Konfersi ini dapat dilakukan apabila:

1. Telah dilakukan pengecekan sistem scara extensif.
2. Adanya toleransi terhadap waktu tunggu (Time delay).
3. User dipaksa harus menggunakan sistem baru.

Resiko strategi direct cut-over ini adalah sebagai berikut:

1. Delay yang lama sehingga menyebabkan banyaknya terjadi kesalahan.
2. User menggunakan sistem yang belum dikenal
3. User tidak berkesempatan membandingkan antara sistem lama terhadap sistem baru.

c. Phased In Method

Phased in method adalah strategi yang menggabungkan Paralel run dengan Direct Cut-Over dengan mengurangi resiko yang mungkin terjadi. Langkah awal yang dilakukan adalah paralel run dan selanjutnya pada pertengahan periode secara bertahap sistem lama digantikan dengan sistem yang baru.

Kelebihan dari strategi ini adalah:

1. User terlibat dalam konfersi ini.
2. Dapat mendeteksi kesalahan apabila terjadi kesalahan sistem maupun data.

Kekurangan dari strategi ini adalah:

1. Mebutuhkan waktu yang lebih lama.
2. Apabila sistemnya besar maka konfersi ini sulit dilakukan.

d. Pilot approach atau Distributed approach

Strategi konfersi ini dilakukan apabila terdapat beberapa lokasi atau site. Misalnya pada sistem bank, franchise, restoran, supermarket dan lainnya. Pengoperasiannya dilakukan pada suatu site terpilih dan apabila hasilnya memuaskan baru dilakukan konfersi di site lainnya.

2.6 *Open Database Connectivity* (ODBC)[WWW04]

2.6.1 Defenisi ODBC

Open Database Connectivity (ODBC) adalah sebuah standar konektivitas antar mesin basis data. Standar ini menyediakan API yang dapat digunakan untuk menjalankan dan menghubungkan sebuah aplikasi dengan sebuah sistem manajemen basis data (SMBD). Para desainer ODBC membuatnya dengan tujuan agar ODBC terbebas dari penggunaan bahasa pemrograman tertentu, sistem manajemen basis data tertentu, dan sistem operasi tertentu.

2.6.2 Spesifikasi ODBC

Spesifikasi ODBC menawarkan API prosedural untuk menggunakan *query* dengan bahasa SQL untuk mengakses sebuah basis data. Sebuah implementasi ODBC, akan menyediakan satu aplikasi atau lebih, pustaka inti ODBC, dan juga "*driver* basis data". Pustaka inti ODBC, yang bersifat independen terhadap aplikasi dan juga DBMS, bertindak sebagai interpreter antara aplikasi dan juga driver basis data, sementara driver basis data mengandung detail-detail mengenai SMBD tertentu. Sehingga, dengan cara seperti ini, para programmer dapat menulis aplikasi basis data, tanpa harus memahami sistem manajemen basis data tertentu, mengingat semuanya telah ditangani oleh ODBC. Akan tetapi, para pembuat driver basis data ODBC hanya harus mengetahui bagaimana caranya memasukkan driver basis data ke dalam pustaka inti ODBC. Dengan begitu, ODBC ini dapat disebut sebagai sistem yang modular.

2.6.3 Komponen Utama ODBC

ODBC memiliki beberapa komponen utama, yakni sebagai berikut:

a. ODBC API

   Sekumpulan panggilan fungsi, kode-kode kesalahan dan sintaksis SQL yang mendefinisikan bagaimana data dalam sebuah DBMS diakses.

b. Driver basis data ODBC

   Driver berupa *dynamic link library* yang mampu memproses panggilan fungsi ODBC untuk sebuah DBMS tertentu.

c. ODBC Driver Manager

Bertugas untuk memuat driver basis data ODBC yang dibutuhkan oleh aplikasi.

2.7 *Windows Application Programming Interface* [WWW05]

Windows Application Programming Interface yang sering disebut sebagai WinAPI atau Windows API adalah sekumpulan antarmuka pemrograman aplikasi yang dibuat oleh Microsoft dalam inti sistem operasi Microsoft Windows. Akses terhadap elemen sistem operasi yang lebih rendah, seperti halnya yang dibutuhkan oleh *device driver*, tidak disediakan oleh Windows API, tapi disediakan oleh *Windows Driver Foundation* atau *Native API* dalam versi-versi baru Windows.
Agar para pengembang perangkat lunak dapat menggunakan versi-versi Windows baru, Microsoft sering merilis *Software Development Kit* (SDK), yang terdiri atas dokumentasi dan alat bantu untuk membangun aplikasi-aplikasi Windows dengan teknologi terbaru Microsoft Windows.

