UNIVERSITAS INDONESIA

RANCANG BANGUN APLIKASI PENDETEKSI INFORMASI JARINGAN GSM BERBASIS SYMBIAN OS

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar menjadi Sarjana Teknik

NOVIANTO WIBOWO

0706199722

PROGRAM STUDI TEKNIK ELEKTRO
FAKULTAS TEKNIK UNIVERSITAS INDONESIA
DEPOK
JUNI 2009

PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya saya sendiri,
dan semua sumber baik yang dikutip maupun dirujuk
telah saya nyatakan dengan benar.

Nama	**: Novianto Wibowo**
NPM	**: 0706199722**
Tanda Tangan	**: .....**
Tanggal	**: 23 Juni 2009**

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh :
Nama : Novianto Wibowo
NPM : 0706199722
Program Studi : Teknik Elektro
Judul Skripsi : **RANCANG BANGUN APLIKASI PENDETEKSI INFORMASI JARINGAN GSM BERBASIS SYMBIAN OS**

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai
bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Teknik pada Program Studi Elektro Fakultas Teknik, Universitas Indonesia

DEWAN PENGUJI

Pembimbing : **Dr. Ing. Kalamullah Ramli M.Eng** (.....)
Penguji : **Dr. Ir. Arman Djohan D.** (.....)
Penguji : **Ir. Arifin Djauhari MT.** (.....)

Ditetapkan di : Depok
Tanggal : 23 Juni 2009

UCAPAN TERIMA KASIH

Puji syukur saya panjatkan kepada Allah SWT, karena atas berkat dan rahmat-Nya, saya dapat menyelesaikan skripsi ini. Penulisan skripsi ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Sarjana Teknik Jurusan Elektro pada Fakultas Teknik Universitas Indonesia. Saya menyadari bahwa, tanpa bantuan dan bimbingan dari berbagai pihak, dari masa perkuliahan sampai pada penyusunan skripsi ini, sangatlah sulit bagi saya untuk menyelesaikan skripsi ini. Oleh karena itu, penulis mengucapkan terima kasih kepada :

Dr. Ing. Kalamullah Ramli M.Eng

Selaku dosen pembimbing yang telah menyediakan waktu, tenaga, dan pikiran untuk mengarahkan saya dalam penyusunan skripsi ini.

Harapan penulis kiranya skripsi ini dapat memberikan pengetahuan yang bermanfaat bagi penulis khususnya dan pembaca pada umumnya. Semoga Allah SWT senantiasa melimpahkan rahmat dan hidayah pada kita semua. Amin.

Depok, 23 Juni 2009

Penulis

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI
TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

Nama : Novianto Wibowo
NPM : : 0706199722
Program Studi : Teknik Elektro
Departemen : Elektro
Fakultas : Teknik
Jenis karya : Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif (*Non-exclusive Royalty-Free Right*)** atas karya ilmiah saya yang berjudul :

"RANCANG BANGUN APLIKASI PENDETEKSI INFORMASI JARINGAN GSM BERBASIS SYMBIAN OS"

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan memublikasikan tugas akhir saya tanpa meminta izin dari saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok
Pada tanggal : 23 Juni 2009
Yang menyatakan

(Novianto Wibowo)

ABSTRAK

Nama : Novianto Wibowo

Program Studi : Teknik Elektro

Judul : Rancang Bangun Aplikasi Pendeteksi Informasi Jaringan GSM Berbasis Symbian OS

Tugas akhir ini membahas tentang bagaimana membangun sebuah aplikasi pada perangkat *mobile* seperti telpon genggam yang memakai Symbian sebagai sistem operasi. Aplikasi dibangun menggunakan *tools* pengembangan berbasis Symbian S60 3rd Ed yang berfungsi sebagai *library interface* dan Carbide C++ sebagi *editor* kodenya. Aplikasi yang dibangun bertujuan untuk mendeteksi informasi jaringan GSM seperti *Cell Id*, *Network Id*, *Country Id* dan *Network Name*. Proses pembuatan aplikasi meliputi pengeditan kode sumber *(coding)*, merubahnya menjadi kode tereksekusi *(building)*, uji coba aplikasi pada emulator *(testing & debugging)*, kemudian proses registrasi aplikasi yaitu menggunakan metode *open signed online (signing)* dan yang terakhir adalah pengiriman aplikasi ke perangkat *(transferring, distributuing)*. Untuk menguji kehandalan aplikasi dilakukan proses uji coba yaitu dengan menggunakan tidak hanya satu operator saja tetapi tiga operator yang bertujuan untuk mengetahui apakah aplikasi bisa mendeteksi jaringan operator yang berbeda. Pengambilan data dilakukan dengan metode acak mengambil sampel 15 titik daerah dan hasilnya ada beberapa titik daerah yang diuji mempunyai nilai *cell id* yang berbeda seperti yang terlihat pada daerah Universitas Gunadarma (Telkomsel : 6700714) dan Departemen Teknik Elektro UI (Telkomsel : 6700713). *Cell id* merupakan metode paling dasar untuk *positioning*. Dan selanjutnya bisa digunakan untuk mengembangkan suatu aplikasi berbasis lokasi yang memudahkan pengguna jaringan seluler untuk mengetahui informasi posisi dimana berada sekarang. Untuk masing – masing operator *cell id* yang memiliki prosentase *coverage* tertinggi adalah :

- Telkomsel (*Cell Id* : 6700713) = 30%
- Indosat (*Cell Id* : 6790319) = 20%
- Exelcomindo (*Cell Id* : 249829) = 20%

Kata Kunci : Symbian , Parmeter jaringan GSM, Symbian S60 3rd Ed, Carbide C++, *Cell Id*, aplikasi berbasis lokasi.

ABSTRACT

Name : Novianto Wibowo

Study Program : Electrical Engineering.

Title : Design and Development Application to Detect GSM Network Information with the Symbian Platform.

The focus of this final project is to develop application in mobile equipment for example hand phone with Symbian operating system. The application is built using Symbian S60 3rd Ed development tools as library interface and Carbide C++ as code editor. The purpose of this application is to detect GSM network information's cell Id, Network Id, Country Id and Network Name. The development phases include editing the source code (coding), creating to become executable code (building), software testing in the emulator (testing & debugging) then software registration process using open signed online (signing) and, last, software deployment process to the device i.e. transferring & distributing. To verify reliability of the software, testing has been performed not only through one but three operators. The purpose is to verify if the software can detect different network or not. The data were collected with random method and taking sample of 15 region nodes. The result shows that some region has different cell id value, for example in Gunadarma University (Telkomsel: 6700714) whereas at Electrical Engineering Department of UI (Telkomsel: 6700713). Cell Id contains basic information for positioning. For further phase, application can be developed for location base services, to make it easier for users to know their own position. For each operator the highest percentage of coverage is listed as follows:

- Telkomsel (*Cell Id* : 6700713) = 30%
- Indosat (*Cell Id* : 6790319) = 20%
- Exelcomindo (*Cell Id* : 249829) = 20%

Keywords: Symbian, GSM Network Parameter, Symbian S60 3rd Edition, Carbide C++, Cell Id, location base services.

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

DAFTAR SINGKATAN

LBS	: *Location Based Service*
GSM	: *Global System Mobile*
BSS	: *Base Station System*
NSS	: *Network Station System*
OSS	: *Operation and Support System*
PLMN	: *Public Land Mobile Network*
ME	: *Mobile Equipment*
SIM	: Subscriber Identity Module
IMMSI	: *International Mobile Subscriber Identity*
MSISDN	: *Mobile Subscriber ISDN*
BTS	: *Base Transceiver Station*
BSC	: *Base Station Controller*
MSC	: *Mobile Switching Center*
HLR	: *Home Location Register*
VLR	: *Visitor Location Register*
AuC	: *Authentication Center*
EIR	: *Equipment Identity Registration*
IMEI	: *International Mobile Equipment Identity*
PIN	: *Personal Identity Number*
TA	: *Timing Advanced*
RTT	: *Round Trip Time*
NMR	: *Network Measurement Report*
OTD	: *Observe Time Difference*
A-GPS	: *Assisted-Global Positioning System*
LTP	: *Location Technology Providers*
NO	: *Network Operators*
SP	: *Service Provider*
API	: *Applications Programming Interface*
SDK	: *Software Development Kit*
DDK	: *Driver Development Kit*
IDE	: *Integrated Development Environment*
SIS	: *Symbian Installation System*
JRE	: *Java Runtime Environment*
GUI	: *Graphic User Interface*
MNC	: *Mobile Network Code*
MCC	: *Mobile Country Code*

BAB 1
PENDAHULUAN

1.1. Latar Belakang

Pada era digital seperti sekarang banyak perangkat yang dibuat untuk membantu memudahkan pekerjaan manusia. Hal ini dikarenakan semakin meningkatnya akan hal – hal yang praktis dan cepat. Demikian juga dengan kasus diatas, banyak orang mulai memikirkan cara – cara yang efektif dan efisien untuk membuat suatu aplikasi yang menyediakan layanan yang dapat membantu dalam mencari atau menentukan posisi seseorang. Dalam hal ini telepon genggam menjadi alternatif sarana perantara utama, karena melihat kenyataan bahwa pada saat ini hampir semua orang mempunyai dan menggunakan telepon genggam.

Aplikasi *Mobile* berbasis lokasi ini sebagai antarmuka perangkat kerasnya menggunakan telepon pintar, beberapa produk telepon genggam telah beredar dipasaran yang menggandeng sistem operasi baik yang bersifat komersial maupun yang bersifat terbuka lisensinya, adapun beberapa nama system operasi pada telepon genggam ataupun telepon pintar adalah :

- Palm / PalmOne, biasanya digunakan oleh HP Treo, Tungsten, Zire
- Pocket PC, dikembangkan oleh Microsoft dari Windows CE
- Symbian, yang paling sering digunakan oleh Nokia
- Linux for Mobile, digunakan oleh Motorola

Symbian merupakan system operasi yang sangat populer dikalangan pengguna telepon genggam, karena paling sering digunakan sebagai sistem operasi dari produk terkenal Nokia. Dengan kepopuleran Symbian sebagai sistem operasi pada telepon genggam banyak pemrogram yang melirik pembangunan perangkat lunak mobile yang di jalankan di atas system operasi Symbian.

Pada perancangan aplikasi yang dikembangkan ditujukan untuk pengguna telepon genggam merk Nokia. Pengembangan sistem adalah Symbian OS versi 9.1, dan karena mayoritas *handset* Symbian yang beredar di pasaran Indonesia adalah Symbian dengan *S60 Series User Interface Platforms*, sehingga pengembangan aplikasi pendeteksi CellId menggunakan platform *Symbian OS v9.1 – S60 3rd Edition SDK*. Bahasa pemrograman yang digunakan adalah C++. Metode pendeteksian *cell-id* dilakukan melalui

CTelephony::GetCurrentNetworkInfo() API. Pemilihan IDE (*Integrated Development Environment*) yang digunakan untuk pengembangan sistem jatuh kepada *Carbide C++* karena sebagai salah satu IDE *Symbian C++* yang cukup populer, juga memiliki fasilitas *On-Device-Debugging* yang sangat berguna untuk penelusuran kesalahan dan pengecekan performa aplikasi.

1.2. Batasan Masalah

Aplikasi yang dibuat ini akan menampilkan informasi dari BTS yang terdeteksi, pada saat aplikasi ini dijalankan di telepon genggam. Tidak disertakan penjelajah peta pada aplikasi ini dikarenakan kurangnya literature ataupun referensi mengenai aplikasi *Location Based Services* (LBS) mengunakan Symbian Carbide C++ beserta tools pendukungnya.

1.3. Tujuan Penulisan

Tugas akhir ini bertujuan untuk membuat aplikasi yang dapat menampilkan informasi Cell Id pada BTS yang terdeteksi melalui telepon genggam yang digunakan oleh pengguna aplikasi ini. Aplikasi ini diharapkan akan dapat menjadi salah satu contoh layanan LBS, yang akan mempunyai nilai tambah jika diterapkan oleh operator seluler untuk menyediakan layanan bagi masyarakat luas.

1.4. Metodologi Penulisan

Metode yang digunakan dalam penulisan tugas akhir ini adalah observasi lapangan dan didukung dengan studi literatur. Adapun prosesnya adalah sebagai berikut :

- Penekanan pada penggunaan software Symbian Carbide C++ beserta *tools* pendukungnya.
- Merumuskan permasalahan,
- Melakukan pengumpulan data melalui penelusuran di Internet
- Merancang sistem dengan mempersiapkan perangkat keras maupun lunak serta menghubungkan perangkat – perangkat yang diperlukan.

- Pembuatan aplikasi berupa file SIS yang siap diinstal pada telepon genggam Nokia yang mengunakan sistem operasi *Symbian S60 3rd Edition* SDK.
- Pengujian kinerja pada aplikasi yang telah dibuat.
- Evaluasi dan perbaikan.

1.5. Sistematika Penulisan

Laporan tugas akhir ini terdiri atas lima bab, dengan pembahasan secara garis besar adalah sebagai berikut :

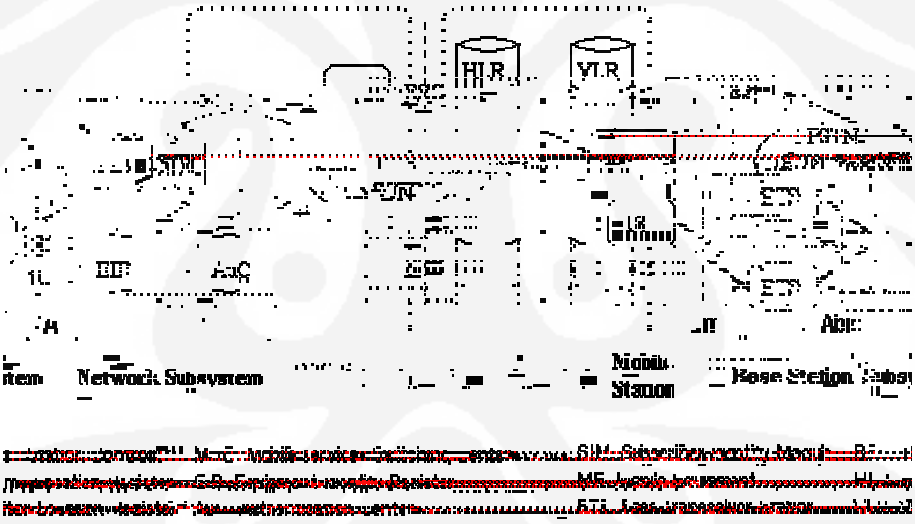
- BAB I : PENDAHULUAN, meliputi Latar Belakang Permasalahan, Batasan Masalah, Tujuan Penulisan, Metodologi Penulisan, dan Sistematika Penulisan.
- BAB II : DASAR TEORI, meliputi konsep dasar *Location Based Services*, pengenalan Carbide C++ beserta *tools* pendukungnya seperti *S60 3rd Edition Maintenance Release* SDK, *ActivePerl*, *Java Run – Time Environment.*
- BAB III : PERANCANGAN APLIKASI, menjelaskan rencana perancangan aplikasi yang akan dibuat.
- BAB IV : IMPLEMENTASI DAN PENGUJIAN APLIKASI, menjelaskan cara kerja dan implementasi aplikasi sekaligus membahas hasil evaluasi dari implementasi aplikasi.
- BAB V : KESIMPULAN DAN SARAN, berisi kesimpulan yang didapat dari pelaksanaan tugas akhir, dan saran – saran yang dapat digunakan untuk pengembangan lebih lanjut dari hasil tugas akhir ini.

BAB 2
DASAR TEORI

2.1 Sistem Komunikasi GSM

Penelitian tentang komunikasi bergerak telah ada sejak lama dimulai dari generasi pertama, generasi kedua, generasi ketiga dan seterusnya. GSM yang merupakan generasi kedua, dan awalnya merupakan jawaban terhadap masyarakat eropa akan kebutuhan komunikasi bergerak yang kompatibel. Selain itu GSM juga merupakan salah satu teknologi komunikasi bergerak yang sekarang ini banyak digunakan dan dikenal oleh masyarakat indonesia.

Berbicara mengenai GSM tidak terlepas dari arsitektur jaringan GSM itu sendiri yang terdiri dari BSS *(Base Station System)*, NSS, *(Network Station System)*, dan OSS *(Operation and Support System)*.



Gambar 2.1. *Layout generic* dari jaringan GSM menurut John's Scourias

Secara umum, *network element* dalam arsitektur jaringan GSM dapat dibagi menjadi:

1. *Mobile Station (MS)*
2. *Base Station Sub-system (BSS)*
3. *Network Sub-system (NSS),*
4. *Operation and Support System (OSS)*

Secara bersama-sama, keseluruhan *network element* di atas akan membentuk sebuah *PLMN (Public Land Mobile Network).*

***Mobile Station* atau MS** merupakan perangkat yang digunakan oleh pelanggan untuk melakukan pembicaraan. Terdiri atas:

- *Mobile Equipment (ME)* atau *handset*, merupakan perangkat GSM yang berada di sisi pengguna atau pelanggan yang berfungsi sebagai terminal *transceiver* (pengirim dan penerima sinyal) untuk berkomunikasi dengan perangkat GSM lainnya.
- Subscriber Identity Module (SIM) atau *SIM Card*, merupakan kartu yang berisi seluruh informasi pelanggan dan beberapa informasi pelayanan. ME tidak akan dapat digunakan tanpa SIM didalamnya, kecuali untuk panggilan darurat. Data yang disimpan dalam SIM secara umum, adalah:
    1. IMMSI *(International Mobile Subscriber Identity)*, merupakan penomoran pelanggan.
    2. MSISDN *(Mobile Subscriber ISDN)*, nomor yang merupakan nomor panggil pelanggan.

***Base Station System* atau BSS**, terdiri atas:

- BTS *Base Transceiver Station*, perangkat GSM yang berhubungan langsung dengan MS dan berfungsi sebagai pengirim dan penerima sinyal.
- BSC *Base Station Controller*, perangkat yang mengontrol kerja BTS-BTS yang berada di bawahnya dan sebagai penghubung BTS dan MSC

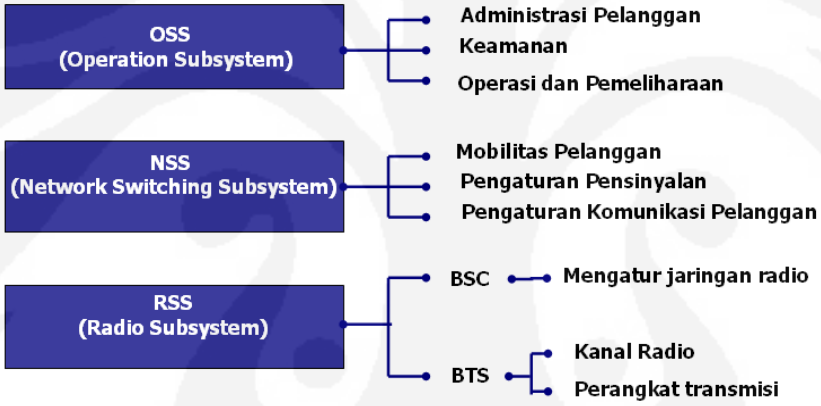
Network Sub System atau NSS, terdiri atas:

- *Mobile Switching Center* atau MSC, merupakan sebuah *network element central* dalam sebuah jaringan GSM. MSC sebagai inti dari jaringan seluler, dimana MSC berperan untuk interkoneksi hubungan pembicaraan, baik antar selular maupun dengan jaringan kabel PSTN, ataupun dengan jaringan data.
- *Home Location Register* atau HLR, yang berfungsi sebagai sebuah *database* untuk menyimpan semua data dan informasi mengenai pelanggan agar tersimpan secara permanen.
- *Visitor Location Register* atau VLR, yang berfungsi untuk menyimpan data dan informasi pelanggan.

- *Authentication Center* atau AuC, yang diperlukan untuk menyimpan semua data yang dibutuhkan untuk memeriksa keabsahaan pelanggan. Sehingga pembicaraan pelanggan yang tidak sah dapat dihindarkan.
- *Equipment Identity Registration* atau EIR, yang memuat data-data pelanggan.

***Operation and Support System* atau OSS**, merupakan sub sistem jaringan GSM yang berfungsi sebagai pusat pengendalian, diantaranya *fault management, configuration management, performance management,* dan *inventory management.*

Dimana gambaran umum masing-masing fungsinya dapat dilihat pada Gambar 2.2 berikut ini:



Gambar 2.2 Fungsi Subsistem GSM

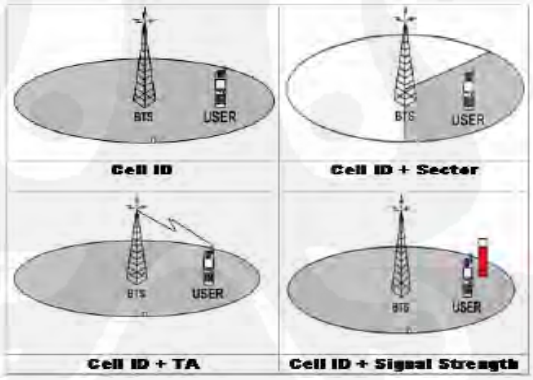
2.2 Konsep Dasar LBS (*Location Based Service*)

LBS *(Location Based Service)* merupakan suatu layanan yang bereaksi aktif terhadap perubahan entitas posisi sehingga mampu mendeteksi letak objek dan memberikan layanan sesuai dengan letak objek yang telah diketahui tersebut. Pada teknologi LBS berbasis jaringan seluler, penentuan posisi sebuah peralatan komunikasi bergerak ditentukan berdasarkan posisi relatif peralatan tersebut terhadap lokasi BTS *(Base Transceiver Station)*. Dalam menentukan posisi dari sebuah telepon genggam yang sedang aktif, secara umum terdapat tiga tingkat metode yang digunakan saat ini, yaitu :

a. Metode *Basic Positioning* yang Berbasis Pada *Cell Identification (Cell ID)*

Penentuan posisi didasarkan pada daerah geografis yang tercakup oleh sebuah *cell* berhubungan dengan daerah cakupan dari sinyal radio. Ketika sebuah handphone terhubung secara aktif dengan sebuah *base station*, berarti handphone tersebut diasumsikan berada dalam *cell* dari base station tersebut. Untuk mengukur jarak dan arah *handset* dari *base station* tidak dapat diketahui dengan pasti. Oleh karena itu, untuk lebih meningkatkan lagi akurasi hasil pencarian, metode *Cell ID* ini seringkali dikombinasikan dengan metode lain misalnya :

- *Timing Advanced (TA)*, dengan menggunakan TA ini, metode *Cell ID* akan ditambahkan sebuah fungsionalitas untuk menghitung *Round Trip Time (RTT)*, yaitu waktu transmisi sebuah *frame* (dari *base station* ke *handphone*) dan waktu penerimaan sebuah *frame* (dari handphone ke base station). Dengan tambahan metode ini, jarak antara *handphone* dan *base station* dapat ditentukan dengan keakuratan 50 m.
- *Network Measurement Report (NMR)*, dengan berdasar pada besar kecilnya sinyal *(Received Signal Strength)* yang diterima handphone yang ada di suatu *"sector cell"*, maka posisi itu dapat ditentukan lebih akurat.



