleh lebih besar dari Min, tetapi hal ini dapat menyebabkan tambahan biaya pada total biaya (Limited Cost).

Apabila terjadi keterlambatan akan dihitung sebagai kerugian. Penghitungan ini terdapat pada kolom *Loosing* seperti pada gambar 4.51. *Loosing* diperoleh dari perkalian waktu tunggu (W1, W2, W3, W4) dari proyek dengan staf pada setiap tahap. W1 adalah waktu tunggu pada tahap *Plans and Requirement*, W2 adalah waktu tunggu pada tahap *Product Design*, W3 adalah waktu tunggu pada tahap *Programming*, W4 adalah waktu tunggu pada tahap *Integration and Test*. Terlihat di sini proyek diurutkan dengan prioritas terbesar berada di paling atas. Prioritas

terbesar akan didahulukan dalam pengambilan sumber daya. Urutan ini dibuat berdasarkan metode yang digunakan (SPT, NPV, Permutation). Jumlah total dari Loosing yang terkecil merupakan metode yang menguntungkan.

Methode: SPT

Summary | Staffing | Waiting Time | Table

Project	Waiting Time W1	W2	W3	W4	Loosing
PLF	0	0	0	0	0
FAM	0	0	0	0	0
ABOM	26	4	25	0	1320
AFE-BIRDS	60	2	51	0	7751

Gambar 4.51 Project Waiting Time Metode SPT

Methode: NPV

Summary | Staffing | Waiting Time | Table

Project	Waiting Time W1	W2	W3	W4	Loosing
AFE-BIRDS	0	0	0	0	0
ABOM	50	32	80	0	3888
FAM	50	34	84	47	4644
PLF	82	52	133	0	3204

Gambar 4.52 Project Waiting Time Metode NPV

Methode: PERMUTATION

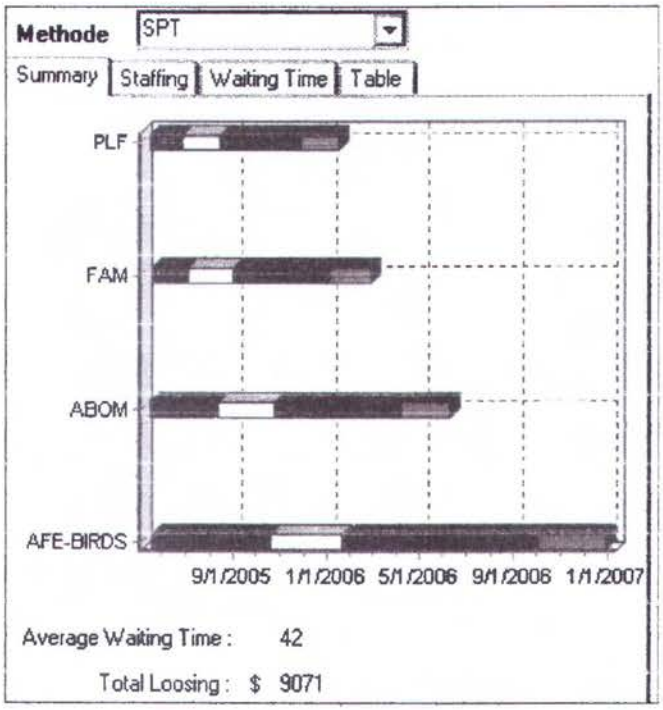
Summary | Staffing | Waiting Time | Table

Project	Waiting Time W1	W2	W3	W4	Loosing
FAM	0	0	0	0	0
ABOM	0	0	0	31	744
AFE-BIRDS	34	0	24	0	3978
PLF	82	42	114	0	2856