2.7.1 Versi-versi Windows API

2.7.1.1 Win16 API

Windows 16 API atau Win16 API merupakan API yang digunakan pertama kali pada versi Windows 16-bit. Pada awalnya, Win16 API disebut dengan Windows API, tapi kemudian diubah menjadi Win16 dalam usaha Microsoft untuk membedakannya dengan versi Windows API yang lebih baru yang berjalan pada Windows 32-bit dan Win32 API. Meskipun memiliki ekstensi EXE, sebenarnya Win16 bukanlah berkas yang dapat dieksekusi, melainkan adalah DLL (*Dynamic Linking Library*).

2.7.1.2 Win32 API

Win32 API merupakan antarmuka pemrograman yang terdapat di dalam sistem operasi Windows 32-bit modern. Seperti halnya Win16 API, Win32 API juga mengimplementasikan fungsi-fungsi di dalam DLL (*Dynamic Linking Library*) pada sistem operasi. DLL inti yang dimiliki oleh Win32 API antara lain kernel32.dll, user32.dll, dan gdi32.dll. Win32 pertama kali muncul

pada tahun 1993, saat Windows NT diluncurkan. Windows 95 juga menggunakan Win32 API, dan pada awalnya dikenal dengan sebutan Win32c, di mana huruf "c" di sana merujuk kepada "*compatibility*", tapi istilah ini akhirnya ditinggalkan oleh Microsoft demi konsistensi nama "Win32".

Dalam Windows NT 4.0 dan para penerusnya termasuk di antaranya versi-versi terbaru Windows, panggilan-panggilan Win32 dieksekusi oleh dua modul, yakni csrss.exe (Client/Server Runtime SubSystem) di dalam modus pengguna dan Win32K.sys pada modus kernel.

2.7.1.3 Win32s API

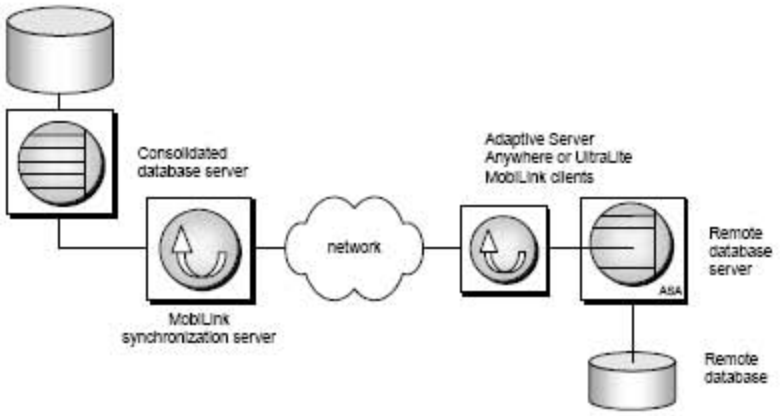
Win32s API merupakan sebuah ekstensi untuk keluarga Windows 3.1x yang mengimplementasikan sekumpulan kecil dari Win32 API untuk sistem-sistem tersebut, yang merupakan sistem operasi 16-bit. Huruf "s" di sana merupakan singkatan dari "subset."