Gambar 2.3. Metode *Basic positioning* yang Berbasis Pada *Cell Id*.

b. Metode *Enhanced Positioning*

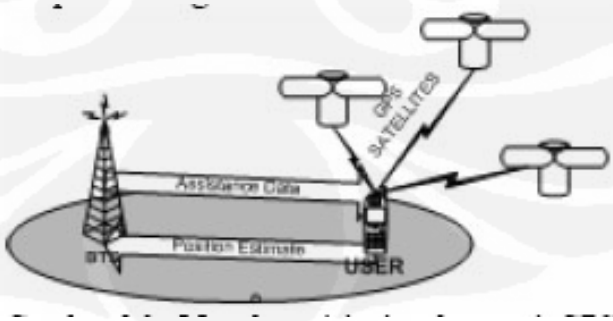
Pada umumnya metode ini menggunakan pendekatan *Observe Time Difference (OTD)*. Dalam jaringan GSM yang sering digunakan adalah *Enhanced-OTD (E-OTD)*. E-OTD adalah metode pencarian posisi yang berdasarkan pada waktu. Untuk menentukan posisi relatif, sebuah

handphone harus aktif terhadap tiga *base station* dan perlu ditentukan terlebih dahulu jarak *handphone* terhadap masing-masing *base station* berdasarkan waktu yang ditempuh oleh sebuah sinyal dari handphone ke masing-masing *base station.* Selanjutnya, dengan menggunakan rumus matematika untuk triangulasi, maka dapat ditentukan posisi dari *handphone* yang sedang aktif tersebut. Dengan menggunakan metode ini akurasinya akan meningkat hingga memiliki ketelitian sampai kurang dari 50m.

Gambar 2.4. Metode *Enhanced Positioning*

c. Metode *Advanced Positioning*

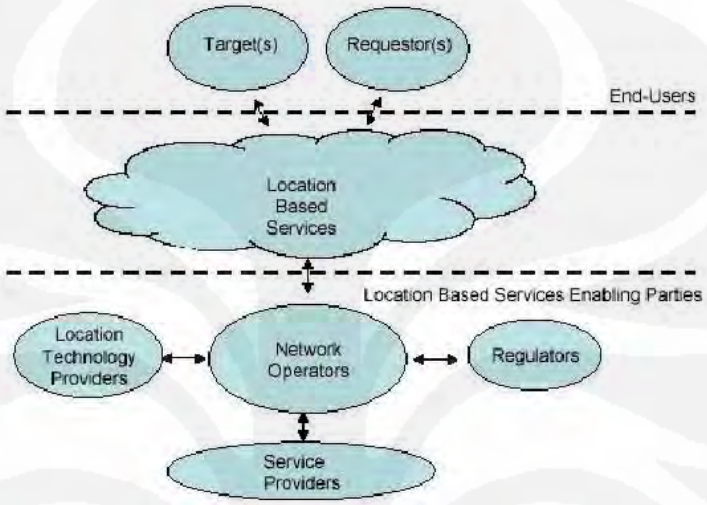
Pada umumnya menggunakan teknologi *Assisted-Global Positioning System (A-GPS)*. A-GPS juga merupakan metode yang berbasis pada waktu. Pada metode ini, akan dilakukan pengukuran waktu tiba dari sebuah sinyal yang dikirim dari tiga buah satelit GPS. Hal ini berarti *handset* harus memiliki fasilitas untuk mengakses GPS. A-GPS juga menghasilkan akurasi secara vertikal dan estimasi jarak yang baik. Akurasinya pun sampai kurang dari 10m.



Gambar 2.5. Metode *Advanced Positioning*

2.3. Pihak - Pihak Pendukung LBS

Adapun untuk dapat menyediakan layanan yang terpadu, LBS harus melibatkan berbagai pihak yang mempunyai peranan masing – masing seperti dijelaskan pada gambar 2.6.



Gambar 2.6. *Players in LBS, GSM MoU Association. GSM Association Permanent Reference Document : SE.23,* (*GSM World Website*, January 2003) figure 4-1. http://www.gsmworld.com/documents/lbs/se23310.pdf

LBS menghubungkan dua pihak, yaitu pengguna (*end-users*) dan pihak – pihak yang menyediakan LBS (*LBS Enabling Parties*). *End-Users* terdiri dari *target*, yaitu pengguna yang posisinya dicari, dan *requestor* , yaitu pengguna yang mencari posisi seseorang atau suatu layanan.

Secara umum ada empat pihak yang saling bekerja sama dalam menyediakan LBS yaitu *Location Technology Providers (LTP), Network Operators (NO), Regulators,* dan *Service Providers (SP).*

2.3.1. *Location Technology Providers (LTP)*

LTP terdiri dari produsen perangkat keras maupun lunak yang memungkinkan pencarian posisi berbasis jaringan dari suatu *handset.* Sistem pencarian posisi membutuhkan kemampuan yang spesifik dari jaringan maupun *handset.*

2.3.2. *Network Operators (NO)*

NO merupakan perusahaan yang memiliki infrastruktur jaringan telekomunikasi, misalnya GSM. NO mampu mengimplementasikan LBS secara efektif pada jaringannya. Tanggung jawab utamanya adalah melindungi pelanggan dari penggunaan data posisi yang tidak semestinya.

2.3.3. *Regulators*

Regulators adalah pihak yang menentukan hukum atau peraturan – peraturan yang menentukan bagaimana LBS bisa diimplementasikan secara legal. Hal terpenting yang perlu diperhatikan adalah privacy dari pengguna, dimana setiap negara mempunyai aturannya sendiri mengenai hal ini.

2.3.4. *Service Provider*

SP merupakan perusahaan yang membuat dan atau menyediakan LBS, yang digunakan melalui NO. SP menyediakan tampilan bagi end-users, hubungan antara LBS, sistem NO, dan sebagainya. SP tidak harus mempunyai data dan infrastruktur untuk menawarkan LBS kepada pelanggan.

2.4. Symbian OS

Symbian OS adalah sistem operasi tak bebas yang dikembangkan oleh Symbian Ltd. yang dirancang untuk digunakan peralatan bergerak (*mobile*). Sebelum Nokia mengumumkan pembelian seluruh sisa saham Symbian Ltd. yang tidak dimilikinya pada 24 Juni 2008, Symbian dimiliki Nokia (47,9%), Ericsson (15,6%), Panasonic (10,5%), Samsung (4,5%), Siemens/BenQ (8,4%), Sony Ericsson (13,1%) [1] . Versi Symbian yang terbaru adalah Symbian OS v9.5s. Sedangkan ponsel yang paling banyak beredar saat ini menggunakan Symbian OS v6.1s, v7.0s, RV 47 75, v8.OS, dan v9.1s. Nokia Nseries rata-rata menggunakan Symbian OS v9.1s, kecuali Nokia N95 yang menggunakan Symbian OS v9.2s.

2.4.1.Gambaran Umum

Saat ini Symbian OS banyak telah banyak digunakan oleh berbagai *vendor* produk peralatan komunikasi bergerak pada berbagai jenis produk mereka yang

bervariasi. Variasi dari sisi perangkat keras ini dimana Symbian OS diimplementasi dapat dimungkinkan karena sistem operasi ini memiliki antarmuka pemprograman aplikasi (*Application Programming Interface; API*). API mendukung terhadap komunikasi dan tingkah laku yang umum pada hardware yang dapat digunakan oleh objek aplikasi lain. Hal ini dimungkinkan karena API merupakan objek antarmuka yang didefenisikan pada level aplikasi, yang berisikan prosedur dan fungsi (dan juga variabel serta struktur data) yang mengelola/memanggil [kernel] dimana sebagai penghubung antara perangkat lunak dan keras. Dengan adanya standar API ini membantu pihak pengembang untuk melakukan penyesuaian atas aplikasi yang dibuatnya agar dapat diinstal pada produk telepon bergerak yang bermacam-macam.

Mirip seperti sistem operasi pada komputer, Symbian OS mampu melakukan operasi secara *multithreading*, *multitasking* dan pengamanan terhadap memori. Dan semua pemrograman pada Symbian dilakukan secara *event-based*, artinya *hardware* CPU menjadi tidak aktif ketika tidak ada inputan berupa aktifitas tertentu. Namun perlu dipahami sistem operasi ini memang ditujukan untuk diinstal pada peralatan mobile dengan keterbatasan sumber daya. *Multithread* dan *multitasking* memberikan kemampuan Symbian OS untuk menjalankan lebih dari satu aplikasi sekaligus.

Namun khusus ini, adanya *preemptive multitasking* kernel akan memberi tiap-tiap program suatu pembagian waktu pemprosesan yang dilakukan bergantian dengan cepat sehingga nampak bagi pemakai seolah-olah proses ini dieksekusi secara bersamaan. Untuk itu telah didefinisikan penjadwalan berdasar prioritas tertentu untuk menentukan proses mana yang berjalan terlebih dahulu dan proses apa berikutnya serta berapa banyak waktu akan jadi diberi.

Symbian OS sendiri bukanlah software yang sifatnya *open source* secara penuh karena meskipun terdapat ketersedian API dan dokumentasinya, yang banyak membantu pihak pengembang aplikasi untuk membuat software yang berjalan di atas sistem operasi ini, dipublikasi untuk umum namun tidak untuk kode *source* sendiri.

2.4.2. Arsitektur Sistem Operasi

Secara umum arsitektur Symbian OS sendiri dapat gambarkan menjadi empat lapisan berdasarkan penggunaan API yang tersedia, yaitu :

2.4.2.1. Lapisan pendukung aplikasi (*Application Utility Layer*)

Lapisan ini terdiri dari berbagai pendukung yang berorientasi pada aplikasi. Hal ini memungkinkan aplikasi lain (diluar sistem operasi) untuk berintegrasi dengan aplikasi dasar yang tersedia pada sistem operasi. Bentuk layanan lain termasuk proses pertukaran data dan manajemen data.

2.4.2.2 Lapisan layanan dan framework antarmuka grafis (*GUI Framework*)

Lapisan ini merupakan framework API yang tersedia untuk memberi dukungan terhadap penanganan input user secara grafis maupun suara yang dapat digunakan oleh aplikasi lain.

2.4.2.3. Lapisan komunikasi

Lapisan ini berfungsi sebagai sistem operasi yang fokus diimplementasi pada peralatan komunikasi mobile, Symbian OS memiliki kumpulan API yang fokus pada lapisan komunikasi. Bagian teratas pada lapisan ini terdapat dukungan pencarian dan pengiriman pesan teks. Berikutnya adalah antarmuka yang memberi dukungan komunikasi seperti Bluetooth dan infrared (IrDA) serta USB. Yang terakhir pada lapisan ini adalah protokol komunikasi berupa TCP/IP, HTTP, WAP dan layanan telepon.

2.4.2.4. Lapisan sistem API dasar

Lapisan ini merupakan kumpulan API yang mendukung pengasksesan data memori, tanggal dan waktu, serta sistem dasar lainnya.

2.4.3 Klasifikasi Sistem Operasi

Klasifikasi ini berdasar fungsionalitas dan hak akses dari API tertentu. Tujuan dari pendefinisian sistem ini selain untuk membedakan API mana saja yang bisa diakses oleh aplikasi yang dibuat oleh pihak pengembang aplikasi, juga

tetap memelihara integrasi dari layanan yang disediakan bagi pihak pengembang aplikasi dengan API yang umum digunakan. Hal ini juga dilakukan untuk memaksimumkan interoperabilitas antara berbagai produk yang menggunakan Symbian OS.

Terdapat empat kategori dalam klasifikasi API yang tersedia, yaitu:

2.4.3.1.(API) Symbian Umum

Komponen ini merupakan komponen (API) inti dari Symbian OS. Setiap pengembang aplikasi dapat berasumsi bahwa komponen ini terdapat pada setiap versi Symbian OS sehingga dapat digunakan pada setiap perangkat telepon bergerak yang menggunakan Symbian OS sebagai sistem operasinya. Dengan kata lain setiap kode program yang hanya menggunakan API pada kategori ini dapat dikompile dan dijalankan tanpa kesalahan pada setiap telepon yang menggunakan Symbian OS. Dengan adanya lisensi kerjasama, pengembang aplikasi dapat menambahkan dengan syarat tidak mengganti ataupun mengubah fungsi API standar yang dikategorikan pada bagian ini.

2.4.3.2. (API) Symbian Umum Tergantikan

Komponen yang memerlukan kostumisasi dari komponen Symbian Umum yang diperlukan untuk bekerja dengan ROM dari sistem dimana ia diinstal. Komponen ini merupakan komponen yang bekerja pada *low-level* dari hardware tertentu. Untuk mendapatkan komponen ini pihak pengembang aplikasi memerlukan lisensi dengan pihak Symbian karena versi komponen ini disediakan oleh pihak Symbian. Namun pada dasarnya komponen ini merupakan komponen standar (umum) yang tersedia pada semua versi Symbian OS.

2.4.3.3. (API) Symbian Opsional

Komponen-komponen ini sifatnya opsional (tidak selalu ada) pada semua versi Symbian OS. Namun jika tersedia, maka pengembang aplikasi mendapat jaminan bahwa aplikasinya dapat menggunakan API pada kategori ini pada versi Symbian OS yang sama.

2.4.3.4. (API) Symbian Opsional Tergantikan

Bentuk kategori ini mirip dengan kategori Symbian Opsional adalah kumpulan API yang tidak terikat dengan API umum yang ada pada versi Symbian OS dan dapat ditambahkan oleh pihak pengembang dengan suatu lisensi dari pihak Symbian.

2.5. *Software Development Kits* (SDK)

A **software development kit (SDK** atau **"devkit")** secara khusus adalah sebuah kumpulan alat pengembangan yang memungkinkan seorang *software engineer* untuk membuat aplikasi untuk beberapa paket perangkat lunak, kerangka perangkat lunak, *hardware platform*, sistem komputer, *video game* konsol, atau sistem operasi [2].

Ini Mungkin sesuatu yang sederhana seperti sebuah antarmuka pemrograman aplikasi (API) dalam bentuk file ke beberapa antarmuka tertentu ke bahasa pemrograman atau termasuk canggih untuk berkomunikasi dengan perangkat keras tertentu dalam sebuah *embedded system*. Biasanya alat bantu termasuk debugging dan utilitas sering disajikan dalam sebuah lingkungan pengembangan terpadu (IDE). SDKs sering juga termasuk contoh kode dan catatan teknis pendukung atau dokumen pendukung lainnya untuk membantu menjelaskan poin dari bahan acuan utama.

Seorang *software engineer* biasanya menerima SDK dari target pengembang sistem . SDK seringkali dapat didownload secara langsung melalui Internet. Banyak SDKs disediakan secara gratis untuk mendorong pengembang untuk menggunakan sistem atau bahasa. Kadang - kadang ini digunakan sebagai alat pemasaran. Misalnya, *Foo Produk* mungkin menyediakan *Widget SDK* gratis untuk mendorong orang untuk menggunakannya. Dengan bisa memprogram *Widget SDK* secara gratis secara tidak langsung akan mendorong mereka membeli lebih banyak widget .

SDK untuk sebuah sistem operasi *add-on* (misalnya, *QuickTime* untuk *Mac OS*) termasuk tambahan pada perangkat lunak itu sendiri, yang akan digunakan untuk tujuan pengembangan, jika tidak perlu untuk didistribusikan kembali. Timbul situasi yang menarik di sini antar *platform* dimana adalah

mungkin untuk mengembangkan aplikasi yang setidaknya dapat dimulai pada sebuah sistem tanpa konfigurasi tambahan terinstal, dan menggunakan gaya *Gestalt run-time environment query* untuk menentukan jika *add-on* ada, dan aplikasi mana yang akan gagal untuk memulai. Dengan kata lain, kemungkinan untuk membangun *single biner* yang akan dijalankan di konfigurasi dengan atau tanpa add-on ini, walaupun operasi dikurangi dengan fungsionalitas dalam situasi yang kedua.

Penyedia layanan SDKs untuk system tertentu atau subsystems terkadang bisa mengganti istilah yang lebih spesifik, bukan perangkat lunak. Sebagai contoh, baik Microsoft dan Apple menyediakan ***Driver Development Kit(DDK)*** untuk mengembangkan perangkat *driver*.

Symbian memiliki beberapa jenis SDK untuk berbagai platform User Interface (UI), yaitu : UIQ, S60, Nokia Series 80, dan Nokia 7710. Pembahasan kali ini akan menitik beratkan kepada pengembangan aplikasi berbasis S60, dengan pertimbangan banyaknya perangkat *SmartPhone* yang menggunakan platform ini. Beberapa edisi S60 SDK, versi Symbian OS, dan perangkat handphone yang didukung:

- S60 3rd Edition - Symbian OS v9.1
  Nokia E60, Nokia E61, Nokia E70, Nokia 3250, Nokia N71, Nokia N80, Nokia N91, Nokia N92, dll
- S60 2nd Edition with Feature Pack 3 - Symbian OS v8.1
  Nokia N70, Nokia N90, dll
- S60 2nd Edition with Feature Pack 2 - Symbian OS v8.0a
  Nokia 6630, Lenovo P930, Nokia 6680, Nokia 6681, Nokia 6682, dll
- S60 2nd Edition with Feature Pack 1 - Symbian OS v7.0s enhanced
  Nokia 3230, Nokia 6670, Nokia 7610, Nokia 6620, Nokia 6260, Panasonic X700, Panasonic X800, Samsung SDH-D720, dll
- S60 2nd Edition - Symbian OS v7.0s
  Nokia 6600, dll
- S60 1st Edition - Symbian OS v6.1
  Nokia 7650, Nokia 3650, Nokia 3600Nokia 3660, Nokia 3620, Nokia N-Gage, Nokia N-Gage QD, Sendo X, Siemens SX1, dll

Tabel 2.1 memuat informasi mengenai perbandingan fitur pada beberapa edisi SDK :

Tabel 2.1. Perbandingan Fitur pada beberapa edisi SDK

Features Support Summary	1st Edition	1st Edition, Feature Pack 1	2nd Edition	2nd Edition, Feature Pack 1	2nd Edition, Feature Pack 2	2nd Edition, Feature Pack 3	3rd Edition	3rd Edition, Feature Pack 1
TCP/IP connectivity	Plug-In	•	•	•	•	•	•	•
Configuration and monitoring tool					•	•	•	•
Generic application wizard					•			
Tools for developing location-based applications							•	•
Event simulation support							•	•
File-based short message service (SMS) and multi-media messaging service (MMS) sending and reception					•	•	•	•
Integration with Nokia Connectivity Framework					•	•	•	•
Bluetooth card protocols				BCSP and H4	BCSP and H4	BCSP and H4	BCSP and H4	BCSP and H4
Scalable UI support for resolution: 176 x 208-pixel screen						•	•	
Scalable UI support for resolution: 208 x 176-pixel screen							•	
Scalable UI support for resolution: 240 x 320-pixel screen (QVGA)						•	•	•
Scalable UI support for resolution: 352 x 416-pixel screen						•	•	
Scalable UI support for resolution: 416 x 352-pixel screen							•	
Scalable UI support for resolution: 320 x 240-pixel screen (QVGA)							•	•
Support for Carbide.c++			•	•	•	•	•	•
Support for CodeWarrior® Development Studio for Symbian OS v2.8 and v3.0	•	•	•	•	•	•		
Support for CodeWarrior® Development Studio for Symbian OS v3.1							•	•
Support for Microsoft Visual Studio .NET 2003 (requires Carbide.vs)	•	•	•	•	•	•	•	•
Support for Carbide.vs 2.0	•	•	•	•	•	•	•	•

Sumber: www. forum.nokia.com

2.6. *Integrated Development Environment*

IDE ***(Integrated Development Environment)*** adalah program komputer yang memiliki beberapa fasilitas yang diperlukan dalam pembangunan perangkat lunak. Tujuan dari IDE adalah untuk menyediakan semua utilitas yang diperlukan dalam membangun perangkat lunak [3].

Sebuah IDE, atau secara bebas dapat diterjemahkan sebagai Lingkungan Pengembangan Terpadu, setidaknya memiliki fasilitas:

- *Editor*, yaitu fasilitas untuk menuliskan kode sumber dari perangkat lunak.
- *Compiler*, yaitu fasilitas untuk mengecek sintaks dari kode sumber kemudian mengubah dalam bentuk binari yang sesuai dengan bahasa mesin.

- *Linker*, yaitu fasilitas untuk menyatukan data binari yang beberapa kode sumber yang dihasilkan *compiler* sehingga data-data binari tersebut menjadi satu kesatuan dan menjadi suatu program komputer yang siap dieksekusi.
- *Debuger*, yaitu fasilitas untuk mengetes jalannya program, untuk mencari kesalahan yang terdapat dalam program.

Sampai tahap tertentu IDE modern dapat membantu memberikan saran yang mempercepat penulisan. Pada saat penulisan kode, IDE juga dapat menunjukan bagian-bagian yang jelas mengandung kesalahan atau keraguan.

2.6.1.Sejarah

Sebagai bahan pembanding, bahasa pemrograman terdahulu disiapkan dengan cara yang berbeda, karena disiapkan melalui penyusunan *flowchart*, menggunakan formulir isian bahkan kartu berlubang.

IDE pada tahap awal memungkinkan perintah kode dituliskan dalam satu atau lebih file teks, lalu dikompilasi melalui perintah baris. Penyusunan tata letak relatif sulit karena hanya berupa perkiraan. Jika terdapat kesalahan pada kode, proses kompilasi harus dihentikan.

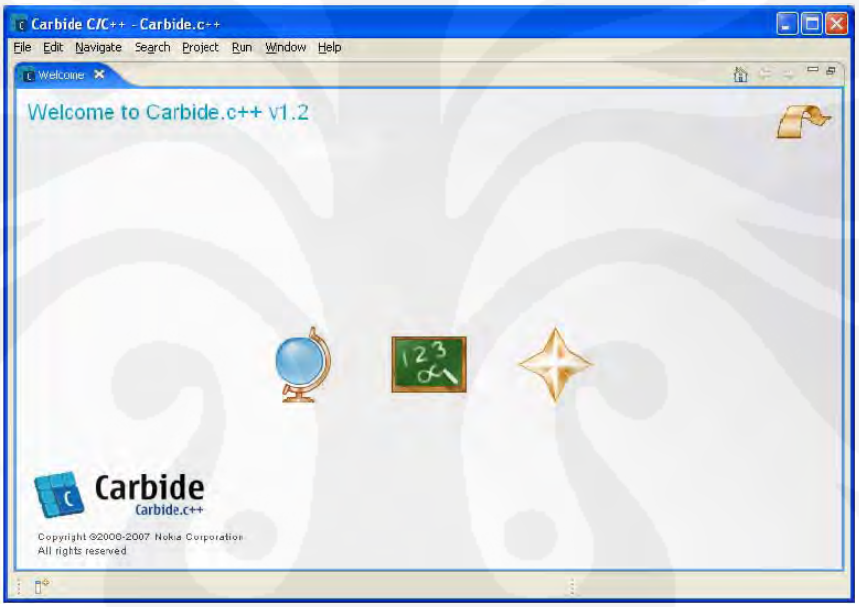
Ada beberapa pilihan IDE yang dapat digunakan untuk mengembangan aplikasi pada Sistem Operasi Symbian, yaitu :

- Nokia Carbide Development Tools for Symbian OS C++
- Nokia Carbide Development Tools for Java - Carbide.j
- Wirelexsoft VistaMax, ARM RealView Development Suite, SymbDev, Xcode Plugin for Symbian OS, VistaMax, dll

Pada pembahasan ini akan dipergunakan Nokia Carbide Development Tools for Symbian OS C++ (menggunakan bahasa pemrograman C++), dengan pertimbangan bahwa Carbide adalah IDE yang direkomendasikan oleh Nokia.

2.7. Carbide C++

Carbide.c + + adalah sebuah *tools* untuk pengembangan perangkat lunak C + + pada Symbian OS. Digunakan untuk mengembangkan perangkat lunak ponsel yang menggunakan sistem operasi symbian, serta aplikasi yang berjalan di ponsel mereka. Ini didasarkan pada Eclipse IDE platform dengan meningkatkan tambahan *plug-in* untuk mendukung pengembangan Symbian OS. Produk yang dikembangkan oleh Nokia sebagai bagian dari pengembangan keluarga perangkat Carbide , yang menggantikan *CodeWarrior* untuk Symbian OS sebagai dasar pengembangan lingkungan untuk Symbian OS.