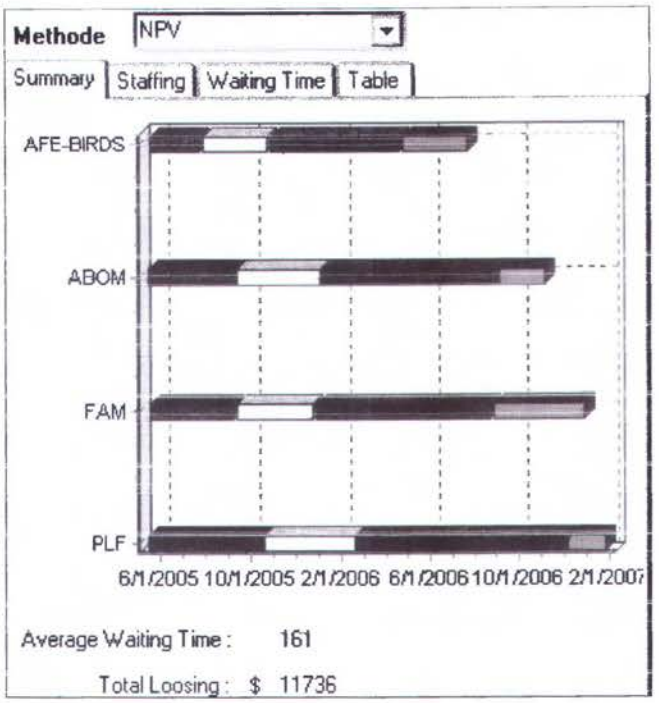
Gambar 4.53 Project Waiting Time Metode Permutation

Hasil akhir dari penjadwalan ini diringkas dalam *Project Summary* seperti pada gambar 4.54 sampai gambar 4.56. *Average Waiting Time* adalah rata-rata waktu

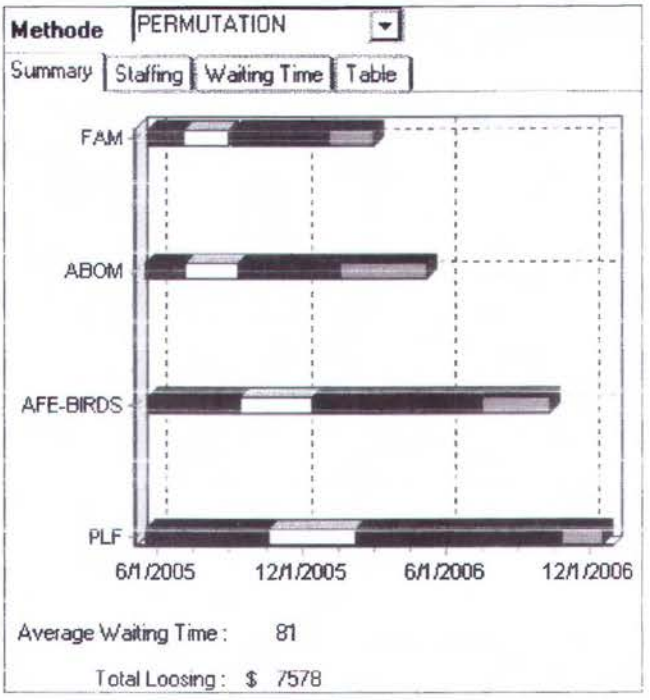
tunggu dari semua proyek. Dan *Total Loosing* adalah jumlah total dari kerugian yang disebabkan keterlambatan. Terlihat di sini proyek sudah diurutkan menurut prioritas yang didahulukan penghitungannya menurut metode yang digunakan. Metode yang baik akan memiliki jumlah kerugian terkecil. Meskipun dengan metode SPT dan Due Date memiliki waktu tunggu rata-rata yang terkecil tapi jumlah kerugiannya bukan yang terkecil. Jadi metode Permutation adalah yang terbaik dengan jumlah kerugian yang terkecil.



