Practical Symbian Development

Fabrizio Sannicolò e Marco Cova

fabrizio.sannicolo@dit.unitn.it, cova@itc.it

Università degli Studi di Trento

Facoltà di Scienze MM.FF.NN.

Modified by Marco Aiello

FOR INTERNAL USE ONLY

Series 60 Platform

- ***Series 60 Platform is a complete smartphone reference design***
- ***It represents a rich environment for delvelopers to create innovative applications utilizing technologies such as WAP, Java, C++, Bluetooth***
- ***The platform builds on the Symbian operating system***
- ***Central to the success of Series 60 Platform is Symbian OS 6.1: it is the foundation of the product***
- ***Symbian OS 6.1 a 32-bit multitasking operating system, wherein events happen asynchronously and applications are designed to interact with one other***

Series 60 SDK for Symbian OS

- *The series 60 SDK for Symbian OS targets Series 60 phones*
- *The main programming language for development is C++*
- *Symbian OS is largely written in C++*
- *A set of robust components and many varied APIs are provided in Developer Platform for Series 60*
- *The range of Series 60 devices includes:*
  - *Nokia Mobile Phones*
  - *Siemens*
  - *Samsung*
  - *Matsushita*
  - *Sendo*

Contents of the Series 60 SDK for Symbian OS

- ***Build tools, instructions and environment***
- ***Debug and release emulator environment***
  (`C:\Symbian\Series60_1_2_B\epoc32\release\winsb\[udeb|urel]\`)
- ***Integrated documentation. Combinated Symbian & Series 60 documentation***
- ***Java 2 runtime environment***
- ***Active Perl 518***
- ***Series 60 Application Wizard***
- ***Code example***

Contents of the Series 60 SDK for Symbian OS

Figure 1 depicts the directory structure of the Series 60 SDK for Symbian OS. Series 60 SDK for Symbian OS is installed under Symbian Directory by default.

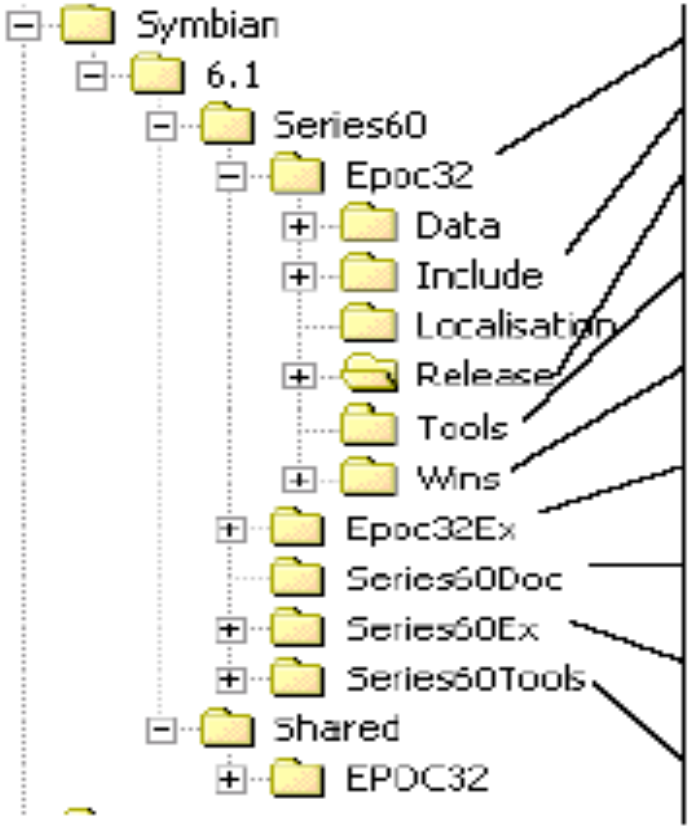
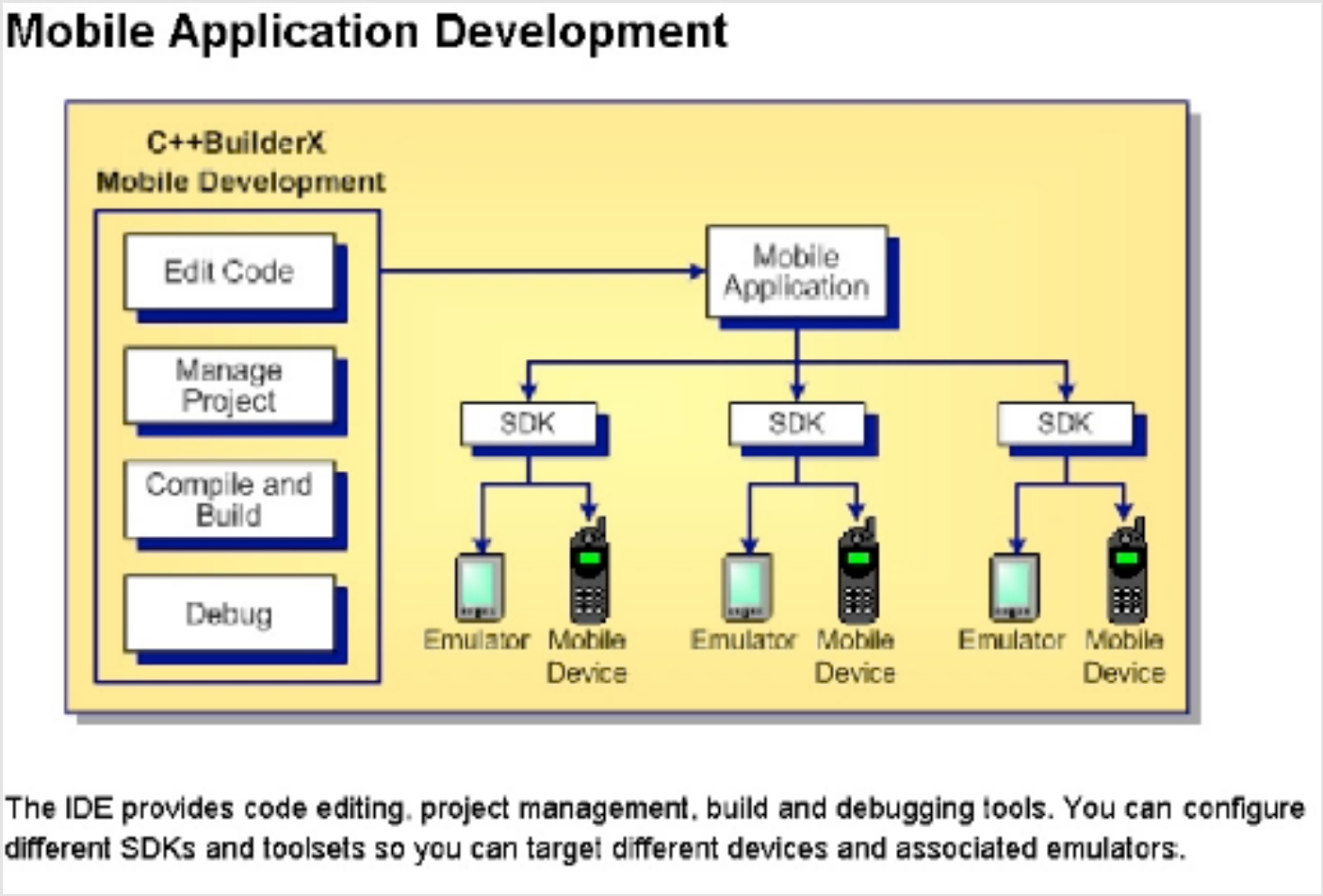