Gambar2.7. Tampilan Carbide C++

2.7.1 Versi

Sebelum versi 2.0, Carbide.c + + ada beberapa edisi yang telah dikeluarkan:

- *Express – Tools* dasar hanya untuk pembangunan non-komersial. Berisi proyek manajemen, kode authoring, emulator & GCC-E dan emulator debug. Edisi Express saat ini ditawarkan secara bebas.
- *Developer* - Target di pengembangan piranti lunak *aftermarket*. Berisi fitur Express, sebuah *UI Designer* (untuk penciptaan pesat UI), dan aplikasi-level pada perangkat-debug untuk S60 dan UIQ ponsel.
- *Profesional* - Target pada produsen telepon Symbian OS, mitra mereka, dan aplikasi / vendor *middleware* yang bekerja berdasarkan permintaan

proyek. Pengembang berisi fitur, sistem level pada perangkat-debug, dan kinerja profil alat.

- OEM - Target awal pengembangan *Embedded access* seperti pembangunan driver, dasar port, dan *hardware* yang bergantung pada pengembangan aplikasi dan *middleware*. Berisi fitur Profesional, dan *stop - mode debugging* menggunakan *Lauterbach* dan *Sophia* di *sirkuit-emulators*.

Dari versi 2.0 Carbide C + + adalah bebas dan ditawarkan 3 edisi (Pengembang, Profesional, dan OEM).

Tabel 2.2 berikut ini memuat fitur – fitur yang tersedia dalam setiap edisi Carbide C++ :

Tabel 2.2 Fitur – fitur yang tersedia pada masing – masing edisi.

Key features	Carbide.c++ Developer Edition	Carbide.c++ Professional Edition	Carbide.c++ OEM Edition
New-project wizard	•	•	•
New-project templates	•	•	•
Project import and export	•	•	•
Integrated project builder	•	•	•
Online tutorial	•	•	•
Public SDK support	•	•	•
R&D SDK (CustKit) support	•	•	•
R&D devices support[1]		•	•
Standard build-target support (WINSC, GCCE)	•	•	•
Enhanced build-target support (ARMv5, ARMv5_ABIv2)	•	•	•
Fully featured emulator-based debugging	•	•	•
Application-level on-device debugging	•	•	•
System-level on-device debugging[1]		•	•
Symbian OS Data View	•	•	•
Crash Debugger console		•	•
ROM-based on-device debugging			•
Kernel debugging			•
Graphical UI designer for S60 and UIQ UIs	•	•	•
Performance Investigator[2]		•	•
Plug-in development support	•	•	•
Executables view	•	•	•
Global search	•	•	•
CodeScanner static-analysis tool		•	•
Knowledgebase Scanning (subset of CodeScanner)	•	•	•
Dependency Explorer tool		•	•

Sumber: www. forum.nokia.com

2.7.2 Menggunakan Carbide.C + +

Untuk melakukan pengembangan Symbian OS C + + diperlukan tidak hanya Carbide.C + +, tetapi juga OS Symbian SDK. SDK ini berisi emulator dan juga *library* dan *file header* yang diperlukan untuk pengembangan Symbian OS.

Setelah terinstal, Carbide.C + + menawarkan kesempatan untuk men-download seperti SDKs namun juga dapat men-downloadnya secara bebas – dari misalnya, Nokia atau UIQ.

2.7.3 Carbide.vs

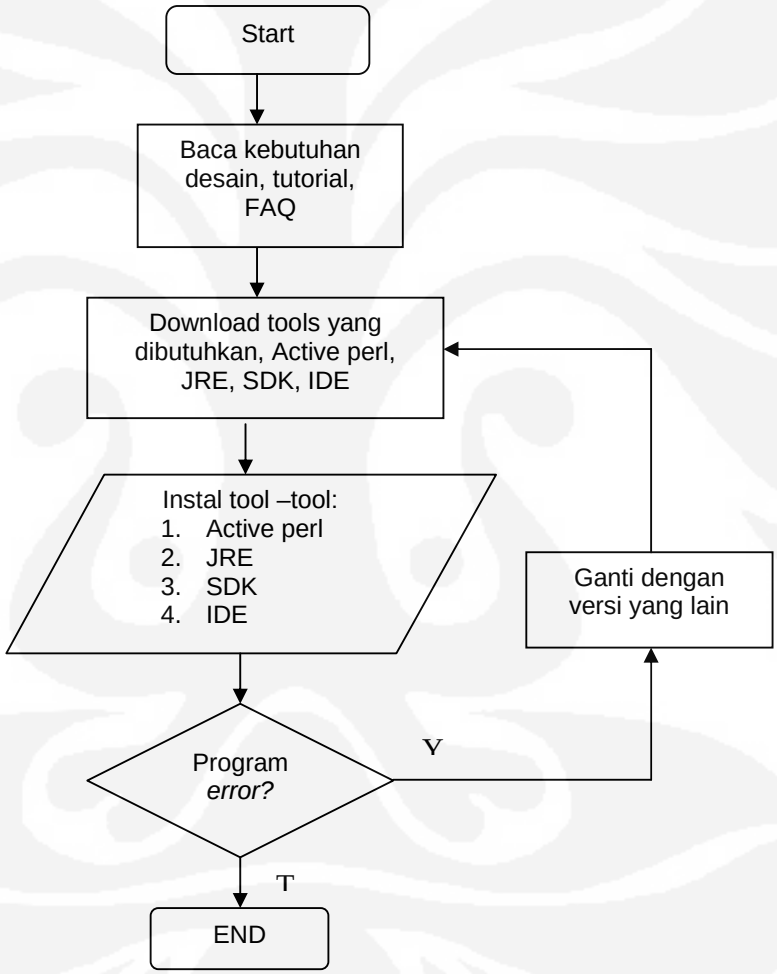
Carbide.vs 2,0 adalah *plugin* Visual Studio 2003 yang memungkinkan para pengembang untuk menggunakan Visual Studio. NET 2003 sebagai IDE untuk Symbian pembangunan. Carbide.vs 3,0 adalah *plugin* Visual Studio 2005 yang memungkinkan efisiensi Symbian OS C + + pengembangan aplikasi menggunakan Microsoft Visual Studio. NET 2005 IDE dan Symbian OS SDKs. Carbide.vs ditujukan pada para pengembang dengan keterampilan Visual Studio yang ingin membuat aplikasi C + + untuk platform Symbian OS.

Carbide.vs Menyediakan kemudahan dalam pembangunan Symbian OS C + + dengan *Wizards* dan fungsi otomatis yang terintegrasi dengan Visual Studio. Pengguna dapat memulai dengan sedikit konfigurasi manual. Carbide.vs juga berisi beberapa fungsi otomatis untuk pengembangan tugas khusus untuk Symbian OS.

BAB 3
PERANCANGAN DAN IMPLEMENTASI PERANGKAT LUNAK

3.1 Proses Perancangan Dan Implementasi

Pada bagian perancangan dan implementasi ini meliputi 3 hal utama yaitu tahap instalasi program, tahap pembuatan program, dan tahap penerapan program pada perangkat. Sebagaimana terlihat pada diagram alir pada Gambar 3.1 dibawah ini :



Gambar 3.1 Diagram alir proses instalasi.

Untuk memulai pemrograman berbasis Symbian tentunya dibutuhkan sarana untuk pengembangan. Sebagai objek pengembangan pada perangkat lunak dibutuhkan sejumlah *tools*. Pada tahap instalasi ini sangat penting untuk diperhatikan bagaimana hubungan *tools* yang satu dengan yang lain. Dimulai dengan membaca panduan, kebutuhan desain meliputi spesifikasi teknis perangkat, SDK apa saja yang didukung.

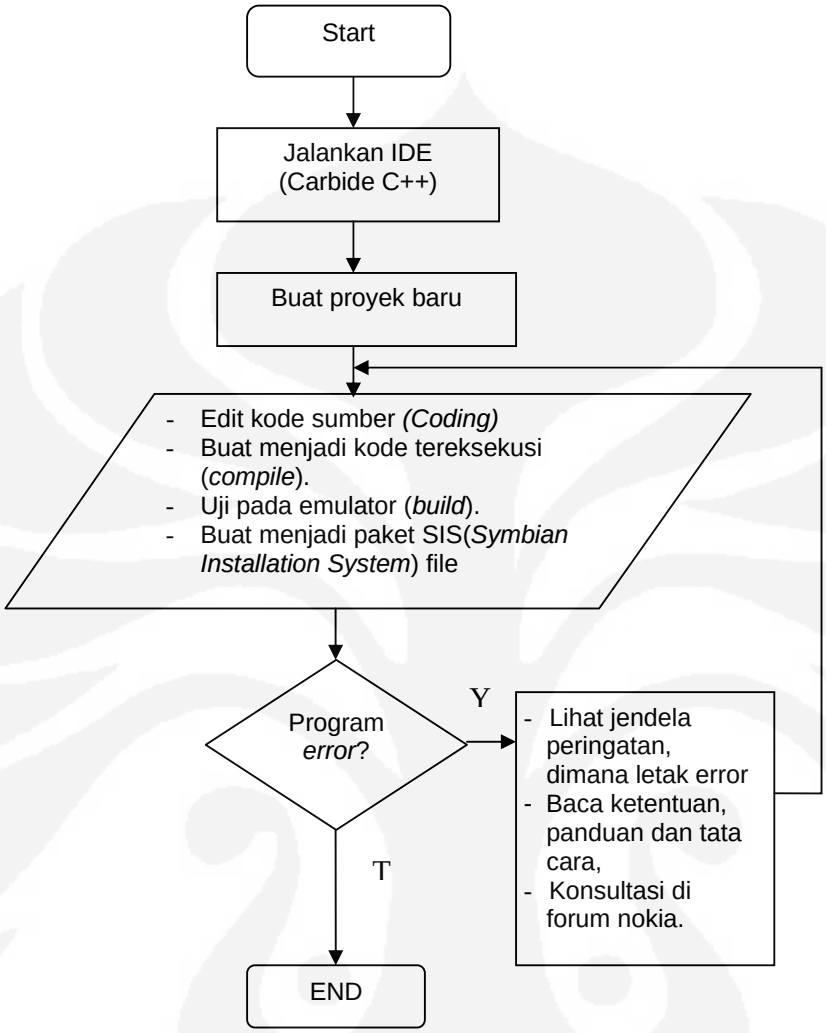
Sesuai dengan spesifikasi teknis SDK yang didukung oleh *handset* maka pada pengembangan aplikasi ini menggunakan SDK 3rd Ed FP 1. Sedangkan untuk pembuatan program atau IDE digunakan Carbide C++, selain itu tidak boleh juga diabaikan keberadaan *3rd party software* meliputi *Activeperl* dan *Java Runtime Environment (JRE)*. Kesemua *tools* pengembangan tersebut bisa didapatkan dengan tanpa membayar lisensi melalui situsnya masing – masing. Setelah semuanya didapatkan diinstall dengan urutan :

1. *Activeperl*
2. JRE
3. SDK
4. IDE

Permasalahan yang timbul dalam proses instalasi misalnya adalah SDK mengalami error, ini bisa disebabkan karena versi dari *perl*, disarankan adalah versi 5.6.1 ke atas. Jika semua sudah berjalan baik selanjutnya dimulai tahap pembuatan proyek.

Tahap pembuatan proyek ini adalah tahap yang paling penting karena di tahap inilah suatu aplikasi dibangun kemudian dibuat menjadi aplikasi yang siap untuk diluncurkan. Setelah semua tools yang dibutuhkan berhasil dijalankan dimulai pembuatan proyek melalui IDE yaitu carbide C++, proses – proses yang dijalankan antara lain :

- Proses edit kode sumber dan file - file *header*
- Membangun proyek untuk membuat menjadi kode tereksekusi.
- Menguji kode tereksekusi ke dalam *emulator*.
- Membangun dan mengepak aplikasi untuk diinstal pada perangkat.



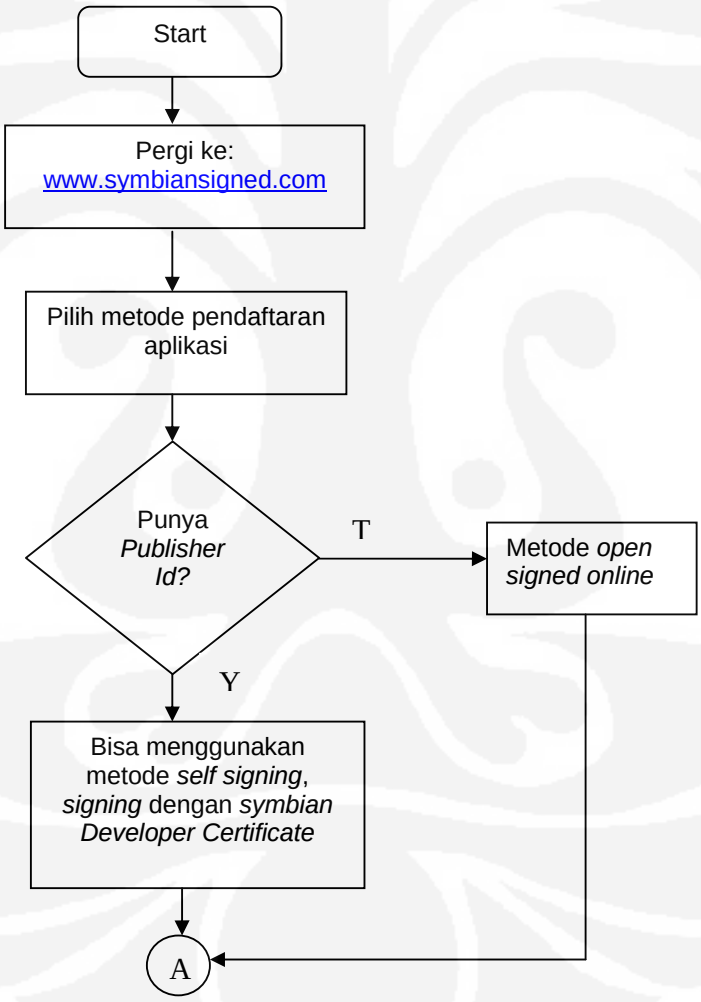
Gambar 3.2 Proses Pembuatan proyek

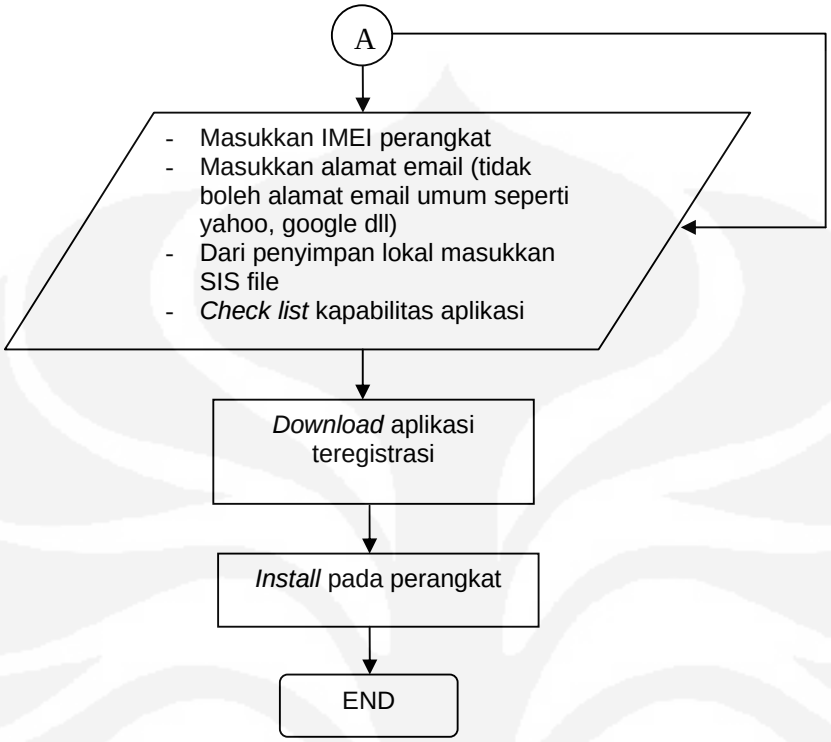
Proses edit kode atau *coding* meliputi penerapan fungsi – fungsi dan *library* symbian C++ yang dibutuhkan dalam rangka mencapai tujuan aplikasi yang diinginkan. Setelah proses *coding* selesai dilanjutkan dengan proses *compile* yaitu untuk melihat apakah terjadi suatu kesalahan pada proses coding. Jika terjadi kesalahan akan muncul pesan error yang menunjukkan dimana kesalahan terjadi. Error yang muncul dapat diperbaiki dengan mengacu kepada petunjuk penggunaan, pertolongan. Untuk pemrograman symbian segala sesuatunya sudah tersedia lengkap di forum nokia.

Apabila kesalahan sudah diperbaiki dan tidak ada lagi *error* untuk lebih baiknya aplikasi terlebih dahulu diluncurkan pada emulator sebagai simulasi. Jika

sudah sesuai dengan keinginan dan kebutuhan, maka aplikasi harus dibuat menjadi paket yang siap untuk dijalankan pada perangkat dengan mengkonversinya menjadi SIS *(Symbian Installation System)* file.

Tahap terakhir adalah instalasi aplikasi pada perangkat. Tetapi sebelum bisa dipasang pada perangkat terlebih dahulu aplikasi harus didaftarkan untuk memperoleh sertifikasi. Untuk penggunaan personal atau diperuntukkan hanya sebagai uji coba bisa menggunakan metode *open signed online*. Metode ini adalah yang paling mudah dikarenakan tidak membutuhkan *publisher ID* dan *symbiansigned account*.





Gambar 3.3. Proses registrasi aplikasi dan penerapan pada perangkat

Jika aplikasi ingin digunakan untuk kebutuhan masal bisa memilih metode *self signing*, *signing* dengan *symbian Developer Certificate.* Tentunya dikuti dengan adanya *publisher ID* yang mana didapatkan dengan membeli pada situs – situs yang berwenang.

Apabila metode *open signed online* yang digunakan, proses yang dilakukan adalah memasukkan IMEI perangkat, kemudian SIS file, alamat email (tidak boleh menggunakan alamat email umum seperti yahoo, google dll) untuk konfirmasi sekaligus alamat untuk mendownload aplikasi dan kapabilitas aplikasi. Setelah itu aplikasi siap dipasang pada perangkat.

3.2 Instalasi Program

Pada tahap instalasi Symbian C++, ada beberapa hal yang harus diperhatikan misalnya ketersediaan perangkat – perangkat lunak pembantu atau disebut juga *3rd party software* selain bagian utama yaitu SDK dan IDE. Tahapannya adalah sebagai berikut :

3.2.1 Instal *active perl*

Banyak *tools* untuk membangun symbian adalah *perl script* sehingga instalasi *perl* diperlukan. *ActivePerl* dari *ActiveState* adalah salah satu *perl* yang lengkap dan siap diinstal sekaligus bebas untuk diambil dari www.activeperl.com. Versi yang disarankan adalah 5.6.1 ke atas.

3.2.2 Instal *Java Run – time*

Java Runtime Environment (JRE) adalah sebuah *platform* perangkat lunak dari Sun Microsystem yang terdiri dari *Java Virtual Machine, Java Libraries*, dan beberapa komponen yang lain. JRE mengijinkan sebuah komputer untuk menjalankan aplikasi berbasis Java.

JRE diperlukan oleh beberapa tools Symbian OS dan S60, sebagai Contoh :

- ECMT, sekumpulan tools dari S60 SDK, contohnya untuk *remote logging*, konfigurasi emulator dll.
- SISAR (Menjadi satu paket dengan Symbian OS 6.1 SDKs ke atas)
- *AIF Builder* (Menjadi satu paket dengan Symbian OS 6.1 SDKs ke atas).

3.2.3 Instal IDE (Carbide .vs 2.0)

Pada pembahasan ini akan dipergunakan Nokia Carbide Development Tools for Symbian OS C++ (menggunakan bahasa pemrograman C++), dengan pertimbangan bahwa Carbide adalah IDE yang direkomendasikan oleh Nokia.

Carbide.c + + adalah sebuah *tools* untuk pengembangan perangkat lunak C + + pada Symbian OS. Digunakan untuk mengembangkan perangkat lunak ponsel yang menggunakan sistem operasi Symbian, serta aplikasi yang berjalan di ponsel mereka. Perangkat lunak ini bisa di *download* secara bebas dari www.forum.nokia.com.

3.2.4 Instal SDK (S60 3rd FP1)

Symbian memiliki beberapa jenis SDK untuk berbagai *platform User Interface* (UI), yaitu : UIQ, S60, Nokia Series 80, dan Nokia 7710. Pembahasan kali ini akan menitik beratkan kepada pengembangan aplikasi berbasis S60, dengan pertimbangan banyaknya perangkat *SmartPhone* yang menggunakan *platform* ini.

Pada perancangan aplikasi ini digunakan S60 3rd FP1 sesuai dengan versi SDK yang didukung oleh *handphone* (Nokia 6120 Classic) yang digunakan sebagai uji coba aplikasi. Perangkat lunak ini juga bisa di *download* gratis di www.forum.nokia.com.

3.2.5 Instal Nokia *PC Suite*

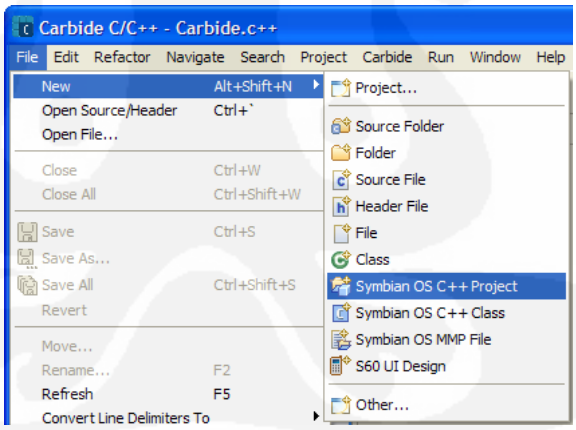
Aplikasi ini bisa didapatkan saat pembelian telepon genggam dengan merk Nokia.

3.3 Mengembangkan Aplikasi Dengan Carbide C++

Pada bagian ini akan ditunjukkan bagaimana suatu aplikasi dapat dibangun menggunakan *wizard* (kotak dialog) ataupun *fitur* yang tersedia pada Carbide C++.

3.3.1 Memulai Carbide C++

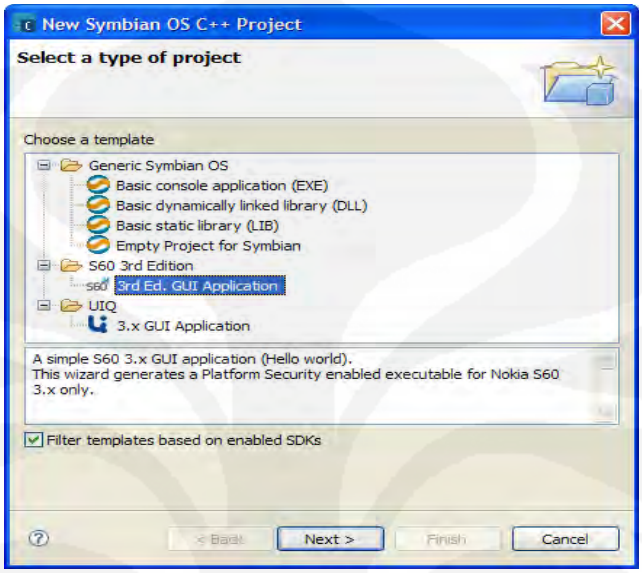
Carbide C++ menyediakan sejumlah *wizard* yang dapat membantu untuk membuat proyek Symbian C++. Untuk memulai dipilih **File > New** dari menu. Pilihan menu diperlihatkan pada Gambar 3.4 di bawah ini :



Gambar 3.4. Ada beberapa pilihan untuk membangun proyek baru pada Carbide C++.