Gambar 4.54 Project Summary Metode SPT

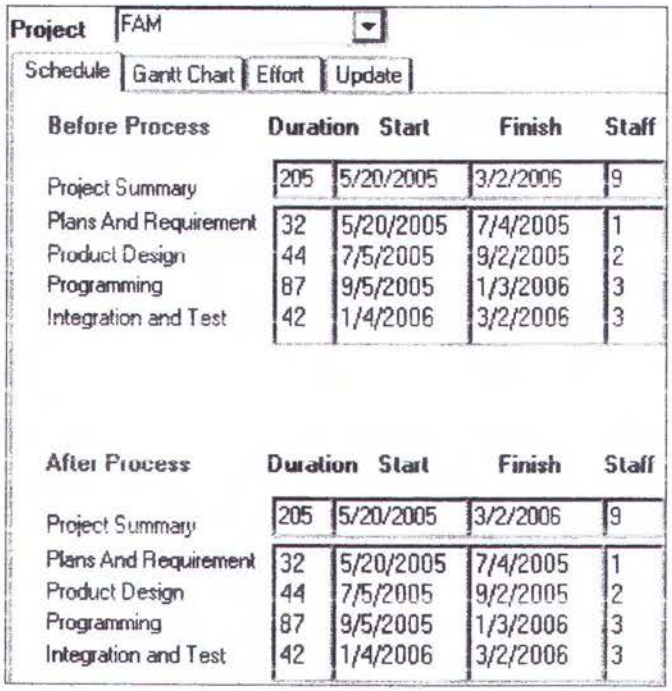


Gambar 4.55 Project Summary Metode NPV



Gambar 4.56 Project Summary Metode Permutation

Pada *Project Schedule Detail* dapat dilihat jadwal proyek satu persatu (dengan metode Permutation) seperti pada gambar 4.57 adalah detail dari proyek FAM, gambar 4.58 adalah detail dari proyek AFE-BIRDS, gambar 4.59 adalah detail dari proyek PLF, gambar 4.60 adalah detail dari proyek ABOM. *Before Process* menunjukkan jadwal dengan staf tak terbatas, sedangkan *After Process* adalah jadwal dengan staf yang terbatas, sehingga terkadang terjadi keterlambatan. Hal ini terlihat pada gambar 4.60 *Duration* After Process lebih besar dari Before Process. Ini menunjukkan adanya keterlambatan sebanyak 254 – 223 = 31 hari. Keterlambatan ini sesuai dengan waktu tunggu yang ditunjukkan pada gambar 4.53, yaitu pada proyek ABOM, kolom W4.



Project: FAM

Schedule | Gantt Chart | Effort | Update

Before Process	Duration	Start	Finish	Staff
Project Summary	205	5/20/2005	3/2/2006	9
Plans And Requirement	32	5/20/2005	7/4/2005	1
Product Design	44	7/5/2005	9/2/2005	2
Programming	87	9/5/2005	1/3/2006	3
Integration and Test	42	1/4/2006	3/2/2006	3

After Process	Duration	Start	Finish	Staff
Project Summary	205	5/20/2005	3/2/2006	9
Plans And Requirement	32	5/20/2005	7/4/2005	1
Product Design	44	7/5/2005	9/2/2005	2
Programming	87	9/5/2005	1/3/2006	3
Integration and Test	42	1/4/2006	3/2/2006	3

Gambar 4.57 Project Schedule Detail FAM

Project: PLF

Schedule | Gantt Chart | Effort | Update

Before Process	Duration	Start	Finish	Staff
Project Summary	177	5/17/2005	1/18/2006	6
Plans And Requirement	29	5/17/2005	6/24/2005	1
Product Design	38	6/27/2005	8/17/2005	1
Programming	73	8/18/2005	11/28/2005	2
Integration and Test	37	11/29/2005	1/18/2006	2

After Process	Duration	Start	Finish	Staff
Project Summary	415	5/17/2005	12/18/2006	6
Plans And Requirement	111	5/17/2005	10/18/2005	1
Product Design	80	10/19/2005	2/7/2006	1
Programming	187	2/8/2006	10/26/2006	2
Integration and Test	37	10/27/2006	12/18/2006	2