2.7.1.4 Win32 for 64-bit Windows

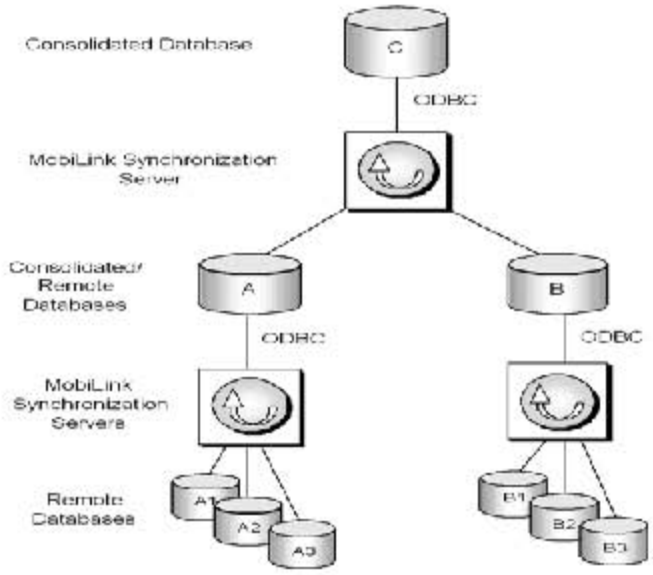
Win32 for 64-bit Windows, yang sebelumnya dikenal dengan sebutan Win64, merupakan sebuah versi Windows API yang ditargetkan untuk digunakan oleh Windows versi 64-bit, yakni Windows XP Professional x64 Edition dan Windows Server 2003 x64 Edition untuk prosesor-prosesor x86-64 dan Windows XP 64-bit Edition dan Windows Server 2003 for Itanium-series untuk prosesor-prosesor IA-64. Dengan kemunculan Win64, Windows NT pun akhirnya masuk ke dalam pasar komputasi 64-bit, dan kompatibilitas aplikasi 32-bit pun masih terjaga. Akan tetapi, memang semua *pointer* memori dialamatkan dengan menggunakan alamat 64-bit, sehingga kode sumber program harus diperiksa ulang untuk melihat apakah ada masalah kompatibilitas dengan *pointer* aritmetika 64-bit dan jika perlu ditulis ulang. Tidak ada penambahan fungsi-fungsi baru yang spesifik ditambahkan ke dalam Windows versi 64-bit.

2.8 MobiLink Synchronization [SYB04]

Mobilink synchronization digunakan untuk sinkronisasi data antara beberapa basis data yang menggunakan *Open Database Connectivity* (ODBC) dengan basis data dari Adaptive Server atau UltraLite. Berikut adalah diagram sistem sinkronisasi dan hirarki dari mobiLink synchronization.



Gambar 2.1 ***Major Parts Synchronization System***



Gambar 2.2 ***MobiLink hierarchy***

Berikut adalah penjelasan dari diagram dan hirarki diatas :

a. Consolidated database adalah sebuah basis data yang berfungsi sebagai pusat data yang menyimpan seluruh informasi tentang sinkronisasi data.

b. Consolidated database server adalah server dari sebuah sistem manajemen basis data yang mengelola consolidated database. Server yang dapat digunakan sebagai consolidated database server adalah semua sistem manajemen basis data yang mendukung ODBC seperti: Adaptive Server Anywhere, Adaptive Server Enterprise, Oracle, IBM DB2 dan Microsoft SQL Server.
c. ODBC connection adalah semua komunikasi diantara MobiLink Synchronization Server dengan consolidated database melalui sebuah koneksi ODBC.
d. MobiLink Synchronization Server adalah server yang mengatur proses sinkronisasi dan menyediakan sebuah antar muka pada semua klien mobilink dan consolidated database server.
e. Network adalah koneksi antara MobiLink Synchronization Server dengan klien.
f. MobiLink client adalah klien yang meminta informasi mengenai sinkronisasi data. Terdapat dua tipe klien dapat menjadi MobiLink client yaitu Adaptive Server Anywhere dan UltraLite.

2.8.1 Konfersi Tipe Data MobiLink Synchronization

Berikut adalah tabel konfersi tipe data yang digunakan untuk sinkronisasi data antara Adaptive Server Anywhere dengan Microsoft SQL Server 2000.

Tabel 2.1 Konfersi Tipe Data Adaptive Server Anywhere dan Microsoft SQL Server 2000

No.	Tipe Data Adaptive Server Anywhere	Tipe Data Microsoft SQL Server
1.	bit	bit
2.	tinyint	tinyint
3.	smallint	smallint
4.	int	int
5.	bigint	bigint
6.	decimal	decimal
7.	numeric	numeric

8.	float	float
9.	real	real
10.	smallmoney	smallmoney
11.	money	money
12.	date	datetime
13.	time	datetime
14.	timestamp	datetime
15.	smalldatetime	datetime
16.	datetime	datetime
17.	char	varchar atau text
18.	character	varchar
19.	varchar	Varchar atau text
20.	long varchar	text
21.	binary	binary atau image
22.	long binary	image
23.	double	float

2.9 Metode Rekayasa Perangkat Lunak dengan Metode *Waterfall* [PRE97]

Metode yang sering juga disebut metode "*waterfall*" atau "*classic life cycle*" ini menggunakan pendekatan yang sistematis dan sekuensial dalam membangun perangkat lunak yang dimulai pada level sistem dan pengembangan melalui tahapan analisis, perancangan, pengkodean, pengujian, dan pemeliharaan.