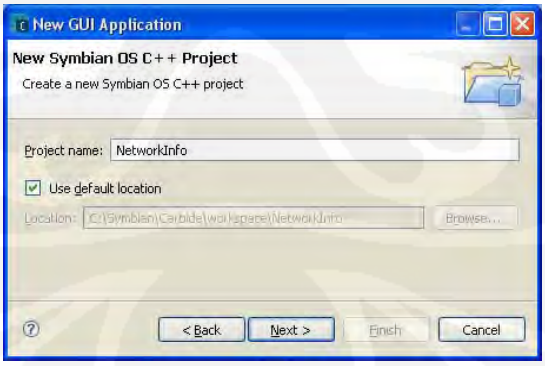
Untuk memulai membuat sebuah proyek menggunakan salah satu wizard yang tersedia pada Carbide C++, dipilih **Symbian OS C++ Project.** Maka Symbian OS C++ *project wizard* akan ditampilkan sebagaimana diperlihatkan pada gambar 3.4. Sejumlah pilihan disediakan untuk generik Symbian OS dan

aplikasi berbasis GUI. Pada Gambar 3.5 *wizard* yang dipilih adalah SDK yang sudah terinstal pada computer .



Gambar 3.5. Setelah memilih type proyek, Symbian OS C++ *wizard* yang baru digunakan untuk memilih type dari aplikasi yang akan dikembangkan.

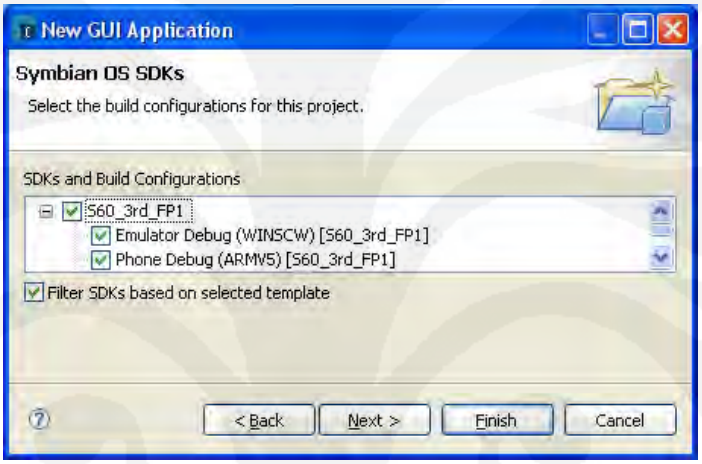
Sekarang *wizard* untuk aplikasi GUI *3rd Edition* yang baru telah terbuka. Langkah pertama adalah memberi nama untuk proyek yang baru dan menentukan dimana lokasi proyek akan disimpan. Diperlihatkan pada Gambar 3.6 berikut :



Gambar 3.6. *Wizard* 3rd Ed yang baru akan menanyakan nama proyek dan lokasi proyek akan disimpan.

Selanjutnya *wizard* meminta untuk memilih aplikasi SDK yang akan dibangun. Kotak dialog sebagaimana ditunjukkan pada Gambar 3.7 menampilkan

semua SDK yang terinstal pada computer, yang mana walaupun tersembunyi secara inisial terdaftar sebagai konfigurasi yang tersedia. Dalam kasus ini S60_3rd_FP1 dipilih karena sesuai dengan SDK yang didukung oleh telepon genggam yang nantinya akan diuji cobakan. Secara bersamaan pembangunan untuk *emulator debug* dan *phone release* terpilih secara otomatis. Ini akan memperbolehkan aplikasi untuk dibangun sebagai uji coba di S60 3rd Ed, *Feature Pack 1 emulator* dan dijalankan pada perangkat telepon genggam.

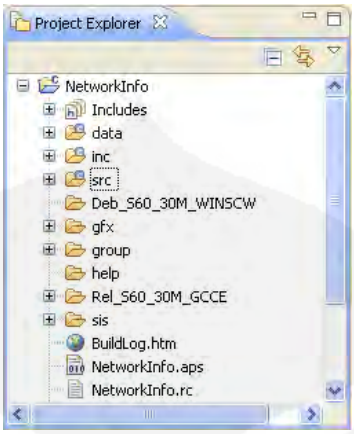


Gambar 3.7. Langkah kedua untuk aplikasi GUI 3rd Ed yang baru, *Wizard* meminta SDK yang tepat dan konfigurasi pembangunan untuk dipilih.

Pada point ini dimungkinkan untuk memilih tombol *finish* untuk membuat proyek. Untuk membuat proyek yang lebih bagus *wizard* mempunyai dua langkah lebih lanjut. Langkah ini memperbolehkan aplikasi UID untuk menjadi spesifik dan lokasi file untuk komponen proyek yang bervariasi. Untuk proyek ini kedua langkah tersebut bisa dilewatkan.

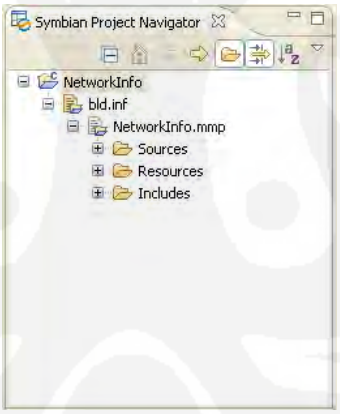
Ketika tombol *finish* sudah dipilih , Carbide C++ mengkonstruksi proyek, membuat seluruh komponen proyek yang diperlukan. Sekali dibuat, file – file ini dilihat dalam dua tampilan. *C/C++ project* dan *Symbian Navigator Project*.

Tampilan C/C++ ditunjukkan pada Gambar 3.8, terdiri dari *folder structure* yang menyimpan komponen proyek Network Info, seperti kode – kode dan *Symbian Installation System (SIS).*



Gambar 3.8. Tampilan C/C++ terdiri dari berbagai macam file - file aplikasi proyek.

Untuk tampilan *Symbian Project Navigator* diperlihatkan pada Gambar 3.9 , tampilan ini menyediakan akses untuk definisi dari file – file proyek, informasi file yang dibangun, makmake file dan tambahan *Makefiles*, seperti icon untuk aplikasi Makefile `Icons_scalable_dc.mk`.



Gambar 3.9. Tampilan untuk *Symbian Project Navigator* terdiri dari berbagai macam file – file definisi proyek.

3.3.2 Bekerja Dengan Proyek

Pada bagian ini sebuah aplikasi yang nantinya akan dikembangkan telah dibangun oleh wizard pada carbide C++. Aplikasi ini sekarang dapat dimodifikasi untuk dibuat suatu aplikasi yang diinginkan. Ada sejumlah langkah – langkah yang harus dilakukan, yang mana termasuk :

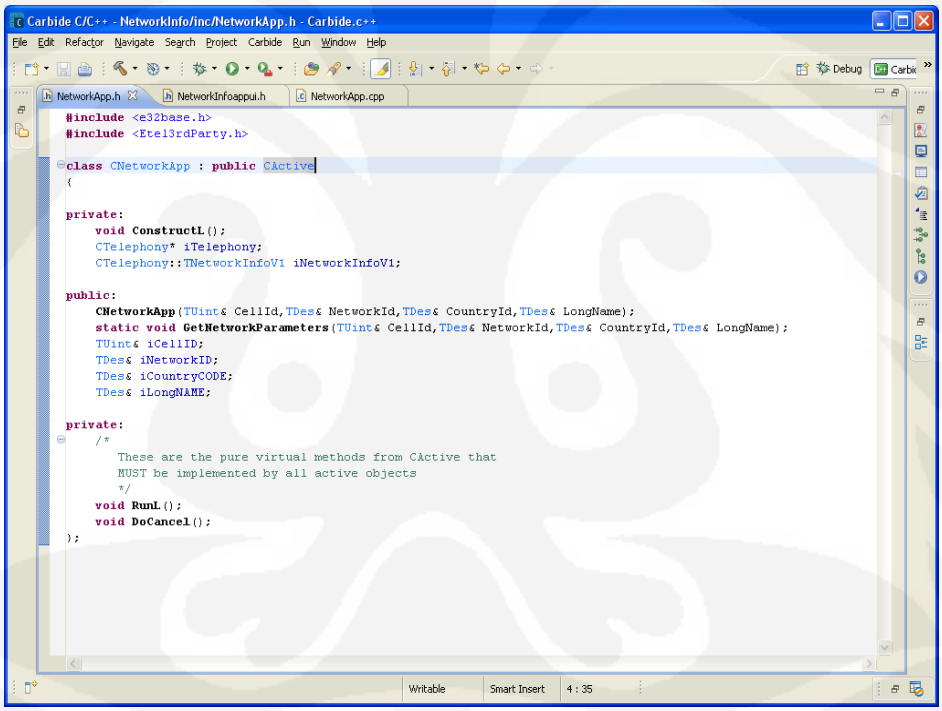
- Proses edit kode sumber dan file - file *header*
- Membangun proyek untuk membuat menjadi kode tereksekusi.

- Menguji kode tereksekusi ke dalam *emulator*.
- Membangun dan mengepak aplikasi untuk diinstal pada perangkat.

3.3.2.1 Proses Edit Kode Sumber Dan File – File Header

Untuk menngimplementasikan secara fungsional aplikasi yang diinginkan, akan sangat dibutuhkan untuk merubah kode sumber dan file – file *header*. File – file sumber proyek disimpan di dalam folder `/src` dan file – file header termasuk `*.h` dan `*.hrh` di dalam `/inc` folder. Pada file header utama yaitu NetworkApp.h agar bisa membaca network parameter yang diakses saat ini, yang paling utama adalah penambahan library `etel3rdparty.h` dan *class* `CTelephony,`

dan isi program nya terlihat pada Gambar 3.10 berikut ini :



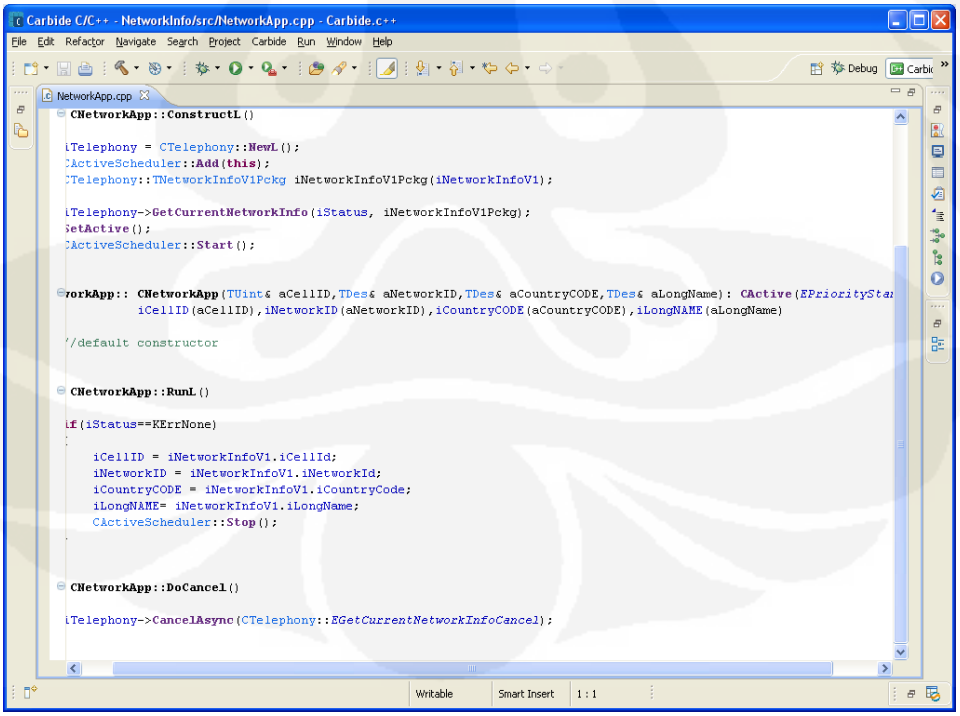
Gambar 3.10. Isi dari file Header NetworkApp.h

Untuk lebih jelasnya *script* dari NetworkApp.h adalah sebagai berikut :

```
#include <e32base.h>
#include <Etel3rdParty.h>

class CNetworkApp : public CActive
{

private:
```

```
        void ConstructL();
        CTelephony* iTelephony;
        CTelephony::TNetworkInfoV1 iNetworkInfoV1;
        CTelephony::TNetworkInfoV1Pckg iNetworkInfoV1Pckg;

public:
        CNetworkApp(
                TUint& CellId, TDes& NetworkId, TDes& CountryId, TDes&
LongName);
        static void GetNetworkParameters(
                TUint& CellId, TDes& NetworkId, TDes& CountryId, TDes&
LongName);
        ~CNetworkApp();
        TUint& iCellID;
        TDes&  iNetworkID;
        TDes&  iCountryCODE;
        TDes&  iLongNAME;

private:
        /*
           These are the pure virtual methods from CActive that
           MUST be implemented by all active objects
         */
        void RunL();
        void DoCancel();
};
```

Selanjutnya ditambahkan pada kode sumber yaitu NetworkApp.cpp, file header NetworkApp.h dengan tetap menggunakan fungsi utama yaitu `CTelephony`, dan isi dari NetworkApp.cpp adalah :



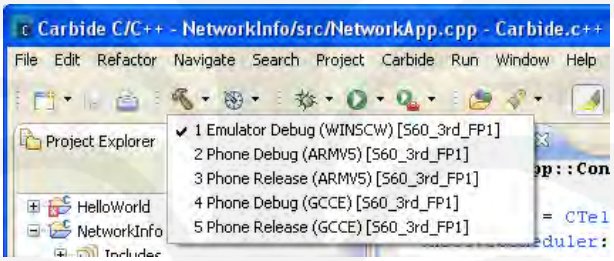
Gambar 3.11. Isi dari file NetworkApp.cpp

Untuk lebih jelasnya isi dari file NetworkApp.cpp diperlihatkan sebagai berikut :

```
#include "NetworkApp.h"

void CNetworkApp::GetNetworkParameters(
        TUint& aCellID, TDes& aNetworkID, TDes& aCountryCODE,
        TDes& aLongName)
{
        CNetworkApp* self=
                new(ELeave) CNetworkApp(
                        aCellID, aNetworkID, aCountryCODE, aLongName);
        CleanupStack::PushL(self);
        self->ConstructL();
        CleanupStack::PopAndDestroy(self);
}

void CNetworkApp::ConstructL()
{
        iTelephony = CTelephony::NewL();
        CActiveScheduler::Add(this);

        iTelephony->GetCurrentNetworkInfo(iStatus, iNetworkInfoV1Pckg);
        SetActive();
        CActiveScheduler::Start();
}

CNetworkApp::CNetworkApp(
        TUint& aCellID,TDes& aNetworkID,TDes& aCountryCODE,TDes&
aLongName):
                CActive(EPriorityStandard),
iNetworkInfoV1Pckg(iNetworkInfoV1),
                iCellID(aCellID), iNetworkID(aNetworkID),
iCountryCODE(aCountryCODE),
                iLongNAME(aLongName)
{
        //default constructor
}

void CNetworkApp::RunL()
{
        if(iStatus==KErrNone)
        {
                iCellID = iNetworkInfoV1.iCellId;
                iNetworkID = iNetworkInfoV1.iNetworkId;
                iCountryCODE = iNetworkInfoV1.iCountryCode;
                iLongNAME= iNetworkInfoV1.iLongName;
                CActiveScheduler::Stop();
        }
}

void CNetworkApp::DoCancel()
{
        iTelephony-
>CancelAsync(CTelephony::EGetCurrentNetworkInfoCancel);
}

CNetworkApp::~CNetworkApp()
{
        if(iTelephony)
        {
                Cancel();
                delete iTelephony;
                iTelephony = NULL;
        }
}
```

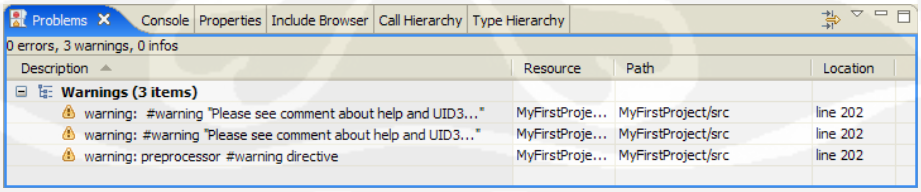
3.3.2.2 Membangun proyek untuk membuat menjadi kode tereksekusi.

Setelah proses edit kode dan file header sudah dilakukan maka langkah selanjutnya adalah memutuskan target untuk aplikasi ini akan diterapkan. Untuk membuat proyek pada proses penerapan ini dari menu Carbide C++ dipilih **Project** > **Build All Targets** membolehkan semua target untuk penerapan aplikasi. Proses ini juga bisa dilakukan di toolbar sebagaimana ditunjukkan pada Gambar 3.12 dibawah ini :



Gambar 3.12 Proyek dapat dibuat untuk sebuah *single target* dari tombol *build* di *toolbar*.

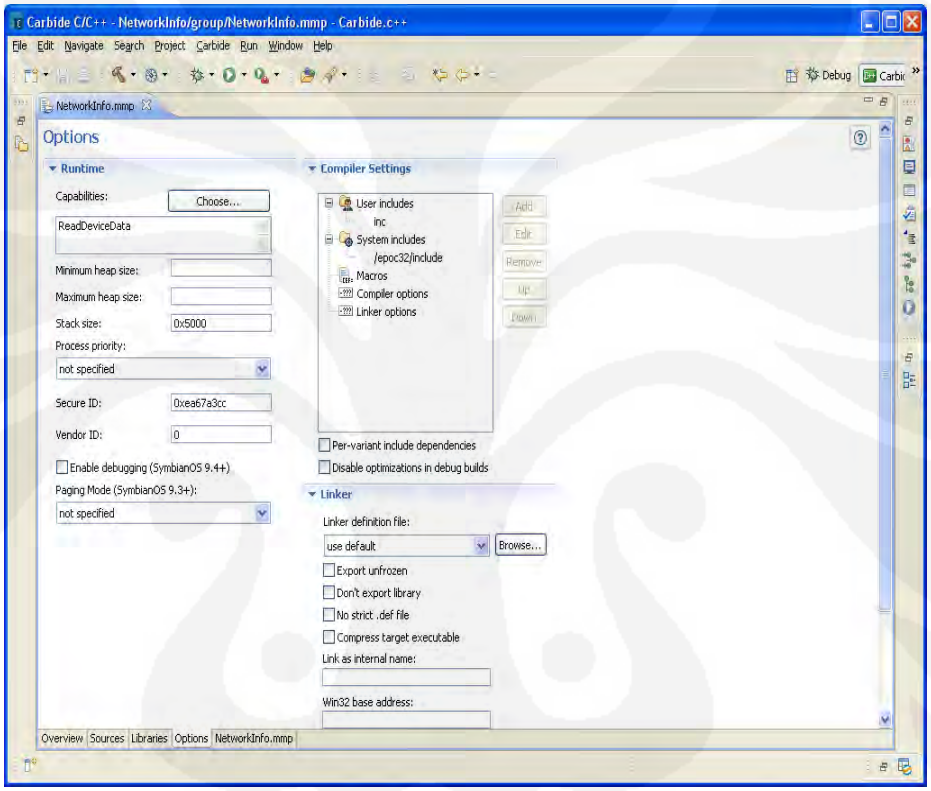
Ketika aplikasi dibuat, tampilan pada konsol menyediakan detail penuh dari proses pembuatan. Sejumlah masalah yang teridentifikasi ditunjukkan pada jendela ***Problems*** sebagaimana ditunjukkan pada gambar. Dua tampilan ini adalah point pertama sebagai referensi untuk memperbaiki kesalahan. Dengan memilih pesan pada jendela *Problems* membuka kode yang relevan pada jalur dimana problem terjadi.



Gambar 3.13. Contoh Tampilan problem memberikan *feedback* terhadap masalah selama proses pembuatan.

Proses pembuatan diatur melalui *Project's makmake file*, yang mana dapat diakses dari *symbian project navigator*. Pembuatan file editor informasi dan MMP editor membolehkan pengembang untuk mengendalikan semua aspek dari pembangunan proyek, termasuk fitur – fitur seperti C++ *linker* dan *compiler*,

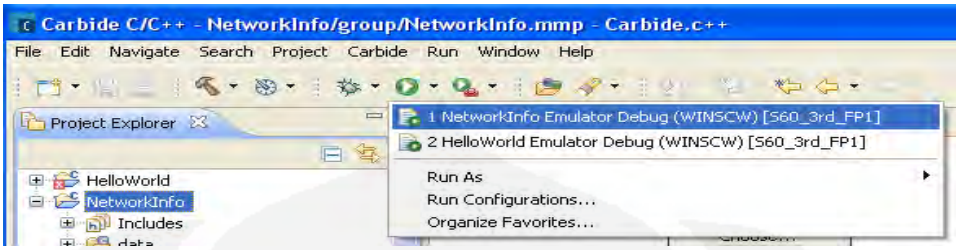
sebagaimana ditunjukkan pada Gambar 3.14. Dikarenakan Carbide C++ menggunakan standar pembuatan informasi dan makmake file untuk mengendalikan proses, ini memungkinkan untuk mengganti antara membuat proyek pada Carbide C++ dan di *Command Line*. Ini akan menjadi berguna dimana otomatisasi pembuatan proses digunakan sebagai bagian dari pengembangan proses.



Gambar 3.14 Pembuatan proyek dikontrol melalui *makmake file*.

3.3.2.3 Menguji Proyek Pada Emulator

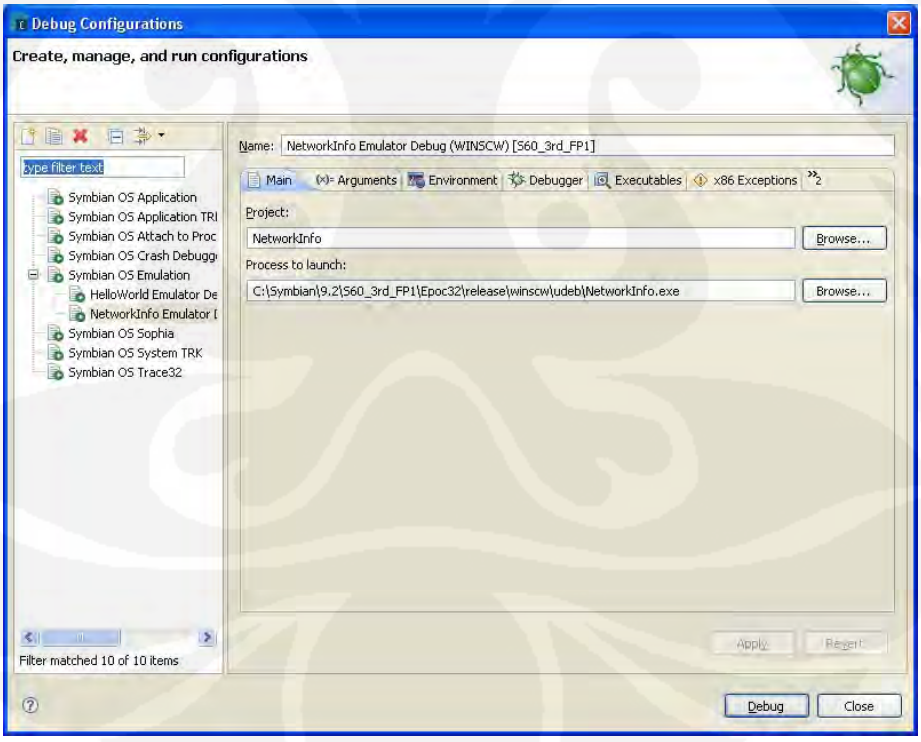
Setelah aplikasi telah berhasil dibuat, langkah selanjutnya adalah mengujinya pada S60 emulator. Aplikasi dapat dijalankan dengan segera menggunakan tombol run pada toolbar sebagaimana ditunjukkan pada Gambar 3.15, atau dari menu *project's context*.