Gambar 4.58 Project Schedule Detail PLF

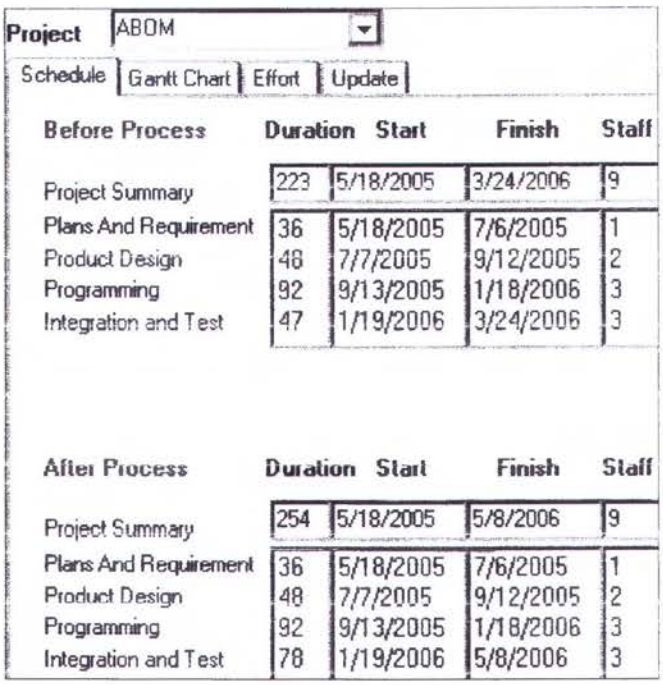
Project: AFE-BIRDS

Schedule | Gantt Chart | Effort | Update

Before Process	Duration	Start	Finish	Staff
Project Summary	308	5/20/2005	7/25/2006	18
Plans And Requirement	50	5/20/2005	7/28/2005	2
Product Design	66	7/29/2005	10/28/2005	4
Programming	128	10/31/2005	4/26/2006	7
Integration and Test	64	4/27/2006	7/25/2006	5

After Process	Duration	Start	Finish	Staff
Project Summary	366	5/20/2005	10/13/2006	18
Plans And Requirement	84	5/20/2005	9/14/2005	2
Product Design	66	9/15/2005	12/15/2005	4
Programming	152	12/16/2005	7/17/2006	7
Integration and Test	64	7/18/2006	10/13/2006	5

Gambar 4.59 Project Schedule Detail AFE-BIRDS


Project ABOM

Schedule | Gantt Chart | Effort | Update

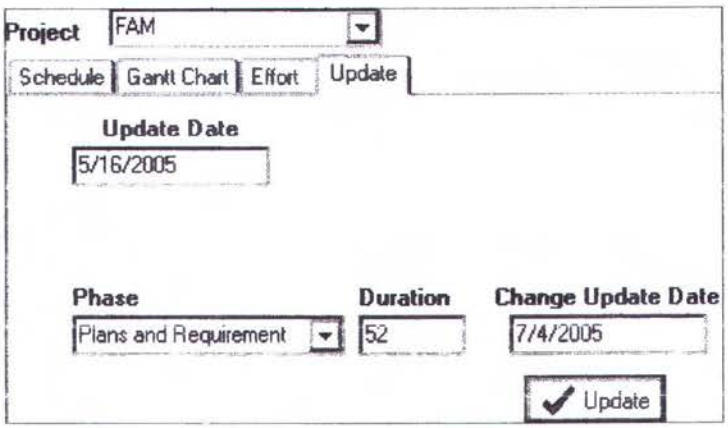
Before Process	Duration	Start	Finish	Staff
Project Summary	223	5/18/2005	3/24/2006	9
Plans And Requirement	36	5/18/2005	7/6/2005	1
Product Design	48	7/7/2005	9/12/2005	2
Programming	92	9/13/2005	1/18/2006	3
Integration and Test	47	1/19/2006	3/24/2006	3