a. Analisis

Dari rumusan sistem yang diperoleh dari tahap pertama, selanjutnya dilakukan analisis yang berkaitan dengan proses dan data yang diperlukan oleh sistem serta keterkaitannya. Tujuan dilakukan tahapan ini adalah untuk memahami sistem yang ada pada saat ini agar dapat mendefinisikan permasalahan sistem sehingga selanjutnya dapat menentukan kebutuhan sistem secara garis besar sebagai persiapan ke tahap perancangan. Analisis di sini dilakukan dengan pemodelan

menggunakan metode *Data Flow Oriented dengan tool Data Flow Diagram* (DFD).

b. Perancangan

Pada tahap perancangan ini diberikan gambaran umum yang jelas kepada pengguna dan rancang bangun yang lengkap tentang sistem yang akan dikembangkan kepada pihak-pihak yang terlibat dalam pengembangan sistem. Tahap perancangan ini dilakukan sebagai persiapan untuk tahap implementasi.

c. Implementasi / *Coding* / Pemrograman

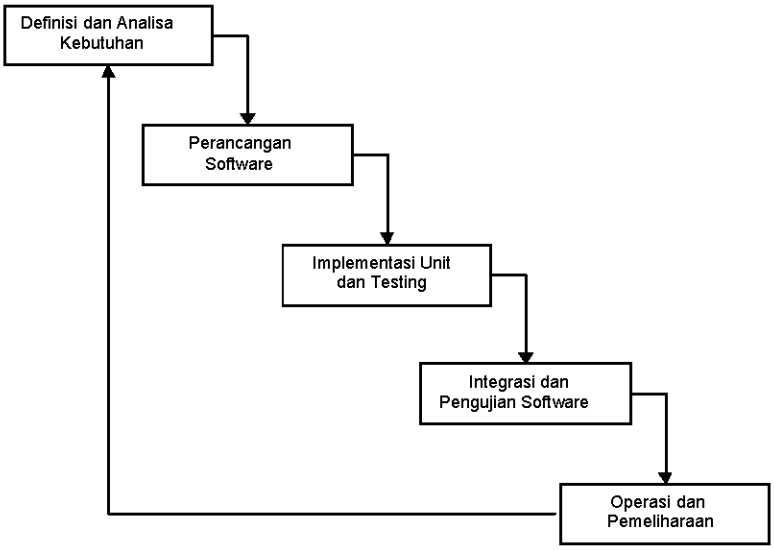
Setelah tahap perancangan sistem, selanjutnya dilakukan konfersi rancangan sistem ke dalam kode-kode bahasa pemrograman yang diinginkan. Pada tahap ini dilakukan pembuatan komponen-komponen sistem yang meliputi modul program, antarmuka dan basis data.

d. Pengujian

Tahap pengujian ini dilakukan untuk mendapatkan serta memastikan bahwa perangkat lunak yang dihasilkan adalah valid dan sesuai dengan kebutuhan yang telah dideskripsikan.

e. Pemeliharaan

Pada tahap pemeliharaan ini perangkat lunak sudah diserahkan kepada pengguna. Pada tahap ini dilakukan evaluasi terhadap sistem yang baru untuk mengetahui apakah sistem telah memenuhi tujuan yang ingin dicapai. Dari hasil evaluasi ini dimungkinkan untuk melakukan perubahan-perubahan yang diperlukan terhadap sistem agar sistem senantiasa dapat digunakan dengan baik.



Gambar 2.3 Metode RPL dengan Metode Waterfall

2.10 Diagram Alir Data (*Data Flow Diagram*) [TRI99]

Diagram alir data (*Data Flow Diagram*) adalah diteknik pemodelan secara grafis yang menggambarkan aliran data dalam sistem serta fungsi-fungsi (proses) yang terlibat dalam transformasi aliran data tersebut. Selain itu, *data flow diagram* (DFD) memberikan informasi tambahan yang digunakan selama tahap analisis. DFD digunakan untuk merepresentasikan sistem atau perangkat lunak pada berbagai tingkatan abstraksi. Artinya, DFD dapat dibagi menjadi beberapa level yang menggambarkan penambahan aliran informasi dan fungsionalitas yang lebih rinci. DFD level 0 (*Data Context Diagram*) merepresentasikan elemen-elemen perangkat lunak atau sistem secara keseluruhan sebagai suatu proses dengan data masukan dan keluaran digambarkan sebagai panah yang masuk dan keluar proses. Selanjutnya pada level yang lebih tinggi (1,2,3,… dan seterusnya), proses tersebut dipecah-pecah untuk memperoleh aliran data dan proses yang lebih rinci.