Gambar 3.15. Pembuatan proyek dari emulator dapat dijalankan dari *toolbar*.

Setelah menggunakan metode *run* sederhana dan sekali emulator telah terbuka, aplikasi perlu untuk dilokasikan pada folder dimana aplikasi terinstal dan dimulai.

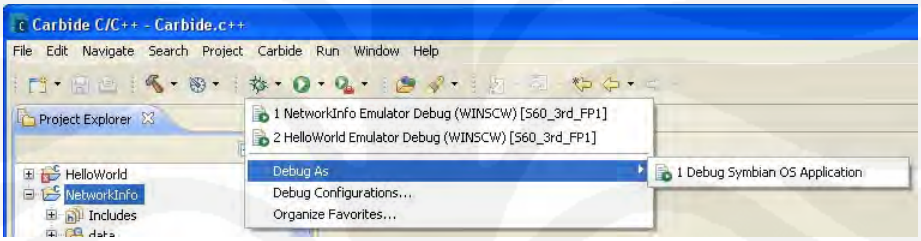
Cara menjalankan proyek dikontrol oleh *launch configuration*, sejumlah parameter – parameter (seperti nama proyek yang digunakan, lokasi emulator dan variabel – variabel lingkungan) yang mendefinisikan bagaimana proyek diluncurkan di emulator. Gambar 3.16. menunjukkan konfigurasi awal dibuat ketika tombol *run* pada *toolbar* ditekan.



Gambar 3.16. Menjalankan sebuah aplikasi memerlukan kreasi dari *launch configuration.*

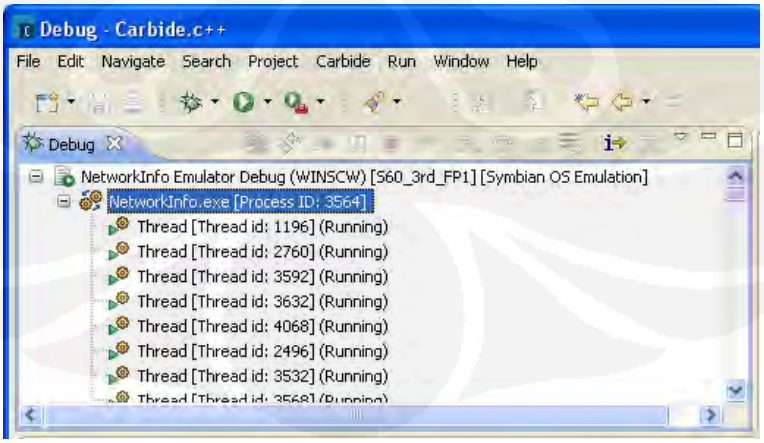
Selama peluncuran awal konfigurasi cukup memuaskan pada keseluruhan kondisi, konfigurasi sesuai keinginan dapat menjadi berguna. Sebuah peluncuran konfigurasi dapat diciptakan dengan memilih **Run As > Run** dari menu *project's context*.

Jika masalah – masalah ditemukan selama pengujian di emulator, langkah selanjutnya adalah menjalankan *debug* untuk aplikasi. Sejalan dengan menjalankan aplikasi, sesi *debug* dikontrol menggunakan *launch configuration*, tetapi *debug* dasar dapat dimulai dengan menekan tombol *debug* pada *toolbar*, sebagaimana diperlihatkan pada Gambar 3.17.



Gambar 3.17. Sesi debug dapat dimulai secara cepat dari *toolbar*.

Ketika sesi debug dimulai, Carbide C++ digantikan konfigurasi *debug*, menampilkan jendela – jendela yang relevan untuk proses debug dan menyediakan akses untuk *range* yang lebar dari fitur dan *tools debug*. Ini termasuk langkah bijaksana untuk eksekusi, *breakpoint* dan pengawasan nilai variabel.



Gambar 3.18. Proses Debug

Setelah proses *debug* berhasil dengan tidak menghasilkan suatu kesalahan, maka akan muncul emulator yang ditunjukkan pada gambar 3.19 dan merepresentasikan sebuah handset yang berfungsi sebagai simulasi.

Gambar 3.19. Tampilan Emulator

Untuk menjalankan aplikasi pada emulator, yaitu dengan masuk pada menu utama kemudian dipilih **Installed** > **NetworkInfo.**

Gambar 3.20. Aplikasi *NetworkInfo* yang sudah terinstal pada emulator.

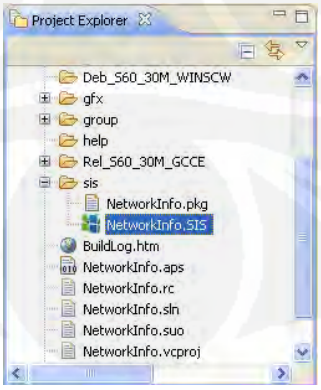
Selanjutnya adalah menguji apakah aplikasi bisa berjalan atau tidak pada emulator. Hasil pengujian diperlihatkan pada Gambar 3.21 . Karena emulator

bukanlah *handset* yang sesungguhnya dan tidak terhubung dengan jaringan aktif maka tidak ada nilai parameter *network* yang ditunjukkan atau bernilai nol.

Gambar 3.21. Hasil uji coba pada emulator.

3.3.2.4 Membuat Dan Menluncurkan Aplikasi Untuk Perangkat.

Setelah aplikasi telah di *debug* dan sukses diuji pada emulator, langkah selanjutnya adalah membuat dan menjalankan aplikasi pada perangkat. Senuah proyek dibuat dengan sebuah Carbide C++ *wizard* terdiri dari semua informasi yang dibutuhkan untuk membuat dan mengepak aplikasi untuk sebuah perangkat. Bagaimanapun, jika tambahan kode sumber telah ditambahkan, paket file yang mana menunjukkan konten dari SIS file dan digunakan untuk menginstal aplikasi pada perangkat, mungkin perlu untuk diperbaharui.



Gambar 3. 22. SIS file yang harus didaftarkan.

Ada aplikasi yang ditulis untuk symbian OS v9 atau diatas harus didaftarkan sebelum dapat diinstal pada perangkat. Ada beberapa bentuk pendaftaran : *self signing*, *signing* dengan *symbian Developer Certificate* dan *Symbian Signed*.

Pada aplikasi ini pendaftaran menggunakan *symbian signed* dengan metode *open signed online*, pendaftaran aplikasi dengan metode ini mempermudah pengembang untuk pengiriman terbatas ke perangkat, salah satunya sebagai pengujian ataupun untuk penggunaan personal. *Open signed application* didaftarkan dibawah *Developer Certificate*, dan pengiriman aplikasi dibatasi oleh IMEI dari perangakat.

Menggunakan pilihan *open signed online* tidak memerlukan *Publisher ID*. Sertifikasi pengembang dilakukan *online* melalui portal, diwakili pengembang, cepat, pilihan bebas untuk sekali pendaftaran aplikasi untuk penggunaan perangkat tunggal dibatasi IMEI. Dalam banyak kondisi *Open Signed* mendekati kebutuhan *freeware*, *open source* dan penggunaan pengembangan personal. Ini juga berguna untuk pengembang yang bekerja pada pengembangan *platform* yang tidak didukung *host* (sebagai contoh Linux atau Mac OS X) yang mempunyai masalah menjalankan *tools* yang diperlukan untuk pilihan yang lain.

Fitur utama dari metode pendaftaran ini adalah :

- Proses *online*.
- Tidak memerlukan *publisher ID*.
- Tidak membutuhkan akun *Symbian Signed*.
- Tidak membutuhkan *tool download*, independen.
- Aplikasi dibatasi untuk satu perangkat dan IMEI.
- Tidak ada biaya untuk pengembang.
- *Valid* untuk 36 bulan dari tanggal pendaftaran.

Langkah – langkah pendaftaran aplikasi menggunakan *Open Signed – Online* :

1. Pergi menuju Symbian Signed website dan mengakses layanan.

   Informasi yang disediakan akan di *encode* ke dalam *developer certificate* yang mana portal akan mengenerate dan menggunakan untuk mendaftarkan aplikasi.

   - Pergi ke www.symbiansigned.com

- Dari halaman selamat datang, ditekan *open signed online link* ke *online submission page*.
- Dimasukkan informasi yang diminta ke dalam online form, termasuk IMEI perangkat dan alamat email (tidak boleh alamat email umum, seperti yahoo, google).
- Dipilih kapabilitas yang diperlukan oleh aplikasi.
- Dari tempat penyimpanan lokal di computer, masukkan file SIS dari aplikasi.
- Masukkan kode keamanan.
- Pilih untuk melihat perjanjian hukum.
- Pilih untuk menerima setelah membaca perjanjian hukum.

Gambar 3.23. Pendaftaran melalui *open signed*, dengan memasukkan IMEI, alamat email dan SIS file.

2. Mengkonfirmasi alamat email.

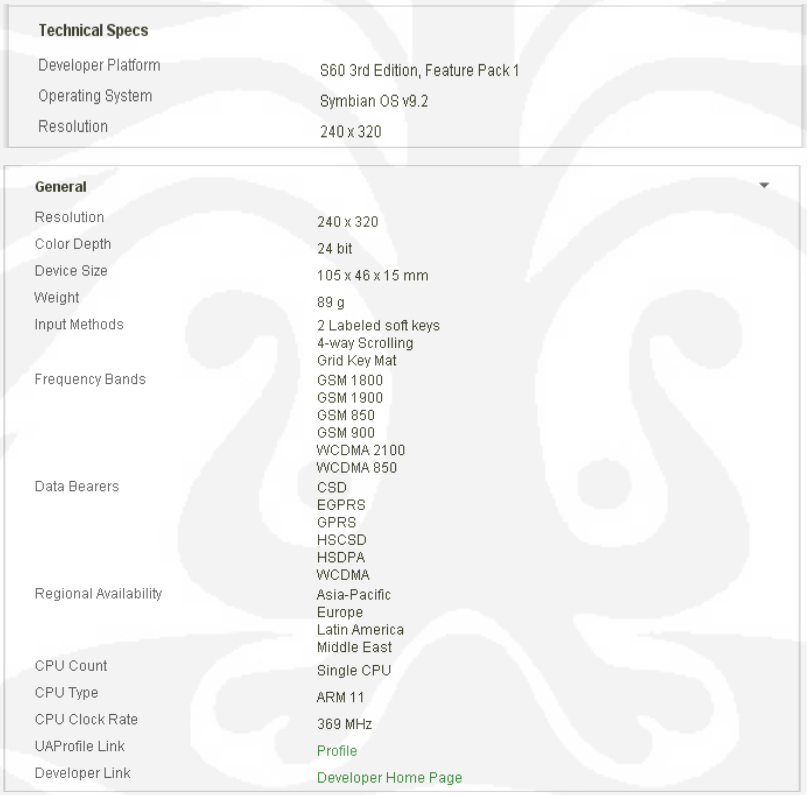
   Sebuah email akan dikirim ke alamat email yang dimasukkan untuk konfirmasi *link*. Selanjutnya adalah menuju alamat *link* konfirmasi tersebut.

3. Akses akun email untuk mengambil aplikasi yang telah terdaftar.

Sebuah email akan dikirim lagi terdiri dari sebuah alamat untuk SIS file yang telah didaftarkan. Langkah selanjutnya adalah pergi ke alamat tersebut dan men *download* aplikasinya.

3.4 Instalasi Pada Perangkat

Untuk perangkat yang akan diujicobakan, dipilih perangkat dengan Merk Nokia bertipe 6120 classic. Sebenarnya semua perangkat Nokia dengan tipe apapun bisa menggunakan aplikasi ini asalkan sudah didukung S60 3rd Ed FP1. Spesifikasi teknis dari perangkat ini adalah :



Technical Specs

Developer Platform	S60 3rd Edition, Feature Pack 1
Operating System	Symbian OS v9.2
Resolution	240 x 320

General

Resolution	240 x 320
Color Depth	24 bit
Device Size	105 x 46 x 15 mm
Weight	89 g
Input Methods	2 Labeled soft keys 4-way Scrolling Grid Key Mat
Frequency Bands	GSM 1800 GSM 1900 GSM 850 GSM 900 WCDMA 2100 WCDMA 850
Data Bearers	CSD EGPRS GPRS HSCSD HSDPA WCDMA
Regional Availability	Asia-Pacific Europe Latin America Middle East
CPU Count	Single CPU
CPU Type	ARM 11
CPU Clock Rate	369 MHz
UAProfile Link	Profile
Developer Link	Developer Home Page

Gambar 3.24. Spesifikasi teknis dari Nokia 6120 Classic (sumber : http://www.forum.nokia.com/devices/6120_classic)

Untuk selanjutnya file SIS yang sudah terdaftar dan sudah di *download* dijalankan pada perangkat dengan cara cukup mudah. Dengan catatan perangkat terhubung ke komputer dan pada komputer sudah terinstal *PC suite.*

BAB 4
UJI COBA DAN ANALISA

4.1 Uji Coba Aplikasi

Setelah pada pengujian di emulator menunjukkan aplikasi bisa berjalan baik dan tidak ditemukan kesalahan, maka hal selanjutnya yang dilakukan adalah mengujinya pada perangkat telepon genggam. Seperti yang telah dijelaskan pada bab sebelumnya lingkungan untuk uji coba aplikasi adalah S60 3rd Ed FP1 dengan menggunakan telepon genggam merk Nokia seri 6120 Classic.

4.2 Langkah – Langkah Pengujian

Langkah pertama yang dilakukan adalah menginstal aplikasi yang sudah teregistrasi pada perangkat. Setelah berhasil diinstal maka selanjutnya adalah menjalankannya, seperti yang terlihat pada gambar di bawah ini :

Gambar 4.1 Hasil uji coba aplikasi pada perangkat.

Dari hasil uji coba didapatkan nilai cell id : 6713435 dengan menggunakan operator telkomsel. Untuk telkomsel *network id* nya adalah 10. *Network id* disebut juga MNC *(Mobile Network Code).* MNC digunakan dengan kombinasi dari MCC *(Mobile Country code)* untuk secara unik mengidentifikasi operator seluler yang menggunakan sistem GSM, CDMA (Untuk di Indonesia) dan beberapa jaringan

mobile satelit. Sedangkan *country id* atau MCC digunakan untuk mengidentifikasi MS dalam jaringan telepon nirkabel terutama sekali untuk jaringan GSM dan jaringan UMTS. Rekomendasi ITU E.212 menegaskan tentang MCC.

MCC adalah bagian dari IMSI *(International Mobile Subscriber Identity)* yang mana secara unik mengidentifikasi pelanggan dan menyimpannya pada SIM. *Country id* yang terdeteksi adalah 510 yang merupakan MCC dari Indonesia[9].

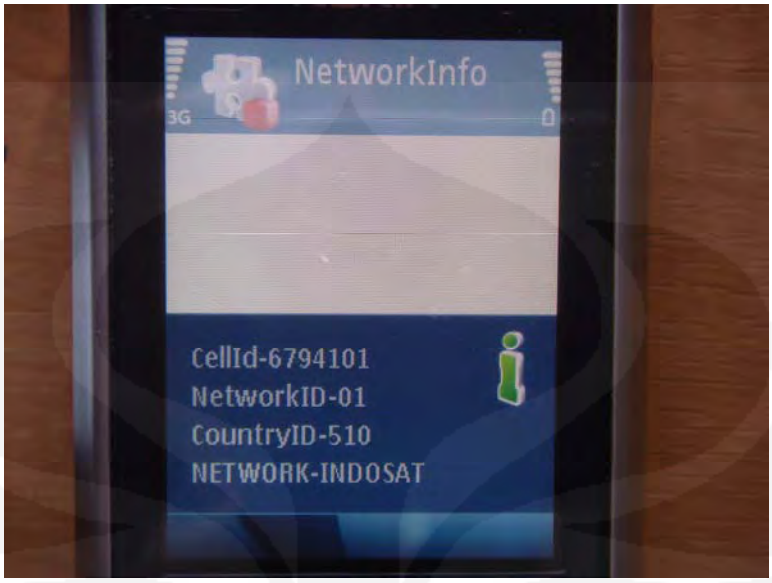
Tabel 4.1 berikut ini memuat daftar lengkap tentang MNC operator – operator seluler di Indonesia. Tersedia juga mengenai informasi kode Negara atau MCC.

Tabel 4.1 Daftar lengkap informasi operator *mobile* di Indonesia.

MCC	MNC	Brand	Operator	Status	Bands (MHz)	References and notes
510	00	PSN	PT Pasifik Satelit Nusantara (ACeS)	Operational	Satellite	
510	01	INDOSAT	PT Indonesian Satellite Corporation Tbk (INDOSAT)	Operational	GSM 900 / GSM 1800 / UMTS 2100	Formerly PT Satelindo
510	03	StarOne	PT Indosat	Operational	CDMA 800	
510	07	TelkomFlexi	PT Telkom	Operational	CDMA 800	
510	08	AXIS	PT Natrindo Telepon Seluler	Operational	GSM 1800 / UMTS 2100	
510	09	SMART	PT Smart Telecom	Operational	CDMA 1900	
510	10	Telkomsel	PT Telekomunikasi Selular	Operational	GSM 900 / GSM 1800 / UMTS 2100	
510	11	XL	PT Excelcomindo Pratama	Operational	GSM 900 / GSM 1800 / UMTS 2100	
510	20	TELKOMMobile	PT Telkom	Unknown	GSM 1800	Merged with Telkomsel
510	21	IM3	PT Indonesian Satellite Corporation Tbk (INDOSAT)	Not operational	GSM 1800	Merged with Indosat (MNC 01) MNC 21 not used anymore
510	27	Ceria	PT Sampoerna Telekomunikasi Indonesia	Operational	CDMA 450	
510	28	Fren/Hepi	PT Mobile-8 Telecom	Operational	CDMA 800	
510	89	3	PT Hutchison CP Telecommunications	Operational	GSM 1800 / UMTS 2100	
510	99	Esia	PT Bakrie Telecom	Operational	CDMA 800	

Sumber : http://en.wikipedia.org/wiki/Mobile_network_code

Selanjutnya untuk menguji kehandalan aplikasi dilakukan pengujian dengan menggunakan operator yang berbeda dan hasilnya diperlihatkan pada Gambar 4.2.



Gambar 4.2 Pengujian dengan operator Indosat.

Untuk operator yang berbeda nilai *cell id* yang didapatkan juga berbeda, karena setiap operator mempunyai sistem penomoran sendiri untuk masing – masing BTS yang berada dalam jaringannya. Nilai *cell Id* yang didapat adalah 6794101,dengan *network id* untuk Indosat adalah 01, sesuai dengan Tabel 4.1 diatas.

Uji coba selanjutnya menggunakan jaringan milik operator Exelcomindo hasil yang didapatkan terlihat pada Gambar 4.3.

Gambar 4.3 Hasil pengujian dengan operator Exelcomindo.

4.3 Pengambilan Data

Pengambilan data dimaksudkan untuk mengetahui sejauh mana aplikasi bereaksi terhadap posisi yang berubah – ubah. Metode pengambilan data dilakukan secara acak dengan mengambil sampel 15 titik daerah. Daerah yang menjadi objek adalah lingkungan di dalam dan sekitar kampus UI. Tabel 4.2 berikut ini menyajikan data – data yang sudah berhasil diambil.

Tabel 4.2 Hasil pengambilan data.

No	Nama Tempat	Cell Id Operator 1 (Telkomsel)	Cell Id Operator 2 (Indosat)	Cell Id Operator 3 (Exelcomindo)
1.	Wisma Permata,Pondok Cina	6713435	6794101	252770
2.	Universitas Gunadarma, Margonda	6700714	6793379	249621
3.	Departemen Elektro, Universitas Indonesia	6700713	6790319	249829
4.	Perpustakaan Pusat UI	6700713	6790320	249829
5.	Fakultas Hukum, Universitas Indonesia	50653	9656	3373
6.	Fakultas MIPA Universitas Indonesia	6700714	6793369	249839
7.	Fakultas Ilmu Kesehatan Masyarakat, Universitas Indonesia	6700714	6793381	249620
8.	Fakultas Ekonomi, Universitas Indonesia	6700713	6790319	249829
9.	FISIP UI	6700713	6793361	252760
10.	Masjid UI	50612	6790320	252770
11.	Margo City, Margonda	50645	6790736	204802
12.	Depok Town Square, Margonda	6713435	6791576	249839
13.	Engineering Centre	6700713	6790319	11713
14.	Markas Resimen Mahasiswa UI	6700313	6793431	252761
15.	Stasiun UI	6700315	6793361	252760

Selanjutnya dari data tabel 4.2 dibuat klasifikasi untuk masing – masing operator yang memperlihatkan daerah – daerah yang memiliki *cell id* yang sama dengan prosentase cakupan pada daerah yang diuji.

Untuk operator Telkomsel ditunjukkan pada Tabel 4.3 dibawah ini :

Tabel 4.3 Persentase *coverage* suatu *Cell Id* untuk Telkomsel

No	Nilai Cell Id	Cakupan Area	Prosentase cakupan pada keseluruhan daerah yang diuji (%).
1.	6700713	- Departemen Elektro UI, - Perpustakaan Pusat UI, - Fakultas Ekonomi UI, - FISIP UI, - Engineering Centre	33,3
2.	6700714	- Universitas Gunadarma Margonda, - Fakultas MIPA UI, - Fakultas Ilmu Kesehatan Masyarakat, UI	20
3.	6713435	- Wisma Permata,Pondok Cina - Depok Town Square, Margonda	13,3
4.	6700313	- Markas Resimen Mahasiswa UI	6,6
5.	6700315	- Stasiun UI	6,6
6.	50612	- Masjid UI	6,6
7.	50645	- Margo City, Margonda	6,6
8.	50653	- Fakultas Hukum UI	6,6
	Total		99, 6

Tabel 4.4 berikut adalah memuat informasi cell id untuk operator Indosat meliputi daerah – daerah dengan *cell Id* sama sekaligus prosentase cakupannya.

Tabel 4.4 Persentase *coverage* suatu *Cell Id* untuk Indosat.

No	Nilai Cell Id	Cakupan Area	Prosentase cakupan pada keseluruhan daerah yang diuji (%).
1.	6790319	- Departemen Elektro UI - Fakultas Ekonomi UI - Engineering Centre	20
2.	6790320	- Perpustakaan Pusat UI, - Masjid UI	13,3
3.	6790736	- Margo City, Margonda	13,3

No	Nilai Cell Id	Cakupan Area	Prosentase cakupan pada keseluruhan daerah yang diuji (%).
4.	6793361	- FISIP UI - Stasiun UI	13,3
5.	6791576	- Depok Town Square, Margonda	6,6
6.	6793379	- Universitas Gunadarma	6,6
7.	6793369	- Fakultas MIPA UI	6,6
8.	6793381	- Fakultas Ilmu Kesehatan Masyarakat UI	6,6
9.	6793431	- Markas Resimen Mahasiswa UI	6,6
10.	6794101	- Wisma Permata, Pondok Cina	6,6
11.	9656	- Fakultas Hukum UI	6,6
	Total		99, 4

Tabel 4.5 berikut adalah memuat informasi cell id untuk operator Exelcomindo meliputi daerah – daerah dengan *cell Id* sama sekaligus prosentase cakupannya.

Tabel 4.5 Persentase *coverage* suatu *Cell Id* untuk Exelcomindo.