After Process	Duration	Start	Finish	Staff
Project Summary	254	5/18/2005	5/8/2006	9
Plans And Requirement	36	5/18/2005	7/6/2005	1
Product Design	48	7/7/2005	9/12/2005	2
Programming	92	9/13/2005	1/18/2006	3
Integration and Test	78	1/19/2006	5/8/2006	3

Gambar 4.60 Project Schedule Detail ABOM

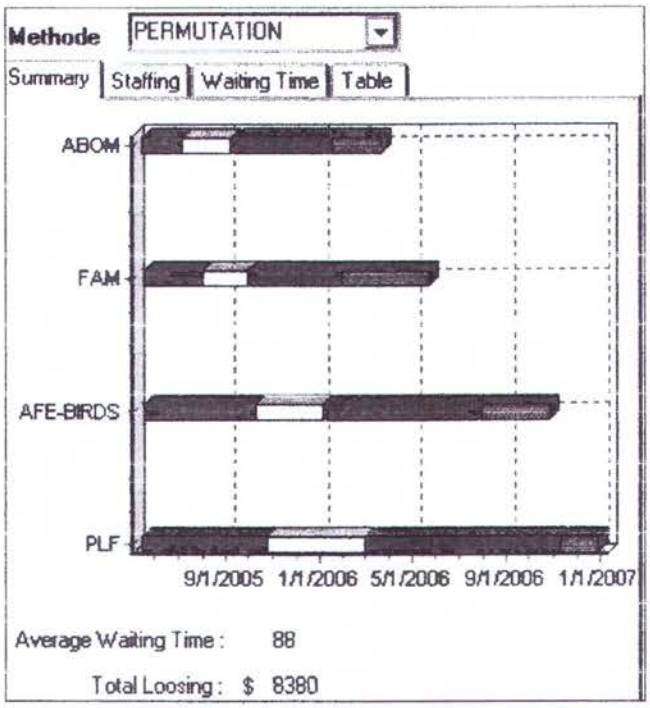
4.8 Update Durasi

Pada waktu pelaksanaan jadwal multi proyek ini dapat dimungkinkan adanya kemunduran atau keterlambatan diantara proyek-proyek yang sedang berlangsung. Ini dapat menyebabkan kemunduran pada proyek lain yang sama sumber dayanya. Untuk itu harus dilakukan penjadwalan ulang untuk tahap-tahap proyek yang belum dilaksanakan. Pada gambar 4.61 ditampilkan pada proyek FAM pada tahap Plans and Requirement, mengalami keterlambatan sehingga yang semula berdurasi 44 hari menjadi 52 hari.



Gambar 4.61 Update Durasi karena Keterlambatan pada Proyek FAM

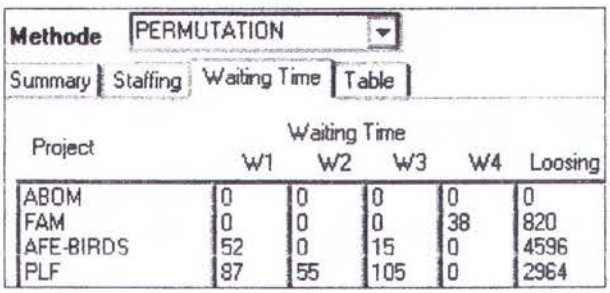
Setelah dilakukan perhitungan kembali, pada gambar 4.62 ditampilkan project sumary dengan metode permutation didapatkan urutan proyek yang berbeda. Hal ini dikarenakan dihitung kembali semua biaya proyek dan dicari yang termurah biayanya. Pada gambar 4.62 ditampilkan perhitungan biaya total dari semua proyek. Dikarenakan adanya keterlambatan, maka biaya total juga bertambah dari $108.278 menjadi $110.155. Sedangkan biaya yang dapat dihemat juga berkurang dari $32.422 menjadi $30.554.