No	Nilai Cell Id	Cakupan Area	Prosentase cakupan pada keseluruhan daerah yang diuji (%).
1.	252760	- FISIP UI - Stasiun UI	13,3
2.	252770	- Masjid UI, - Wisma Permata Pondok Cina	13,3
3.	249829	- Departemen Elektro UI - Perpustakaan Pusat UI - Fakultas Ekonomi Ui	20
3.	249839	- Fakultas MIPA UI - Depok Town Square, Margonda	13,3
4.	204802	- Margo City, Margonda	6,6
5.	249620	- Fakultas Ilmu Kesehatan Masyarakat UI	6,6
6.	249621	- Universitas Gunadarma, Margonda	6,6

No	Nilai Cell Id	Cakupan Area	Prosentase cakupan pada keseluruhan daerah yang diuji (%).
7.	252761	- Markas Resimen Mahasiswa UI	6,6
8.	11713	- Engineering Centre	6,6
9.	3373	- Fakultas Hukum UI	6,6
	Total		99, 5

4.4 Analisa Data

4.4.1 Sisi Perangkat Keras.

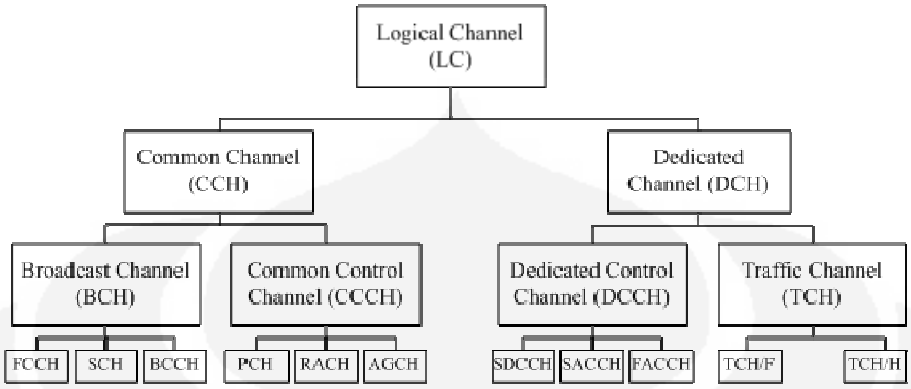
Dari data yang dihasilkan, terlihat di beberapa tempat nilai *cell id* berbeda walaupun dengan operator yang sama. Seperti yang terjadi pada Wisma Permata daerah Pondok Cina dengan Universitas Gunadarma yang berada di jalan Margonda. Hal ini dipengaruhi beberapa faktor seperti pentransmissian, kuat sinyal yang diterima oleh perangkat, kemudian radius *coverage* dari suatu BTS.

4.4.2 Pentransmissian Informasi

Sebagian besar dari informasi yang ditransmisikan antara perangkat dan BTS umumnya berupa informasi pelanggan (berupa suara atau data) dan control data pensinyalan. Tergantung pada tipe informasi yang ditransmisikan pada kanal logika yang berbeda. Kanal logika ini membawa data user, baik bit informasi (suara/data) maupun *signaling* pada perangkat *mobile* atau *base station*.

Kanal logika digambarkan ke dalam beberapa kanal fisik *(time slot)*. Sebagai contoh : Percakapan digital dibawa dengan kanal logika yang disebut kanal trafik (TCH), yang mana selama transmisi dapat dialokasikan sebuah kanal fisik tertentu.

Pembagian kanal logika pada GSM ditunjukkan pada Gambar 4.4 berikut ini :



Gambar 4.4 Kanal logika pada GSM

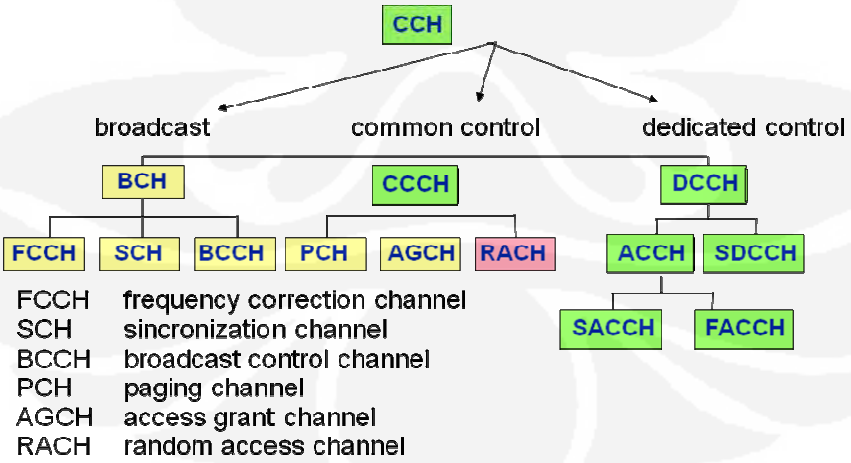
Kanal logika terdiri atas :

1. Kanal Trafik disebut juga (TCH : *traffic channels*)
2. Kanal Kontrol disebut juga (CCH : *control channels*)

Adapun untuk pendeteksian cell id melalui perangkat *mobile* menggunakan kanal kontrol. Pada kanal kontrol terdapat kanal signaling digunakan untuk komunikasi antara perangkat – perangkat jaringan. *Control Channels* (CCH) terdiri dari tiga tipe yaitu :

1. *Broadcast channel* (BCH)
2. *Common Control Channel* (CCCH)
3. *Dedicated Control Channel* (DCCH)

Selanjutnya pembagian CCH diperlihatkan pada Gambar 4.5 dibawah ini :



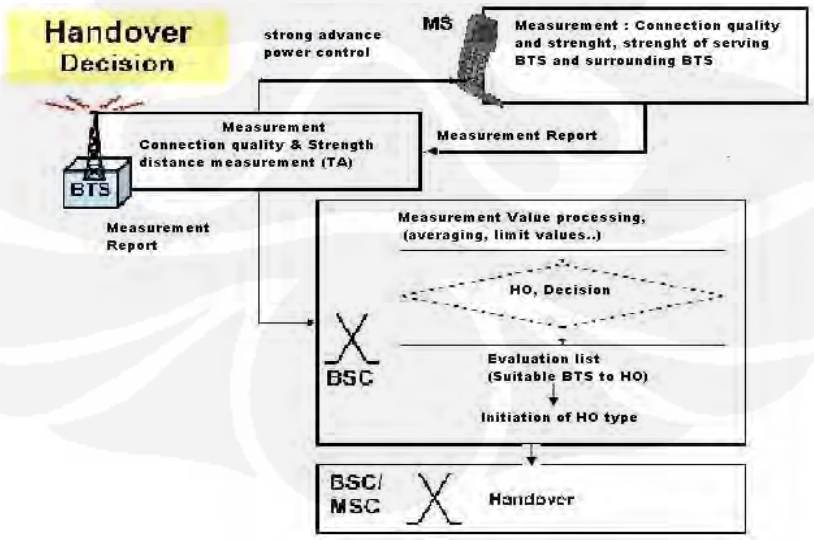
Gambar 4.5 Kontrol *channel* pada GSM.

Pada gambar diatas bagian CCH yang berkaitan dengan pentrasnmissian informasi cell dari BTS ke MS adalah BCCH (*Broadcast Control Channel*). BCCH merupakan informasi BTS mengenai penggunaan frekuensi, kombinasi kanal, kelompok *paging*, dan informasi sel sekitar yang dimonitor MS secara periodik paling sedikit tiap 30 detik ketika MS *switch on*.

4.4.4 Proses *Handover*

Handover merupakan proses pengalihan kanal *traffic* secara otomatis pada *Mobile Station* (MS) yang sedang digunakan untuk berkomunikasi tanpa terjadinya pemutusan hubungan. Hal ini menjelaskan bahwa *handover* pada dasarnya adalah sebuah *call* koneksi yang bergerak dari satu sel ke sel lainnya. Secara umum *Handover d*apat didefenisikan sebagai prosedur, dimana ada perubahan layanan pada MS dari satu *Base Station* (BS) ke BS yang lain.

Proses ini memerlukan alat pendeteksi untuk mengubah status *dedicated node* (persiapan *handover*) dan alat untuk men*switch* komunikasi yang sedang berlangsung dari suatu kanal pada sel tertentu ke kanal yang lain pada sel yang lain. Keputusan untuk sebuah *handover* dibuat oleh *Base Station Centre* (BSC), yaitu dengan mengevaluasi secara *permanent* pengukuran yang diambil oleh BTS dan MS. Pengukuran rata-rata (Px) oleh BSC dibandingkan dengan nilai-nilai ambang batas (*threshold*); jika Px melebihi nilai *threshold* maka dimulai proses *handover* dengan mencari sebuah sel target yang cocok.



Gambar 4.6 Proses *handover*.

Tahap-tahap dari proses *handover*

- Tahap Pengukuran (*Measurement*), dilakukan pengukuran informasi penting yang dibutuhkan untuk tahap decision. Pengukuran arah DL yang lakukan oleh MS adalah sebesar Ec/Io dari CPICH sel yang sedang melayani dan sel-sel tetangga.
- Tahap Keputusan (*Decision*), hasil pengukuran di bandingkan dengan *threshold* yang telah di tetapkan sebelumnya. Kemudian akan diputuskan apakah akan dilakukan *handover* atau tidak. Algoritma *handover* yang berbeda akan memiliki kondisi *trigger* yang berbeda pula. Tahap Eksekusi (*Execution*), proses *handover* selesai dan parameter relatif diubah berdasarkan jenis *handover*-nya. Sebagai contoh hubungan dengan *Node B* apakah ditambah atau diputuskan.

4.5 Analisa Sisi Perangkat Lunak

Pada aplikasi yang dibangun API *(Application Programming Interface)* yang digunakan adalah `Etel3rdParty.h` dengan menggunakan *class* `CTelephony`. *class* ini menyediakan antarmuka sederhana untuk system telephony pada telepon[5]. Pada dasarnya *class* ini menyediakan 2 layanan yaitu :

1. Dapat menemukan informasi tentang telepon. *Class* ini menyediakan dukungan untuk mendapatkan kembali pengaturan telepon, saluran informasi, fungsi panggilan, informasi jaringan, tambahan pelayanan pengaturan dasar.
2. Dapat memutar angka, menjawab dan mengendalikan panggilan suara.

Bagian dari library ini yang memuat tentang informasi jaringan ditunjukkan sebagai berikut :

```
/*********************************************/
      // CTelephony::TNetworkInfoV1

      /**
      Maximum size of the network country code.
      */
      enum { KNetworkCountryCodeSize   = 4  };
      /**
      Maximum size of the network identity..
      */
      enum { KNetworkIdentitySize      = 8  };
      /**
```

```
Maximum size of the network display tag..
*/
enum { KNetworkDisplayTagSize    = 32 };
/**
Maximum size of the short network name.
*/
enum { KNetworkShortNameSize     = 8  };
/**
Maximum size of the long network name.
*/
enum { KNetworkLongNameSize      = 16 };
/**
Maximum size of the short network name version 2.
*/
enum { KNetworkShortNameSizeV2   = 10  };
/**
Maximum size of the long network name version 2.
*/
enum { KNetworkLongNameSizeV2    = 20 };

/**
Defines Current Network Modes.
*/
enum TNetworkMode
      {
      /**
      Network mode is unknown.
      */
      ENetworkModeUnknown,
      /**
      Mobile device is not registered.
      */
      ENetworkModeUnregistered,
      /**
      GSM/GPRS or DCS1800 network.
      */
      ENetworkModeGsm,
      /**
      AMPS network.
      */
      ENetworkModeAmps,
      /**
      CDMA (IS-95) network.
      */
      ENetworkModeCdma95,
      /**
      CDMA (cdma2000) network.
      */
      ENetworkModeCdma2000,
      /**
      WCDMA (UTRA Frequency Division Duplex (FDD)) network.
      */
      ENetworkModeWcdma,
      /**
      TD-CDMA (UTRA Time Division Duplex (TDD)) network.
       */
      ENetworkModeTdcdma
      };

/**
The access technology that the network is based on.
*/
enum TNetworkAccess
      {
      /**
```

```
            This is used when there is no network activity and therefore
no RAT active.
            */
            ENetworkAccessUnknown,
            /**
            The access technology is GSM.
            */
            ENetworkAccessGsm,
            /**
            The access technology is GSM COMPACT.
            However GSM COMPACT systems which use GSM frequency bands
but with the CBPCCH broadcast channel are considered as a separate access
technology from GSM.
            */
            ENetworkAccessGsmCompact,
            /**
            The access technology is UTRAN (UMTS Network).
            */
            ENetworkAccessUtran
            };
```

Bagian ini berfungsi untuk mendapatkan informasi jaringan saat ini yang terdaftar dengan menggunakan metode tak sinkron. Funsi ini tidak tersedia atau tidak berfungsi ketika perangkat berada dalam mode *flight.*

BAB 5
KESIMPULAN

Kesimpulan :

1. Dari hasil pengujian di lapangan *Network parameter* pada GSM meliputi *CellID, Network ID, Country ID* dan *Network Name* bisa dideteksi dengan menggunakan fungsi dan *library* API symbian C++.
2. Kekurangan dari aplikasi ini masih belum bisa mendeteksi nama lokasi dikarenakan fungsi dan *library* pendukungnya yang tidak lagi untuk publik.
3. Dari hasil pengujian di 15 titik dilapangan, didapatkan data bahwa tiap *cell id* memiliki *coverage* tertentu, dan untuk masing – masing operator cell id yang memiliki persentase coverage tertinggi adalah :
   - Telkomsel (*Cell Id* : 6700713) = 30%
   - Indosat (*Cell Id* : 6790319) = 20%
   - Exelcomindo (*Cell Id* : 249829) = 20%
4. Dalam pembuatan aplikasi ini ada 3 tahap utama yang terpenting yaitu instalasi tools pendukung, pengembangan proyek pada Carbide C++, dan penerapan pada perangkat dengan registrasi di *SymbianSigned.*
5. Metode pendaftaran di *symbiansigned* menggunakan *open signed online* dengan pertimbangan untuk pengujian tingkat personal dan ketiadaan *publisher ID.*
6. *Tools* di dalam platform Symbian sangat layak untuk dipertimbangkan sebagai media pengembangan aplikasi mobile dikarenakan *open source* dan bebas.
7. Untuk pengembangan yang lebih lanjut bisa ditambahkan *cell name* dengan menggunakan fungsi dan library API yang terkait.
8. Symbian OS bisa dikembangkan dengan IDE yang lain tidak harus IDE ber *platform* C++. Untuk pengembangan aplikasi khususnya mengenai LBS bisa menggunakan IDE berbasis Java. Dan agar lebih akurat metode yang digunakan adalah A-GPS bukan *CellId.*

DAFTAR REFERENSI

[1] *Symbian OS*. (n.d). Diakses 15 April 2009. http://id.wikipedia.org/wiki/Symbian_OS

[2] *Software Development Kit*.(n.d). Diakses 15 April 2009. http://en.wikipedia.org/wiki/Software_development_kit

[3] *Carbide c++*.(n.d). Diakses 18 April 2009. http://www.forum.nokia.com/Tools_Docs_and_Code/Tools/IDEs/Carbide.c++/

[4] *S60 Platforms and device SDKs*. (n.d). Diakses 20 April 2009. http://www.forum.nokia.com/Tools_Docs_and_Code/Tools/Platforms/S60_Platform_SDKs/

[5] *Ctelephony in Telephony Etel_3rd_Party_API*. (n.d). Diakses 01 Mei 2009 http://library.forum.nokia.com/

[6] Symbianyucca. "Retrieve GSM cell id". Online Posting. 15 May 2003. Home > Community > Discussion Boards > Development Platforms > Symbian C++ > Symbian Networking & Messaging. Diakses 13 Mei 2009. http://discussion.forum.nokia.com/forum/

[7] Wiki (2009, 10 April). *How do I start programming for Symbian O?*. Diakses 25 April 2009. http://wiki.forum.nokia.com/index.php/How_do_I_start_programming_for_Symbian_OS%3F.

[8] *Open Signed Online*. (n.d). Diakses 20 Mei 2009 https://www.symbiansigned.com/app/page/public/openSignedOnline.do

[9] Kadir, Abdul. *Pemrograman C++*. (1995). Yogyakarta: Andi.

[10] Raharjo, Budi. (2006) *Pemrograman C++*. Bandung: Informatika.

[11] Wibisono, Gunawan, Usman, U.K., Hantoro, G.D. (2007) *Konsep Teknologi Seluler*. Bandung: Informatika.

DAFTAR LAMPIRAN

```
// Etel3rdParty.h
//
// Copyright (c) 2003-2005 Symbian Software Ltd.  All rights reserved.
//

// Etel3rdParty.dll interface, using CTelephony class defined here.

/**
@file

Etel 3rd Party API Header file

Describes the Etel 3rd Party API - classes, methods and types.
*/


#ifndef ETEL3RDPARTY_H__
#define ETEL3RDPARTY_H__

#include <e32base.h>

const TInt   KTelephonyMajorVersionNumber=7; ///< Major version-number
const TInt   KTelephonyMinorVersionNumber=0; ///< Minor version-number
const TInt   KTelephonyBuildVersionNumber=0; ///< Build version-number

// Panic numbers returned to applications. 0 upwards...
const TInt KTelephonyPanicIllegalReuse = 0;    ///< Client code has
attempted to re-post an asynchronous request before the original request
has completed.

class CTelephonyFunctions;       // forward declaration

/**
This class provides a simple interface to the phone's telephony
system. It provides two services:

1. You can find out information about the phone. This class
provides support for retrieving Phone Settings, Line Information, Call
Functionality, Network Information and (basic) Supplementary Service
Settings.

2. You can dial, answer and control voice calls.  You cannot make fax or
data calls.

Architecturally, CTelephony provides an interface to the Telephony
Multimode API.  This interface exposes only a subset of the complete
Multimode functionality.

@publishedAll
@released
 */
class CTelephony : public CBase
      {
public:

//////////////////////////////////////////////////////////
//General Functionality
//////////////////////////////////////////////////////////

      /**
      Unique parameter class identifier.
```

```
	*/
	enum
		{
		/**
		Unique reference identifier for Etel 3rd Party v1.0
parameter classes.
		*/
		KEtelISVV1 = 1,
		/**
		Unique reference identifier for Etel 3rd Party v2.0
parameter classes.
		*/
		KEtelISVV2 = 2
		};

	class TEtelISVType
	/**
	Specifies which version of an API parameter a client has used.
	For most ETel ISV v1.0/2.0 parameters this will equal KETelISV1.
	*/
		{
	public:
		IMPORT_C TInt VersionId() const;
	protected:
		TEtelISVType();
	protected:
		TInt iVersionId;
		};

	/**
	The maximum number of digits allowed in a multimode telephone
number.
	*/
	enum { KMaxTelNumberSize = 100 };

	/**
	Address types.
	*/
	enum TTelNumberTON
		{
		/**
		User or the network has no knowledge of the type of number.
	    */
		EUnknownNumber,
		/**
		International number.
	    */
		EInternationalNumber,
		/**
		National number.
	    */
		ENationalNumber,
		/**
		Administration/service number specific to the serving
network, e.g. used to
		access an operator.
	    */
		ENetworkSpecificNumber,
		/**
		Subscriber number.
	    */
		ESubscriberNumber,
		/**
		Alphanumeric number.
	    */
```

```
            EAlphanumericNumber,
            /**
            Abbreviated number.
        */
            EAbbreviatedNumber
            };

      /**
      Number Plan Indicator.
      */
      enum TTelNumberNPI
            {
            /**
            User or the network has no knowledge of the numbering plan.
        */
            EUnknownNumberingPlan            = 0,
            /**
            ISDN/telephony numbering plan.
        */
            EIsdnNumberPlan                        = 1,
            /**
            Data numbering plan.
        */
            EDataNumberPlan                        = 3,
            /**
            Telex numbering plan.
        */
            ETelexNumberPlan                 = 4,
            /**
            Service centre specific plan used to indicate a numbering
plan specific to external
            Short Message entities attached to the SMSC.
        */
            EServiceCentreSpecificPlan1      = 5,
            /**
            Service centre specific plan used to indicate a numbering
plan specific to external
            Short Message entities attached to the SMSC.
        */
            EServiceCentreSpecificPlan2      = 6,
            /**
            National numbering plan.
        */
            ENationalNumberPlan              = 8,
            /**
            Private numbering plan.
        */
            EPrivateNumberPlan               = 9,
            /**
            ERMES numbering plan.
        */
            EERMESNumberPlan                 = 10
            };

      /**
      A typedef to hold the telephone number.
      */
      typedef TBuf<KMaxTelNumberSize> TTelNumber;

      class TTelAddress
      /**
      Defines API abstraction of a mobile telephone number.
      */
            {
      public:
```

```
		IMPORT_C TTelAddress();
	public:
		/**
		Type of number.
		*/
		TTelNumberTON                           iTypeOfNumber;
		/**
		Number plan.
		*/
		TTelNumberNPI                           iNumberPlan;
		/**
		Telephone number.
		*/
		TTelNumber                              iTelNumber;
		};

//////////////////////////////////////////////////////////
// Phone Functionality
//////////////////////////////////////////////////////////

	/**
	Max size of Manufacturer Id.
	*/
	enum { KPhoneManufacturerIdSize = 50 };
	/**
	Max size of Phone Model Id.
	*/
	enum { KPhoneModelIdSize         = 50 };
	/**
	Max size ofSerial Number.
	*/
	enum { KPhoneSerialNumberSize    = 50 };


	class TPhoneIdV1 : public TEtelISVType
	/**
	Defines the mobile phone identity.
	*/
		{
	public:
		IMPORT_C TPhoneIdV1();
	public:
		/**
		Phone manufacturer identification, in character string
format.
		*/
		TBuf<KPhoneManufacturerIdSize>   iManufacturer;
		/**
		Phone model identification, in character string format.
		*/
		TBuf<KPhoneModelIdSize>          iModel;
		/**
		Phone serial number (IMEI or ESN), in character string
format.
		*/
		TBuf<KPhoneSerialNumberSize>     iSerialNumber;
		};

	/**
	A typedef'd packaged CTelephony::TPhoneIdV1 for passing through a
generic API method
	*/
	typedef TPckg<TPhoneIdV1> TPhoneIdV1Pckg;

	/**
```

```
	Maximum size of IMSI identifier.
	*/
	enum { KIMSISize = 15 };

	class TSubscriberIdV1 : public TEtelISVType
	/**
	Defines the Subscriber (IMSI) Id
	*/
		{
	public:
		IMPORT_C TSubscriberIdV1();
	public:
		/**
		IMSI identifier.
		*/
		TBuf<KIMSISize> iSubscriberId;
		};

	/**
	A typedef'd packaged CTelephony::TSubscriberIdV1 for passing
through a generic API method.
	*/
	typedef TPckg<TSubscriberIdV1> TSubscriberIdV1Pckg;

	/**
	The flight mode status.
	*/
	enum TFlightModeStatus
		{
		/**
		Flight mode is off.
		*/
		EFlightModeOff      = 0,
		/**
		Flight mode is on.
		*/
		EFlightModeOn = 1,
		};

	class TFlightModeV1 : public TEtelISVType
	/**
	Defines the flight mode status.
	*/
		{
	public:
		IMPORT_C TFlightModeV1();
	public:
		/**
		The current status of the mobile radio interface and
bluetooth.
		*/
		TFlightModeStatus iFlightModeStatus;
		};

	/**
	A typedef'd packaged CTelephony::TFlightModeV1 for passing through
a generic API method.
	*/
	typedef TPckg<TFlightModeV1> TFlightModeV1Pckg;

	/**
	The mobile phone indicators.
	*/
	enum TPhoneIndicators
		{
```

```
		/**
		If bit-flag is set to '1' indicates that the battery charger
		is connected to the phone. If bit-flag is set to '0'
indicates
		that the battery charger is disconnected

		For capability: If bit-flag is set to '0' indicates that the
		battery charger indication reporting is unavailable.
		*/
		KIndChargerConnected	= 0x00000001,
		/**
		If bit-flag is set to '1' indicates that network service is
		available. If bit-flag is set to '0' indicates that network
		service is unavailable

		For capability: If bit-flag is set to '0' indicates that the
		network availability indication reporting is unavailable.
		*/
		KIndNetworkAvailable	= 0x00000002,
		/**
		If bit-flag is set to '1' indicates that a call is in
		progress. If set to '0' indicates that a call is not in
		progress

		For capability: If bit-flag is set to '0' indicates
		that the call is in progress indication reporting is
		unavailable.
		*/
		KIndCallInProgress	= 0x00000004
		};

	class TIndicatorV1 : public TEtelISVType
	/**
	Contains indicator parameters:
	@see TPhoneIndicators
	*/
		{
	public:
		IMPORT_C TIndicatorV1();
	public:
		/**
		The value of the indicators.  It is the sum
		of CTelephony::TPhoneIndicators constants.
		*/
		TUint32 iIndicator;
		/**
		The supported (available) indicator capability that the
telephony service module offers.
		It is the sum of CTelephony::TPhoneIndicators constants.
		*/
		TUint32 iCapabilities;
		};

	/**
	A typedef'd packaged CTelephony::TIndicatorV1 for passing through
a generic API method.
	*/
	typedef TPckg<TIndicatorV1> TIndicatorV1Pckg;

	/**
	The mobile phone battery status.
	*/
	enum TBatteryStatus
		{
		/**
```

```
		The phone software can not determine the phone's current
power status.
		*/
		EPowerStatusUnknown,
		/**
		The phone is powered by the battery.
		*/
		EPoweredByBattery,
		/**
		A battery is connected, but the phone is externally powered.
		*/
		EBatteryConnectedButExternallyPowered,
		/**
		No battery is connected.
		*/
		ENoBatteryConnected,
		/**
		Power fault.
		*/
		EPowerFault
		};

	class TBatteryInfoV1 : public TEtelISVType
	/**
	Defines contents of the battery status of the phone.
	*/
		{
	public:
		IMPORT_C TBatteryInfoV1();
	public:
		/**
		The power and battery status.
		*/
		TBatteryStatus iStatus;
		/**
		The percentage battery charge level.
		*/
		TUint iChargeLevel;
		};

	/**
	A typedef'd packaged CTelephony::TBatteryInfoV1 for passing
through a generic API method.
	*/
	typedef TPckg<TBatteryInfoV1> TBatteryInfoV1Pckg;

	class TSignalStrengthV1 : public TEtelISVType
	/**
	Defines the phone's current signal strength.
	*/
		{
	public:
		IMPORT_C TSignalStrengthV1();
	public:
		/**
		The value of signal strength.
		*/
		TInt32 iSignalStrength;
		/**
		The absolute number of signal "bars" that the phone should
display.
		*/
		TInt8 iBar;
		};
```

```
	/**
	A typedef'd packaged CTelephony::TSignalStrengthV1 for passing
through a generic API method.
	*/
	typedef TPckg<TSignalStrengthV1> TSignalStrengthV1Pckg;

	/**
	Pin 1/Pin 2 security
	*/
	enum TIccLock
		{
		/**
		Lock PIN1 on the ICC.
		*/
		ELockPin1,
		/**
		Lock PIN2 on the ICC.
		*/
		ELockPin2
		};

	/**
	Status of an ICC lock.
	*/
	enum TIccLockStatus
		{
		/**
		The status of the lock is unknown.
		*/
		EStatusLockUnknown,
		/**
		Lock is closed.

		User can not access functionality governed by this lock
until
		user correctly enters associated security code.
		*/
		EStatusLocked,
		/**
		Lock is open.

		User can access functionality governed by this lock
		*/
		EStatusUnlocked,
		/**
		Lock is blocked.

		User should enter the unblocking code to be able to switch
back to the unlocked mode.
		Blocking is enabled after a number of unsuccessful attempts
to verify PIN1/2.
		*/
		EStatusBlocked
		};

	/**
	Setting of the ICC lock.
	*/
	enum TIccLockSetting
		{
		/**
		The setting of the lock is unknown.
		*/
		ELockSetUnknown,
		/**
```

```
            Lock is enabled, and is available for use. Its status may be
            CTelephony::EStatusLocked, CTelephony::EStatusUnlocked,
            or CTelephony::EStatusBlocked.

            The associated security code will be required to unlock the
lock
            the next time the lock's status is
CTelephony::EStatusLocked.
            */
            ELockSetEnabled,
            /**
            Lock is disabled.  It is not available for use, so its
status
            is always CTelephony::EStatusUnlocked.
            */
            ELockSetDisabled
            };

      class TIccLockInfoV1 : public TEtelISVType
      /**
      Defines the Icc Lock parameters.
      */
            {
      public:
            IMPORT_C TIccLockInfoV1();
      public:
            /**
            The current status of the lock.
            */
            TIccLockStatus iStatus;
            /**
            The current availability of the lock. When not available
(i.e. the
            lock is not in use) then its status will always be
CTelephony::EStatusUnlocked.
            */
            TIccLockSetting iSetting;
            };

      /**
      A typedef'd packaged CTelephony::TIccLockInfoV1 for passing
through a generic API method.
      */
      typedef TPckg<TIccLockInfoV1> TIccLockInfoV1Pckg;

//////////////////////////////////////////////////////////
// Line Functionality
//////////////////////////////////////////////////////////

      /**
      Line types
      */
      enum TPhoneLine
            {
            /**
            Voice line.
            */
            EVoiceLine,
            /**
            Data line.
            */
            EDataLine,
            /**
            Fax line.
            */
```

```
		EFaxLine,
		};

	/**
	Describes the possible call or line states.
	*/
	enum TCallStatus
		{
		/**
		Indicates that the status is unknown.
		*/
		EStatusUnknown,
		/**
		Idle line status (no active calls).
		*/
		EStatusIdle,
		/**
		Call dialling status .
		*/
		EStatusDialling,
		/**
		Call ringing status.
		*/
		EStatusRinging,
		/**
		Call answering status.
		*/
		EStatusAnswering,
		/**
		Call connecting status.
		*/
		EStatusConnecting,
		/**
		Call connected status.
		*/
		EStatusConnected,
		/**
		Call is undergoing temporary channel loss and it may or may
not be reconnected.
		*/
		EStatusReconnectPending,
		/**
		Call disconnecting status.
		*/
		EStatusDisconnecting,
		/**
		Call on hold.
		*/
		EStatusHold,
		/**
		Call is transferring.
		*/
		EStatusTransferring,
		/**
		Call in transfer is alerting the remote party.
		*/
		EStatusTransferAlerting
		};

	class TCallStatusV1 : public TEtelISVType
	/**
	Defines the current call or line status.
	*/
		{
	public:
```

```
		IMPORT_C TCallStatusV1();
	public:
		/**
		The current call or line status.
		*/
		TCallStatus iStatus;
		};

	/**
	A typedef'd packaged CTelephony::TCallStatusV1 for passing through
a generic API method.
	*/
	typedef TPckg<TCallStatusV1> TCallStatusV1Pckg;

	/**
	Select a call defined by it current stats.
	*/
	enum TCallSelect
		{
		/**
		Currently active call.
		*/
		EActiveCall,
		/**
		Currently held (applicable only for voice) call.
		*/
		EHeldCall,
		/**
		Call during the setup/disconnecting process -
(dialling/connecting/answering/hanging up).
		*/
		EInProgressCall
		};

	class TCallSelectionV1 : public TEtelISVType
	/**
	Defines parameters to select a call, determined by its call state
for a specific line
	*/
		{
	public:
		IMPORT_C TCallSelectionV1();
	public:
		/**
		The current phone line selection.
		*/
		TPhoneLine iLine;
		/**
		The current call selection.
		*/
		TCallSelect iSelect;
		};

	/**
	A typedef'd packaged CTelephony::TCallSelectionV1 for passing
through a generic API method.
	*/
	typedef TPckg<TCallSelectionV1> TCallSelectionV1Pckg;

	/**
	Etel 3rd Party owned call identifiers
	*/
	enum TCallId
		{
		/**
```

```
                Call owned by this ISV application.
                */
                EISVCall1,
                /**
                Call owned by this ISV application.
                */
                EISVCall2,
                /**
                Max number of calls supported by Etel 3rd Party.
                */
                EISVMaxNumOfCalls
                };

        /**
        Remote party identity status.
        */
        enum TCallRemoteIdentityStatus
                {
                /**
                The remote party's identity can not be determined.
                */
                ERemoteIdentityUnknown,
                /**
                The remote party's identity is available.
                */
                ERemoteIdentityAvailable,
                /**
                The remote party has suppressed the transmission of its
identity.
                */
                ERemoteIdentitySuppressed
                };

        /**
        The direction of the call.
        */
        enum TCallDirection
                {
                /**
                The direction of the call is unknown.
                */
                EDirectionUnknown,
                /**
                The call was originated by this phone, i.e. it is an
outgoing call.
                */
                EMobileOriginated,
                /**
                The call was terminated by this phone, i.e. it is an
incoming call.
                */
                EMobileTerminated
                };

        /**
        Enumerated network security types.
        */
        enum TPhoneNetworkSecurity
                {
                /**
                The encryption level is NONE.
                */
                ECipheringOff,
                /**
```

```
            The encryption level is GSM.(standard encryption algorithms
for 2nd Generation Mobile networks).
            */
            ECipheringGSM,
            /**
            The encryption level is WDCMA.(standard encryption
algorithms for 3rd Generation Mobile networks).
            */
            ECipheringWCDMA
            };

      class TCallInfoV1 : public TEtelISVType
      /**
      Defines general information about a call.
      */
            {
      public:
            IMPORT_C TCallInfoV1();
      public:
            /**
            The current status of the call.
            */
            TCallStatus iStatus;
            /**
            The time & date the call started.
            */
            TDateTime iStartTime;
            /**
            The current duration of the call.
            */
            TTimeIntervalSeconds iDuration;
            /**
            The original number (including DTMF) dialled for an outgoing
call.
            */
            TTelAddress iDialledParty;
            /**
            The reason for termination of a finished call.
            Will equal KErrNone if the call ended normally and
KErrNotFound if the call has not ended.
            */
            TInt iExitCode;
            /**
            This attribute indicates whether Ciphering Status of a Call
is enabled or disabled.
            */
            TPhoneNetworkSecurity iSecurity;
            /**
            The call id of an ISV application owned call. For calls not
owned by this ISV application (-1) is returned.
            */
            TInt iCallId;
            };

      /**
      A typedef'd packaged CTelephony::TCallInfoV1 for passing through a
generic API method.
      */
      typedef TPckg<TCallInfoV1> TCallInfoV1Pckg;

      /**
      Maximum size of the calling party name.
      */
      enum { KCallingNameSize = 80 };
```

```
	class TRemotePartyInfoV1 : public TEtelISVType
		{
		/**
		Defines information about the remote party of a call.
		*/
	public:
		IMPORT_C TRemotePartyInfoV1();
	public:
		/**
		Indicates whether the remote party information in the rest
of this structure is valid or not.
		*/
		TCallRemoteIdentityStatus iRemoteIdStatus;
		/**
		Calling party name available through the CNAP supplementary
service (if provisioned).
		*/
		TBuf<KCallingNameSize> iCallingName;
		/**
		The phone number of the remote party if available.
		*/
		TTelAddress iRemoteNumber;
		/**
		The direction of the call and hence the role of the remote
party.
		i.e. if the call is mobile originated then the remote party
is the called party.
		Whereas if the call is mobile terminated then the remote
party is the calling party.
		*/
		TCallDirection iDirection;
		};

	/**
	A typedef'd packaged CTelephony::TRemotePartyInfoV1 for passing
through a generic API method.
	*/
	typedef TPckg<TRemotePartyInfoV1> TRemotePartyInfoV1Pckg;

//////////////////////////////////////////////////////////
// Call Functionality
//////////////////////////////////////////////////////////

	/**
	Caller Id restriction settings.
	*/
	enum TCallerIdentityRestrict
		{
		/**
		The default setting should be used if the user has not
explicitly requested their identity to be restricted/allowed.
		*/
		EIdRestrictDefault,
		/**
		The user has explicitly requested their identity to be sent
for this call.
		*/
		ESendMyId,
		/**
		The user has explicitly requested their identity not to be
sent for this call.
		*/
		EDontSendMyId
		};
```

```
	class TCallParamsV1 : public TEtelISVType
		{
		/**
		Defines the parameters used for set-up of a call.
		*/
	public:
		IMPORT_C TCallParamsV1();
	public:
		/**
		Call Id restriction setting to be used for this call.
		*/
		TCallerIdentityRestrict iIdRestrict;
		};

	/**
	A typedef'd packaged CTelephony::TCallParamsV1 for passing through
a generic API method.
	*/
	typedef TPckg<TCallParamsV1> TCallParamsV1Pckg;

	/**
	Mobile call control capabilities.
	*/
	enum TMobileCallControlCaps
		{
		/**
		Indicates that this call can be put on hold.
		This implies that the call is currently active and that
there is no other held call.
		*/
		KCapsHold    = 0x00000200,
		/**
		Indicates that this call can be resumed.
		This implies that the call is currently on hold and that
there is no other active call.
		*/
		KCapsResume = 0x00000400,
		/**
		Indicates that this call's state can be swapped to the
opposite state.
		This implies that this call is either active or held.
		There may be another call in the opposite state and if this
is the case then both calls will be simultaneously swapped to their
opposite state.
		*/
		KCapsSwap    = 0x00000800
		};

	class TCallCapsV1 : public TEtelISVType
		{
		/**
		Defines the dynamic capabilities of a call.
		@see TMobileCallControlCaps
		*/
	public:
		IMPORT_C TCallCapsV1();
	public:
		/**
		Call Id restriction setting to be used for this call.
		It is the sum of CTelephony::TMobileCallControlCaps
constants.
		*/
		TUint32 iControlCaps;
		};
```

```
	/**
	A typedef'd packaged CTelephony::TCallParamsV1 for passing through
a generic API method.
	*/
	typedef TPckg<TCallCapsV1> TCallCapsV1Pckg;


//////////////////////////////////////////////////////////
// Network Functionality
//////////////////////////////////////////////////////////

	/**
	The registration status of the phone.
	*/
	enum TRegistrationStatus
		{
		/**
		Registration status is unknown.
		*/
		ERegistrationUnknown,
		/**
		Not registered. The ME can not detect any other networks and
is not currently searching a new operator to register to.
		*/
		ENotRegisteredNoService,
		/**
		Not registered. The ME can detect other networks on which it
is possible to make emergency calls only.
		*/
		ENotRegisteredEmergencyOnly,
		/**
		Not registered, but the ME is currently searching a new
operator to register to.
		*/
		ENotRegisteredSearching,
		/**
		Registered, network busy.
		*/
		ERegisteredBusy,
		/**
		Registered on home network.
		*/
		ERegisteredOnHomeNetwork,
		/**
		Registration denied.
		*/
		ERegistrationDenied,
		/**
		Registered, roaming.
		*/
		ERegisteredRoaming
		};

	class TNetworkRegistrationV1 : public TEtelISVType
		/**
		Defines the current network registration status
		*/
		{
	public:
		IMPORT_C TNetworkRegistrationV1();
	public:
		/**
	  The current network registration status.
		*/
		TRegistrationStatus iRegStatus;
```

```
        };

    /**
    A typedef'd packaged CTelephony::TNetworkRegistrationV1 for
passing through a generic API method.
    */
    typedef TPckg<TNetworkRegistrationV1> TNetworkRegistrationV1Pckg;

    /*********************************************/
    // CTelephony::TNetworkInfoV1

    /**
    Maximum size of the network country code.
    */
    enum { KNetworkCountryCodeSize   = 4  };
    /**
    Maximum size of the network identity..
    */
    enum { KNetworkIdentitySize      = 8  };
    /**
    Maximum size of the network display tag..
    */
    enum { KNetworkDisplayTagSize    = 32 };
    /**
    Maximum size of the short network name.
    */
    enum { KNetworkShortNameSize     = 8  };
    /**
    Maximum size of the long network name.
    */
    enum { KNetworkLongNameSize      = 16 };
    /**
    Maximum size of the short network name version 2.
    */
    enum { KNetworkShortNameSizeV2   = 10  };
    /**
    Maximum size of the long network name version 2.
    */
    enum { KNetworkLongNameSizeV2    = 20 };

    /**
    Defines Current Network Modes.
    */
    enum TNetworkMode
        {
        /**
        Network mode is unknown.
        */
        ENetworkModeUnknown,
        /**
        Mobile device is not registered.
        */
        ENetworkModeUnregistered,
        /**
        GSM/GPRS or DCS1800 network.
        */
        ENetworkModeGsm,
        /**
        AMPS network.
        */
        ENetworkModeAmps,
        /**
        CDMA (IS-95) network.
        */
        ENetworkModeCdma95,
```

```
		/**
		CDMA (cdma2000) network.
		*/
		ENetworkModeCdma2000,
		/**
		WCDMA (UTRA Frequency Division Duplex (FDD)) network.
		*/
		ENetworkModeWcdma,
		/**
		TD-CDMA (UTRA Time Division Duplex (TDD)) network.
		 */
		ENetworkModeTdcdma
		};

	/**
	The access technology that the network is based on.
	*/
	enum TNetworkAccess
		{
		/**
		This is used when there is no network activity and therefore
no RAT active.
		*/
		ENetworkAccessUnknown,
		/**
		The access technology is GSM.
		*/
		ENetworkAccessGsm,
		/**
		The access technology is GSM COMPACT.
		However GSM COMPACT systems which use GSM frequency bands
but with the CBPCCH broadcast channel are considered as a separate access
technology from GSM.
		*/
		ENetworkAccessGsmCompact,
		/**
		The access technology is UTRAN (UMTS Network).
		*/
		ENetworkAccessUtran
		};

	/**
	Phone network status.
	*/
	enum TNetworkStatus
		{
		/**
		Status is unknown.
		*/
		ENetworkStatusUnknown,
		/**
		A network that the mobile device is allowed to register to.
		*/
		ENetworkStatusAvailable,
		/**
		Currently registered network.
		*/
		ENetworkStatusCurrent,
		/**
		A network that the ME is not allowed to register to.
		*/
		ENetworkStatusForbidden
		};

	/**
```

```
Mobile phone network band information.
*/
enum TNetworkBandInfo
	{
	/**
	The current band and band class is unknown.
	*/
	EBandUnknown,
	/**
	The network operates at 800MHz on Band A.
	*/
	E800BandA,
	/**
	The network operates at 800MHz on Band B.
	*/
	E800BandB,
	/**
	The network operates at 800MHz on Band C.
	*/
	E800BandC,
	/**
	The network operates at 1900MHz on Band A.
	*/
	E1900BandA,
	/**
	The network operates at 1900MHz on Band B.
	*/
	E1900BandB,
	/**
	The network operates at 1900MHz on Band C.
	*/
	E1900BandC,
	/**
	The network operates at 1900MHz on Band D.
	*/
	E1900BandD,
	/**
	The network operates at 1900MHz on Band E.
	*/
	E1900BandE,
	/**
	The network operates at 1900MHz on Band F.
	*/
	E1900BandF
	};

/**
Defines information related to a mobile phone network.

@see CTelephony::TNetworkInfoV2
*/
class TNetworkInfoV1 : public TEtelISVType
	{
public:
	IMPORT_C TNetworkInfoV1();
public:
	/**
	Mode of the network.
	*/
	TNetworkMode iMode;
	/**
	Status of the network
	*/
	TNetworkStatus iStatus;
	/**
```

```
            The MCC of the network.
            */
            TBuf<KNetworkCountryCodeSize>    iCountryCode;
            /**
            The network identity (NID in CDMA and MNC in GSM).
            */
            TBuf<KNetworkIdentitySize>              iNetworkId;
            /**
            The alpha-tag displayed when this is the serving network.
            */
            TBuf<KNetworkDisplayTagSize>     iDisplayTag;
            /**
            On GSM/WCDMA networks, the short name (up to 8 characters)
of the network operator.
            */
            TBuf<KNetworkShortNameSize>      iShortName;
            /**
            On CDMA networks, the band and band class of the CDMA
network operator.
            */
            TNetworkBandInfo iBandInfo;
            /**
            On CDMA networks, the system identity (SID) of the CDMA or
AMPS network
            */
            TBuf<KNetworkIdentitySize>              iCdmaSID;
            /**
            On GSM/WCDMA networks, the long name (up to 16 characters)