Gambar 4.62 Project Summary setelah Update

Gambar 4.63 Perhitungan Cost setelah Update

Pada gambar 4.64 ditampilkan waktu tunggu untuk masing-masing tahap proyek setelah dilakukan update. Karena Proyek FAM berada di urutan terbawah, maka memiliki prioritas terendah, sehingga pada tahap 3 (Programming) mempunyai waktu tunggu yang besar.



Gambar 4.64 Waiting Time setelah Update

Pada gambar 4.65 ditampilkan proyek PLF secara detail. Pada gambar 4.66 ditampilkan proyek ABOM secara detail. Pada gambar 4.67 ditampilkan proyek AFE-BIRDS secara detail. Pada gambar 4.68 ditampilkan proyek FAM secara detail.

Terlihat pada gambar, durasi proyek PLF (*Integration and Test*) menjadi lebih pendek. Ini dikarenakan setelah update proyek PLF mendapat prioritas lebih tinggi dari sebelumnya, sehingga dapat mengambil sumber daya terlebih dahulu. Hal ini terjadi juga pada proyek ABOM, yang durasinya dari 446 hari menjadi 275 hari. Sedang pada durasi proyek FAM berubah secara signifikan menjadi lebih panjang dari sebelumnya. Ini disebabkan prioritas proyek FAM berubah dari yang paling tinggi menjadi yang paling rendah.

Project PLF

Schedule | Gantt Chart | Effort | Update

Before Process	Duration	Start	Finish	Staff
Project Summary	415	5/17/2005	12/18/2006	6
Plans And Requirement	111	5/17/2005	10/18/2005	1
Product Design	80	10/19/2005	2/7/2006	1
Programming	187	2/8/2006	10/26/2006	2
Integration and Test	37	10/27/2006	12/18/2006	2

After Process	Duration	Start	Finish	Staff
Project Summary	424	5/17/2005	12/29/2006	6
Plans And Requirement	116	5/17/2005	10/25/2005	1
Product Design	93	10/26/2005	3/3/2006	1
Programming	178	3/6/2006	11/8/2006	2
Integration and Test	37	11/9/2006	12/29/2006	2

Gambar 4.65 Project Detail PLF setelah Update

Project ABOM

Schedule | Gantt Chart | Effort | Update

Before Process	Duration	Start	Finish	Staff
Project Summary	254	5/18/2005	5/8/2006	9
Plans And Requirement	36	5/18/2005	7/6/2005	1
Product Design	48	7/7/2005	9/12/2005	2
Programming	92	9/13/2005	1/18/2006	3
Integration and Test	78	1/19/2006	5/8/2006	3

After Process	Duration	Start	Finish	Staff
Project Summary	223	5/18/2005	3/24/2006	9
Plans And Requirement	36	5/18/2005	7/6/2005	1
Product Design	48	7/7/2005	9/12/2005	2
Programming	92	9/13/2005	1/18/2006	3
Integration and Test	47	1/19/2006	3/24/2006	3

Gambar 4.66 Project Detail ABOM setelah Update

Project: AFE-BIRDS

Schedule | Gantt Chart | Effort | Update

Before Process	Duration	Start	Finish	Staff
Project Summary	366	5/20/2005	10/13/2006	18
Plans And Requirement	84	5/20/2005	9/14/2005	2
Product Design	66	9/15/2005	12/15/2005	4
Programming	152	12/16/2005	7/17/2006	7
Integration and Test	64	7/18/2006	10/13/2006	5

After Process	Duration	Start	Finish	Staff
Project Summary	375	5/20/2005	10/26/2006	18
Plans And Requirement	102	5/20/2005	10/10/2005	2
Product Design	66	10/11/2005	1/10/2006	4
Programming	143	1/11/2006	7/28/2006	7
Integration and Test	64	7/31/2006	10/26/2006	5

Gambar 4.67 Project Detail AFE-BIRDS setelah Update

Project: FAM

Schedule | Gantt Chart | Effort | Update

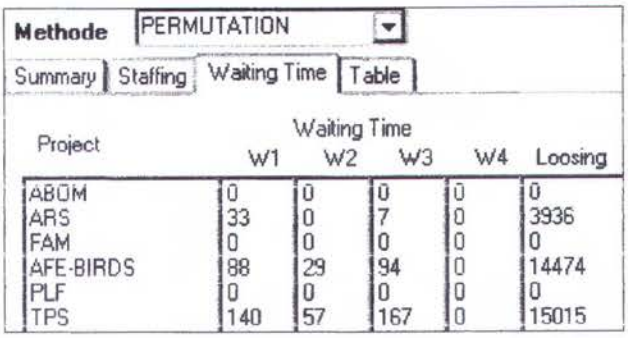
Before Process	Duration	Start	Finish	Staff
Project Summary	225	5/20/2005	3/2/2006	9
Plans And Requirement	52	5/20/2005	8/1/2005	1
Product Design	44	8/2/2005	9/30/2005	2
Programming	87	10/3/2005	1/31/2006	3
Integration and Test	42	2/1/2006	3/30/2006	3

After Process	Duration	Start	Finish	Staff
Project Summary	263	5/20/2005	5/23/2006	9
Plans And Requirement	52	5/20/2005	8/1/2005	1
Product Design	44	8/2/2005	9/30/2005	2
Programming	87	10/3/2005	1/31/2006	3
Integration and Test	80	2/1/2006	5/23/2006	3

Gambar 4.68 Project Detail FAM setelah Update

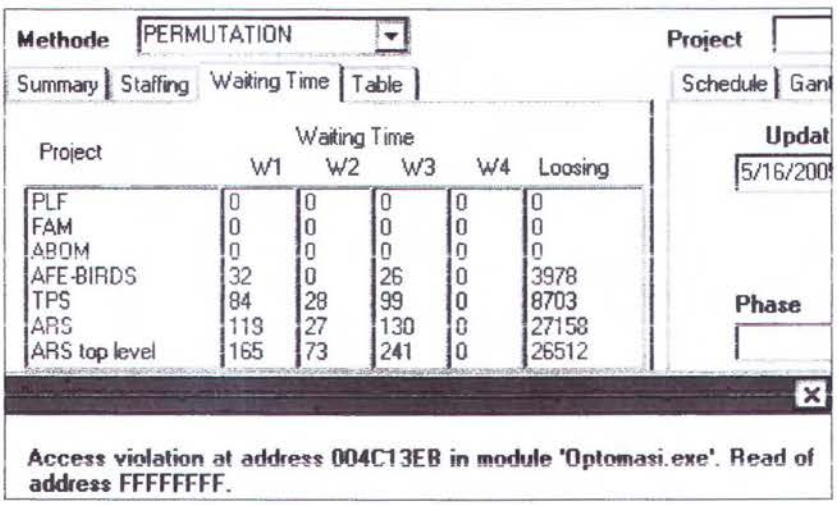
4.9 Penjadwalan dengan 6 dan 7 Proyek

Dengan 6 buah proyek yang bersamaan, metode Permutation masih dapat digunakan, seperti pada gambar 4.69.