of the network operator.
            */
            TBuf<KNetworkLongNameSize>              iLongName;
            /**
            On GSM/WCDMA networks, the type of network access.
            */
            TNetworkAccess iAccess;
            /**
            On GSM/WCDMA networks, boolean indicating whether the
location area is known.
            */
            TBool iAreaKnown;
            /**
            On GSM/WCDMA networks, the location area code.  Only valid
when iAccess is true.
            */
            TUint iLocationAreaCode;
            /**
            On GSM/WCDMA networks, the cell identity code.  Only valid
when iAccess is true.
            */
            TUint iCellId;
            };

      /**
      A typedef'd packaged CTelephony::TNetworkInfoV1 for passing
through a generic API method.
      */
      typedef TPckg<TNetworkInfoV1> TNetworkInfoV1Pckg;

      class TNetworkNameV1 : public TEtelISVType
      /**
      Defines the name network name.

   Note: This information is only available on GSM/WCDMA networks.
      */
            {
```

```
	public:
		IMPORT_C TNetworkNameV1();
	public:
		/**
		The displayed name (up to 16 characters) of the network
provider.
		*/
		TBuf<KNetworkLongNameSize>		iNetworkName;
		};

	/**
	A typedef'd packaged CTelephony::TNetworkNameV1 for passing
through a generic API method.
	*/
	typedef TPckg<TNetworkNameV1> TNetworkNameV1Pckg;

	class TOperatorNameV1 : public TEtelISVType
	/**
	Defines the current operator.

    Note: This information is only available on GSM/WCDMA networks.
	*/
		{
	public:
		IMPORT_C TOperatorNameV1();
	public:
		/**
		 The displayed name (up to 16 characters) of the network
operator.
		*/
		TBuf<KNetworkLongNameSize>		iOperatorName;
		};

	/**
	A typedef'd packaged CTelephony::TOperatorNameV1 for passing
through a generic API method.
	*/
	typedef TPckg<TOperatorNameV1> TOperatorNameV1Pckg;


	/**
	Defines information related to a mobile phone network.
	TNetworkInfoV2 differs from Version 1 (TNetworkInfoV1) in that the
	length of iShortName and iLongName are 10 and 20 respectively
	(rather than 8 and 16).

	TNetworkInfoV2 should be used in preference to TNetworkInfoV1.
The
	short and long Netowork names provided in this V2 class are of the
	length supplied by the platform.  The V1 class's names will be
	truncated if the platform provides names longer than its limits.
	*/
	class TNetworkInfoV2 : public TEtelISVType
		{
	public:
		IMPORT_C TNetworkInfoV2();
	public:
		/**
		Mode of the network.
		*/
		TNetworkMode iMode;
		/**
		Status of the network.
		*/
		TNetworkStatus iStatus;
```

```
            /**
            The MCC of the network.
            */
            TBuf<KNetworkCountryCodeSize>    iCountryCode;
            /**
            The network identity (NID in CDMA and MNC in GSM).
            */
            TBuf<KNetworkIdentitySize>       iNetworkId;
            /**
            The alpha-tag displayed when this is the serving network.
            */
            TBuf<KNetworkDisplayTagSize>     iDisplayTag;
            /**
            On GSM/WCDMA networks, the short name (up to 10 characters)
of the network operator.
            */
            TBuf<KNetworkShortNameSizeV2>    iShortName;
            /**
            On CDMA networks, the band and band class of the CDMA
network operator.
            */
            TNetworkBandInfo                 iBandInfo;
            /**
            On CDMA networks, the system identity (SID) of the CDMA or
AMPS network.
            */
            TBuf<KNetworkIdentitySize>       iCdmaSID;
            /**
            On GSM/WCDMA networks, the long name (up to 20 characters)
of the network operator.
            */
            TBuf<KNetworkLongNameSizeV2>     iLongName;
            /**
            On GSM/WCDMA networks, the type of network access.
            */
            TNetworkAccess                   iAccess;
            /**
            On GSM/WCDMA networks, boolean indicating whether the
location area is known.
            */
            TBool                            iAreaKnown;
            /**
            On GSM/WCDMA networks, the location area code.  Only valid
when iAccess is true.
            */
            TUint                            iLocationAreaCode;
            /**
            On GSM/WCDMA networks, the cell identity code.  Only valid
when iAccess is true.
            */
            TUint                            iCellId;
            };

      /**
      A typedef'd packaged CTelephony::TNetworkInfoV2 for passing
      through a generic API method.
      */
      typedef TPckg<TNetworkInfoV2> TNetworkInfoV2Pckg;

      /**
      Defines the name network name.
      TNetworkNameV2 differs from Version 1 (TNetworkNameV1) in that the
      length of iNetworkName is 20 (rather than 16).
```

```
       TNetworkNameV2 should be used in preference to TNetworkNameV1.
The
       Netowork name provided in this V2 class is of the length supplied
       by the platform.  The V1 class's network name will be truncated if
       the platform provides a name longer than V1 name's limit.

    Note: This information is only available on GSM/WCDMA networks.
       */
       class TNetworkNameV2 : public TEtelISVType
              {
       public:
              IMPORT_C TNetworkNameV2();
       public:
              /**
              The displayed name (up to 20 characters) of the network
provider.
              */
              TBuf<KNetworkLongNameSizeV2> iNetworkName;
              };

       /**
       A typedef'd packaged CTelephony::TNetworkNameV2 for passing
through
       a generic API method.  TNetworkNameV2 differs from Version 1 in
       that the length of iNetworkName is 20 rather than 16.
       */
       typedef TPckg<TNetworkNameV2> TNetworkNameV2Pckg;

/////////////////////////////////////////////////////////
// (Basic) Supplementary Services Functionality
/////////////////////////////////////////////////////////

       /**
       Supplementary Service status.
       */
       enum TSupplServiceStatus
              {
              /**
              The supplementary service is currently active and operative.
              */
              EStatusActive,
              /**
              The supplementary service is currently deactivated or
quiescent.
              */
              ENotActive,
              /**
              In GSM/WCDMA mode, the supplementary service is not
provisioned.
              In CDMA mode, this value has no meaning.
              */
              ENotProvisioned,
              /**
              In GSM/WCDMA mode, the supplementary service is not
available in serving network.
              In CDMA mode, the supplementary service is not available in
the phone.
              */
              ENotAvailable,
              /**
              The phone can not determine supplementary service status.
              */
              EUnknown
              };
```

```
	/**
	Call Forwarding conditions.
	*/
	enum TCallForwardingCondition
		{
		/**
		All calls to this subscriber are forwarded.
		*/
		ECallForwardingUnconditional,
		/**
		Calls are forwarded when this subscriber is busy.
		*/
		ECallForwardingBusy,
		/**
		Calls are forwarded when this subscriber does not reply
within a timeout period.
		*/
		ECallForwardingNoReply,
		/**
		Calls are forwarded when this subscriber is unreachable.
		*/
		ECallForwardingNotReachable
		};

	class TCallForwardingSupplServicesV1 : public TEtelISVType
	/**
	Defines information about the call forwarding service.

    Note: This information is only available on GSM/WCDMA networks.
	*/
		{
	public:
		IMPORT_C TCallForwardingSupplServicesV1();
	public:
		/**
		The status of the call forwarding supplementary service.
		It applies to the condition in iCallForwardingCondition and
		to the line specified to
CTelephony::GetCallForwardingStatus().
		*/
		TSupplServiceStatus iCallForwarding;
		/**
		The provisioned call forwarding condition to which
iCallForwarding applies.
		*/
		TCallForwardingCondition iCallForwardingCondition;
		};

	/**
	A typedef'd packaged CTelephony::TCallForwardingSupplServicesV1
for passing through a generic API method.
	*/
	typedef TPckg<TCallForwardingSupplServicesV1>
TCallForwardingSupplServicesV1Pckg;


	/**
	Call Barring conditions.
	*/
	enum TCallBarringCondition
		{
		/**
		All incoming calls to this subscriber are barred (BAIC).
		*/
		EBarAllIncoming,
```

```
            /**
            All incoming calls to this subscriber are barred when
roaming outside the home PLMN country (BAIC-roam).
            */
            EBarIncomingRoaming,
            /**
            All outgoing calls by this subscriber are barred (BAOC).
            */
            EBarAllOutgoing,
            /**
            All outgoing international calls by this subscriber are
barred (BOIC).
            */
            EBarOutgoingInternational,
            /**
            All outgoing international calls except those directed to
the home PLMN country by this subscriber are barred (BOIC-ExHC).
            */
            EBarOutgoingInternationalExHC
            };

      class TCallBarringSupplServicesV1 : public TEtelISVType
      /**
      Defines information about the call barring service.
      */
            {
      public:
            IMPORT_C TCallBarringSupplServicesV1();
      public:
            /**
            The status of the call barring supplementary service.
            It applies to the condition in iCallBarringCondition and
            to the service group specified to
CTelephony::GetCallBarringStatus().
            */
            TSupplServiceStatus iCallBarring;
            /**
          The provisioned call barring condition to which iCallBarring
applies.
            */
            TCallBarringCondition iCallBarringCondition;
            };

      /**
      A typedef'd packaged CTelephony::TCallBarringSupplServicesV1 for
passing through a generic API method.
      */
      typedef TPckg<TCallBarringSupplServicesV1>
TCallBarringSupplServicesV1Pckg;

      class TCallWaitingSupplServicesV1 : public TEtelISVType
      /**
      Defines information about the call waiting service.
      */
            {
      public:
            IMPORT_C TCallWaitingSupplServicesV1();
      public:
            /**
            The status of the call waiting supplementary service
            for the service group specified to
CTelephony::GetCallWaitingStatus().
            */
            TSupplServiceStatus iCallWaiting;
            };
```

```
	/**
	A typedef'd packaged CTelephony::TCallWaitingSupplServicesV1 for
passing through a generic API method.
	*/
	typedef TPckg<TCallWaitingSupplServicesV1>
TCallWaitingSupplServicesV1Pckg;

	/**
	Phone ID services.
	*/
	enum TIdentityService
		{
		/**
		The identity service is unspecified.
		*/
		EIdServiceUnspecified,
		/**
		The caller's ID is presented to the called party.
		*/
		EIdServiceCallerPresentation,
		/**
		The caller's ID is restricted to the called party.
		*/
		EIdServiceCallerRestriction
		};

	/**
	Service status of the Phone ID services.
	*/
	enum TIdentityServiceStatus
		{
		/**
		The interrogated identity service is provisioned and
permanently active.
		*/
		EIdServiceActivePermanent,
		/**
		The interrogated identity service is provisioned and active.
		By default, the number is restricted unless overridden by
the user.
		*/
		EIdServiceActiveDefaultRestricted,
		/**
		The interrogated identity service is provisioned and active.
		By default, the number is displayed unless specifically
restricted by the user.
		*/
		EIdServiceActiveDefaultAllowed,
		/**
		The interrogated identity service is not provisioned.
		*/
		EIdServiceNotProvisioned,
		/**
		The status of the interrogated Identity service is unknown.
		*/
		EIdServiceUnknown
		};

	class TIdentityServiceV1 : public TEtelISVType
	/**
	Defines the call identity service status.

    Note: This information is only available on GSM/WCDMA networks.
	*/
```

```
            {
        public:
            IMPORT_C TIdentityServiceV1();
        public:
            /**
            The status of the call identity service.
            */
            TIdentityServiceStatus iIdentityStatus;
            };

        /**
        A typedef'd packaged CTelephony::TIdentityServiceV1 for passing
through a generic API method.
        */
        typedef TPckg<TIdentityServiceV1> TIdentityServiceV1Pckg;

        /**
        Service group identifier.
        */
        enum TServiceGroup
            {
            /**
            Voice service group
            */
            EVoiceService,
            /**
            Fax service group
            */
            EFaxService,
            /**
            Data service group
            */
            EDataService
            };

        /**
        Cancel apending request
        */
        enum TCancellationRequest
            {
            /**
            Cancel a pending GetPhoneId request.
            */
            EGetPhoneIdCancel,                              // 0
            /**
            Cancel a pending GetSubscriberId request.
            */
            EGetSubscriberIdCancel,                         // 1
            /**
            Cancel a pending GetFlightMode request.
            */
            EGetFlightModeCancel,                           // 2
            /**
            Cancel a pending FlightModeChange request.
            */
            EFlightModeChangeCancel,                        // 3
            /**
            Cancel a pending GetIndicator request.
            */
            EGetIndicatorCancel,                            // 4
            /**
            Cancel a pending IndicatorChange request.
            */
            EIndicatorChangeCancel,                         // 5
            /**
```

```
Cancel a pending GetBatteryInfo request.
*/
EGetBatteryInfoCancel,                          // 6
/**
Cancel a pending BatteryInfoChange request.
*/
EBatteryInfoChangeCancel,                       // 7
/**
Cancel a pending GetSignalStrength request.
*/
EGetSignalStrengthCancel,                       // 8
/**
Cancel a pending SignalStrengthChange request.
*/
ESignalStrengthChangeCancel,                    // 9
/**
Cancel a pending GetLockInfo request.
*/
EGetLockInfoCancel,                             // 10
/**
Cancel a pending Pin 1 LockInfoChange request.
*/
EPin1LockInfoChangeCancel,                      // 11
/**
Cancel a pending Pin 2 LockInfoChange request.
*/
EPin2LockInfoChangeCancel,                      // 12
/**
Cancel a pending VoiceLineStatusChange request.
*/
EVoiceLineStatusChangeCancel,                   // 13
/**
Cancel a pending Call1StatusChange request.
*/
EOwnedCall1StatusChangeCancel,                  // 14
/**
Cancel a pending Call2StatusChange request.
*/
EOwnedCall2StatusChangeCancel,                  // 15
/**
Cancel a pending OwnedCall1RemotePartyInfoChange request.
*/
EOwnedCall1RemotePartyInfoChangeCancel,         // 16
/**
Cancel a pending OwnedCall2RemotePartyInfoChange request.
*/
EOwnedCall2RemotePartyInfoChangeCancel,         // 17
/**
Cancel a pending SendDTMFTones request.
*/
ESendDTMFTonesCancel,                           // 18
/**
Cancel a pending DialNewCall request.
*/
EDialNewCallCancel,                             // 19
/**
Cancel a pending Hold request.
*/
EHoldCancel,                                    // 20
/**
Cancel a pending Resume request.
*/
EResumeCancel,                                  // 21
/**
Cancel a pending Swap request.
```

```
                */
                ESwapCancel,                                    // 22
                /**
                Cancel a pending Hangup request.
                */
                EHangupCancel,                                  // 23
                /**
                Cancel a pending AnswerIncomingCall request.
                */
                EAnswerIncomingCallCancel,                      // 24
                /**
                Cancel a pending GetNetworkRegistrationStatus request.
                */
                EGetNetworkRegistrationStatusCancel,            // 25
                /**
                Cancel a pending NetworkRegistrationStatusChange request.
                */
                ENetworkRegistrationStatusChangeCancel,         // 26
                /**
                Cancel a pending GetCurrentNetworkInfo request.
                */
                EGetCurrentNetworkInfoCancel,                   // 27
                /**
                Cancel a pending CurrentNetworkInfoChange request.
                */
                ECurrentNetworkInfoChangeCancel,                // 28
                /**
                Cancel a pending GetCurrentNetworkName request.
                */
                EGetCurrentNetworkNameCancel,                   // 29
                /**
                Cancel a pending GetOperatorName request.
                */
                EGetOperatorNameCancel,                         // 30
                /**
                Cancel a pending GetCallForwardingStatus request.
                */
                EGetCallForwardingStatusCancel,                 // 31
                /**
                Cancel a pending GetCallBarringStatus request.
                */
                EGetCallBarringStatusCancel,                    // 32
                /**
                Cancel a pending GetCallWaitingStatus request.
                */
                EGetCallWaitingStatusCancel,                    // 33
                /**
                Cancel a pending GetIdentityServiceStatus request.
                */
                EGetIdentityServiceStatusCancel,                // 34
                /**
                Cancel a pending FaxLineStatusChange request.
                */
                EFaxLineStatusChangeCancel,                     // 35
                /**
                Cancel a pending DataLineStatusChange request.
                */
                EDataLineStatusChangeCancel                     // 36
                };

////////////////////////////////////////////////////////
// Notification Functionality
////////////////////////////////////////////////////////

        /**
```

```
	Notification events
	*/
	enum TNotificationEvent
		{
		/**
		Register interest for receiving a notification for flight
mode changes.

		 New flight mode returned in a CTelephony::TFlightModeV1Pckg, a
packaged CTelephony::TFlightModeV1.

		Pass CancelAsync() CTelephony::EFlightModeChangeCancel to
cancel.
		*/
		EFlightModeChange,			// 0
		/**
		Register interest for receiving a notification for phone
indicator changes.

		New indicators returned in a CTelephony::TIndicatorV1Pckg, a
packaged CTelephony::TIndicatorV1.

		Pass CancelAsync() CTelephony::EIndicatorChangeCancel to cancel.
		*/
		EIndicatorChange,			// 1
		/**
		Register interest for receiving a notification for Battery
information changes.

		New Battery information returned in a
CTelephony::TBatteryInfoV1Pckg, a packaged CTelephony::TBatteryInfoV1.

		Pass CancelAsync() CTelephony::EBatteryInfoChangeCancel to
cancel.
		*/
		EBatteryInfoChange,			// 2
		/**
		Register interest for receiving a notification for Signal
Strength changes.

		New Signal Strength returned in a
CTelephony::TSignalStrengthV1Pckg, a packaged
CTelephony::TSignalStrengthV1.

		Pass CancelAsync() CTelephony::ESignalStrengthChangeCancel to
cancel.
		*/
		ESignalStrengthChange,			// 3
		/**
		Register interest for receiving a notification for Icc Pin1
Lock Information changes.

		New Icc Pin1 Lock Information returned in a
CTelephony::TIccLockInfoV1Pckg, a packaged CTelephony::TIccLockInfoV1.

		Pass CancelAsync() CTelephony::EPin1LockInfoChangeCancel to
cancel.
		*/
		EPin1LockInfoChange,			// 4
		/**
		Register interest for receiving a notification for Icc Pin2
Lock Information changes.

		New Icc Pin2 Lock Information returned in a
CTelephony::TIccLockInfoV1Pckg, a packaged CTelephony::TIccLockInfoV1.
```

```
        Pass CancelAsync() CTelephony::EPin2LockInfoChangeCancel to
cancel.
            */
            EPin2LockInfoChange,                // 5
            /**
            Register interest for receiving a notification for Voice
Line Status changes.

            New Voice Line Status returned in a
CTelephony::TCallStatusV1Pckg, a packaged CTelephony::TCallStatusV1.

        Pass CancelAsync() CTelephony::EVoiceLineStatusChangeCancel to
cancel.
            */
            EVoiceLineStatusChange,             // 6
            /**
            Register interest for receiving a notification for owned
Call 1 Status changes.

            New Call 1 Status returned in a
CTelephony::TCallStatusV1Pckg, a packaged CTelephony::TCallStatusV1.

        Pass CancelAsync() CTelephony::EOwnedCall1StatusChangeCancel to
cancel.
            */
            EOwnedCall1StatusChange,            // 7
            /**
            Register interest for receiving a notification for owned
Call 2 Status changes.

            New Call 2 Status returned in a
CTelephony::TCallStatusV1Pckg, a packaged CTelephony::TCallStatusV1.

        Pass CancelAsync() CTelephony::EOwnedCall2StatusChangeCancel to
cancel.
            */
            EOwnedCall2StatusChange,            // 8
            /**
            Register interest for receiving a notification for Call 1
Remote Party Info changes.

            New Call 1 Remote Party Info returned in a
CTelephony::TRemotePartyInfoV1Pckg, a packaged
CTelephony::TRemotePartyInfoV1.

        Pass CancelAsync()
CTelephony::EOwnedCall1RemotePartyInfoChangeCancel to cancel.
            */
            EOwnedCall1RemotePartyInfoChange,   // 9
            /**
            Register interest for receiving a notification for Call 2
Remote Party Info changes.

            New Call 2 Remote Party Info returned in a
CTelephony::TRemotePartyInfoV1Pckg, a packaged
CTelephony::TRemotePartyInfoV1.

        Pass CancelAsync()
CTelephony::EOwnedCall2RemotePartyInfoChangeCancel to cancel.
            */
            EOwnedCall2RemotePartyInfoChange,   // 10
            /**
            Register interest for receiving a notification for Network
registration status changes.
```

```
            New Network registration status returned in a
CTelephony::TNetworkRegistrationV1Pckg, a packaged
CTelephony::TNetworkRegistrationV1.

        Pass CancelAsync()
CTelephony::ENetworkRegistrationStatusChangeCancel to cancel.
            */
            ENetworkRegistrationStatusChange,    // 11
            /**
            Register interest for receiving a notification for Network
information changes.

            New Network information returned in a
CTelephony::TNetworkInfoV1Pckg, a packaged CTelephony::TNetworkInfoV1.

        Pass CancelAsync() CTelephony::ECurrentNetworkInfoChangeCancel to
cancel.
            */
            ECurrentNetworkInfoChange,          // 12
            /**
            Register interest for receiving a notification for Fax Line
Status changes.

            New Fax Line Status returned in a
CTelephony::TCallStatusV1Pckg, a packaged CTelephony::TCallStatusV1.

        Pass CancelAsync() CTelephony::EFaxLineStatusChangeCancel to
cancel.
            */
            EFaxLineStatusChange,               // 13
            /**
            Register interest for receiving a notification for Data Line
Status changes.

            New Data Line Status returned in a
CTelephony::TCallStatusV1Pckg, a packaged CTelephony::TCallStatusV1.

        Pass CancelAsync() CTelephony::EDataLineStatusChangeCancel to
cancel.
            */
            EDataLineStatusChange               // 14
            };

public:
      // Constructors
      IMPORT_C static CTelephony* NewLC();
      IMPORT_C static CTelephony* NewL();

      // Destructor - virtual and class not intended
      // for derivation, so not exported

      ~CTelephony();

      // General Functionality
      inline TVersion Version() const;

      // Phone Functionality
      IMPORT_C void GetPhoneId(TRequestStatus& aReqStatus, TDes8& aId)
const;
      IMPORT_C void GetSubscriberId(TRequestStatus& aReqStatus, TDes8&
aId) const;
      IMPORT_C void GetFlightMode(TRequestStatus& aReqStatus, TDes8&
aMode) const;
```

```
	IMPORT_C void GetIndicator(TRequestStatus& aReqStatus, TDes8&
aIndicator) const;
	IMPORT_C void GetBatteryInfo(TRequestStatus& aReqStatus, TDes8&
aBatteryInfo) const;
	IMPORT_C void GetSignalStrength(TRequestStatus& aReqStatus, TDes8&
aSignalStrength) const;
	IMPORT_C void GetLockInfo(TRequestStatus& aReqStatus, const
TIccLock& aLock, TDes8& aLockInfo) const;
	IMPORT_C void SendDTMFTones(TRequestStatus& aReqStatus, const
TDesC& aTones) const;
	// Line Functionality
	IMPORT_C TInt GetLineStatus(const TPhoneLine& aLine, TDes8&
aStatus) const;
	IMPORT_C TInt GetCallInfo(TDes8& aCallSelect, TDes8& aCallInfo,
TDes8& aRemoteInfo) const;
	// Call Functionality
	IMPORT_C void DialNewCall(TRequestStatus& aStatus, TDes8&
aCallParams,
		const TTelNumber& aTelNumber, TCallId& aCallId, const
TPhoneLine aLine=EVoiceLine) const;
	IMPORT_C TInt GetCallDynamicCaps(const TCallId& aCallId, TDes8&
aCallCaps) const;
	IMPORT_C TInt GetCallStatus(const TCallId& aCallId, TDes8&
aCallStatus) const;
	IMPORT_C void Hold(TRequestStatus& aReqStatus, const TCallId&
aCallId) const;
	IMPORT_C void Resume(TRequestStatus& aReqStatus, const TCallId&
aCallId) const;
	IMPORT_C void Swap(TRequestStatus& aReqStatus, const TCallId&
aCallId1,
		const TCallId& aCallId2) const;
	IMPORT_C void Hangup(TRequestStatus& aReqStatus, const TCallId&
aCallId) const;
	IMPORT_C void AnswerIncomingCall(TRequestStatus& aReqStatus,
TCallId& aCallId,
		const TPhoneLine aLine=EVoiceLine) const;
	// Network Functionality
	IMPORT_C void GetNetworkRegistrationStatus(TRequestStatus&
aReqStatus, TDes8& aStatus) const;
	IMPORT_C void GetCurrentNetworkInfo(TRequestStatus& aReqStatus,
TDes8& aNetworkInfo) const;
	IMPORT_C void GetCurrentNetworkName(TRequestStatus& aReqStatus,
TDes8& aNetworkName) const;
	IMPORT_C void GetOperatorName(TRequestStatus& aReqStatus, TDes8&
aOperator) const;
	// (Basic) Supplementary Services Functionality
	IMPORT_C void GetCallForwardingStatus(TRequestStatus&
aRequestStatus, const	TCallForwardingCondition aCondition, TDes8&
aSSInfo, const TServiceGroup aServiceGroup=EVoiceService) const;
	IMPORT_C void GetCallBarringStatus(TRequestStatus& aRequestStatus,
const TCallBarringCondition aCondition, TDes8& aSSInfo, const
TServiceGroup aServiceGroup=EVoiceService) const;
	IMPORT_C void GetCallWaitingStatus(TRequestStatus& aRequestStatus,
TDes8& aSSInfo, const TServiceGroup aServiceGroup=EVoiceService) const;
	IMPORT_C void GetIdentityServiceStatus(TRequestStatus& aReqStatus,
const TIdentityService& aService, TDes8& aStatus) const;
	// Cancel Request Functionality
	IMPORT_C TInt CancelAsync(TCancellationRequest aCancel) const;
	// Notification Functionality
	IMPORT_C void NotifyChange(TRequestStatus& aReqStatus, const
TNotificationEvent& aEvent, TDes8& aDes) const;

	IMPORT_C TInt FindDeviceStatus(TInt& aCallStatus);
	IMPORT_C TInt EstablishDataCall(TInt& aDataChannel, const TDesC&
aDialNumber);
```

```
	IMPORT_C void EstablishDataCall(TRequestStatus& aRequestStatus,
TInt& aDataChannel, const TDesC& aDialNumber);
	IMPORT_C TInt TerminateDataCall();
	IMPORT_C TInt ReceiveDataCall(TInt& aDataChannel);
	IMPORT_C void ReceiveDataCall(TRequestStatus& aRequestStatus,
TInt& aDataChannel);
	IMPORT_C void CancelEstablishDataCall();
	IMPORT_C void CancelReceiveDataCall();

private:
	CTelephony();// C++ constructor - not exported. Called from
NewLC()
	void ConstructL();  // second-phase constructor
	CTelephonyFunctions*	iTelephonyFunctions;	///< Object
to do the real work
	};

#include "telephony.inl"
// ETEL3RDPARTY_H__
#endif
```