Gambar 4.69 Metode Permutasi dengan 6 Buah Proyek

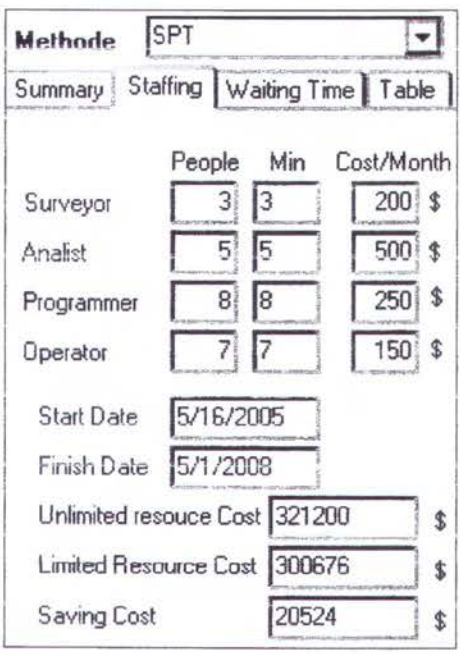
Apabila terdapat lebih dari 6 proyek yang sedang berjalan bersamaan, dengan metode Permutation program Delphi 5.0 tidak dapat melakukan perhitungan atau terjadi error seperti terlihat pada gambar 4.70.



Gambar 4.70 Error Pada Metode Permutasi bila Dimasukkan 7 buah Proyek

Error tidak terjadi pada metode lainnya (SPT, Due Date, dan NPV), hal ini terlihat pada gambar 70 – 73. Pada kasus ini metode yang terbaik adalah **Due Date**,

karena mempunyai Saving Cost yang tertinggi. Untuk kasus lain metode lain (SPT atau NPV) dapat juga menjadi yang terbaik. Pada gambar ini juga terlihat bahwa jumlah sumber daya (people) harus ditambah, tidak boleh kurang dari jumlah minimal kebutuhan proyek yang terbesar (Min).



Gambar 4.71 Metode SPT dengan 7 Proyek

Gambar 4.72 Metode Due Date dengan 7 Proyek

Gambar 4.73 Metode NPV dengan 7 proyek

BAB V

PENUTUP

Dari hasil analisa yang telah dilakukan pada penelitian ini diperoleh beberapa kesimpulan yaitu:

- Dengan menggunakan metode COCOMO II dan melakukan kalibrasi konstanta A dan C dengan data lokal, didapatkan hasil estimasi biaya pengembangan perangkat lunak yang lebih akurat sesuai dengan kondisi organisasi.
- Dengan menggunakan pendekatan heuristik akan didapat penjadwalan multi proyek dengan sumber daya terbatas yang lebih optimal.
- Bila jumlah proyek yang bersamaan kurang dari 7, maka metode Permutation yang terbaik. Bila jumlah proyek 7 atau lebih, metode Permutation tidak dapat digunakan, sehingga metode SPT, Due Date, atau NPV yang lebih baik.
- Untuk mencapai optimal, jumlah staf yang dimiliki harus sama dengan jumlah kebutuhan staf dari proyek yang terbesar.

DAFTAR PUSTAKA

[1]. Ash, Robert C. (1999). *Activity Scheduling In The Dynamic, Multi-Project Setting : Choosing Heuristic Through Deterministic Simulation*. Indiana University.

[2]. Boehm, BW, Abts, C, Brown, AW, Chulani S, Clark, BK, Horowitz, E, Madachy, R, Reifer, DJ, Steece, B. (2000), *Software Cost Estimation With* COCOMO II, Prentice Hall, New Jersey.

[3]. Hamid, Tarek K. Abdel, and Madnick, Stuart E. (1983). *The Dynamic of Software Project Scheduling*. Massachusetts Institute of Technology.

[4]. Khare, Vineet (2002). *Heuristic Scheduling Based on Policy Learning*.The University of Birmingham.

[5]. Pressman, R. (1997). *Software Engineering : A Practice's Approach*. McGraw-Hill Book Company.

[6]. Sauer, J.(1998). *A Multi-Site Scheduling*. University of Oldenburg.

[7]. SEI Capability Maturity Model, www2.umassd.edu/SWPI/processframework/cmm/cmm.html

[8]. Tsoi, Ho Leung. (1999). *A Framework for Management Software Project Development*. Griffith University.