Figure 1. Directory structure of the SDK

Introducing Mobile Development

- *The Symbian OS build tools are actually command line-based and would work without any IDE*
- *BTW development tools, there are several choices*
  - *Microsoft Visual C++ 6*
  - *Microsoft Visual Studio .net*
  - *Eclipse*
  - *Borland C++ BuilderX Mobile Edition*
- *We refer to Borland suite because it is free and it takes away much of the*

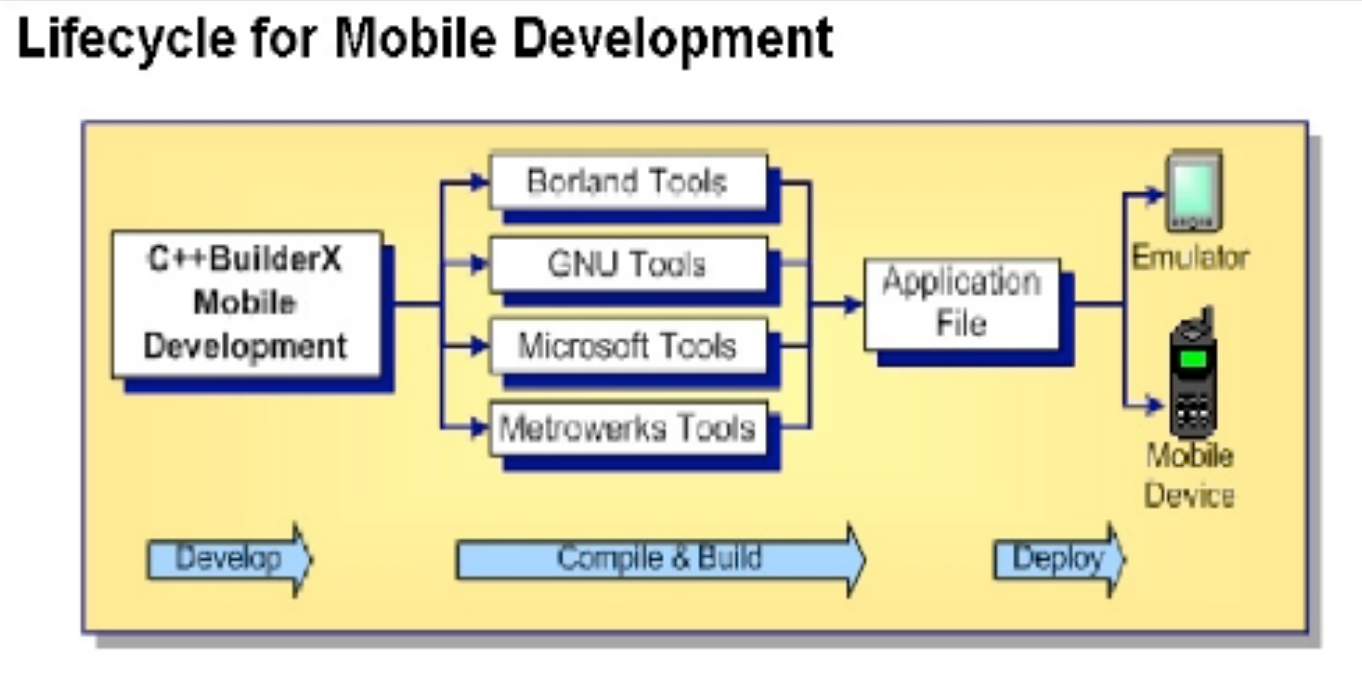
Mobile Application Development

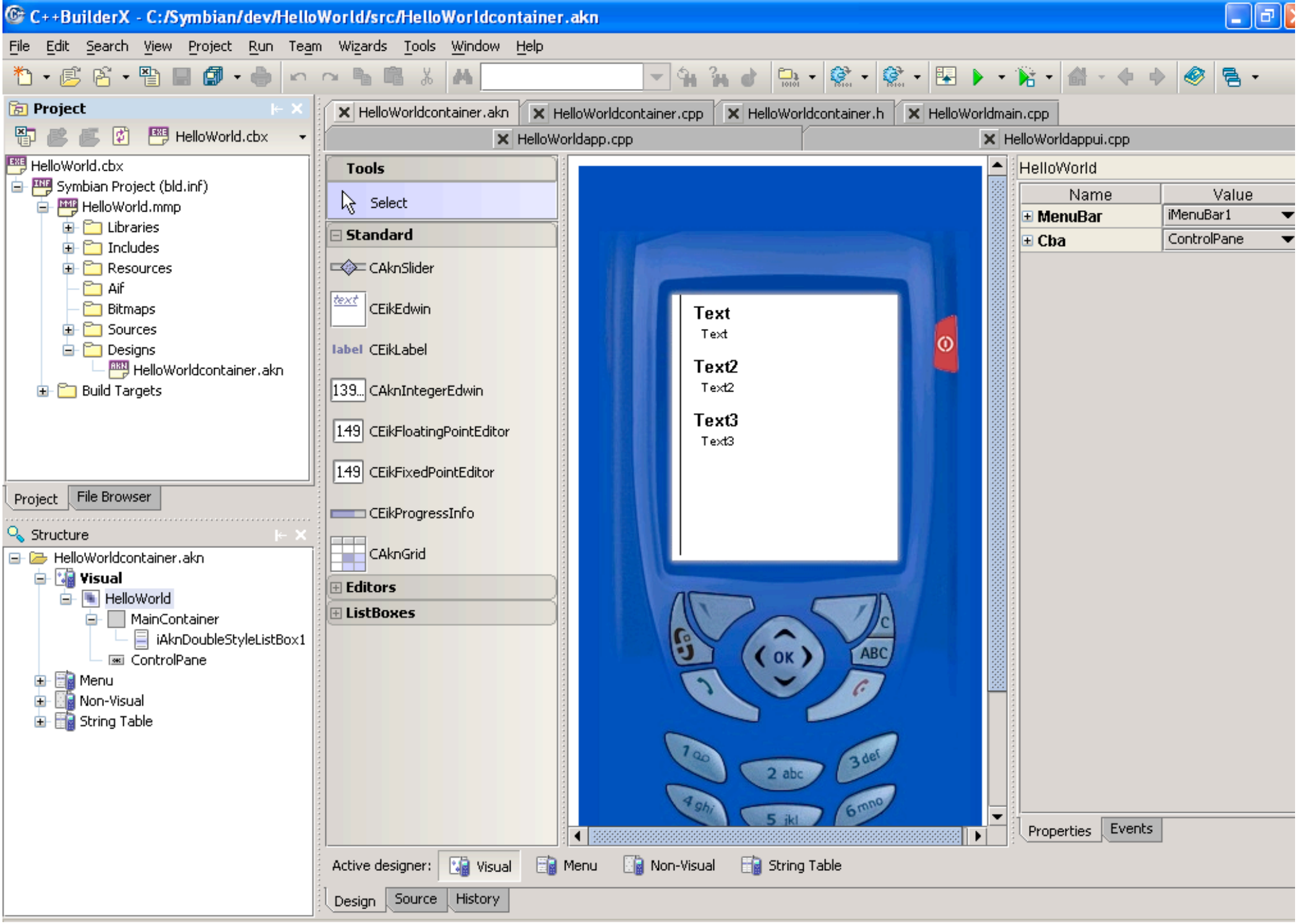


Mobile Application Development Lifecycle

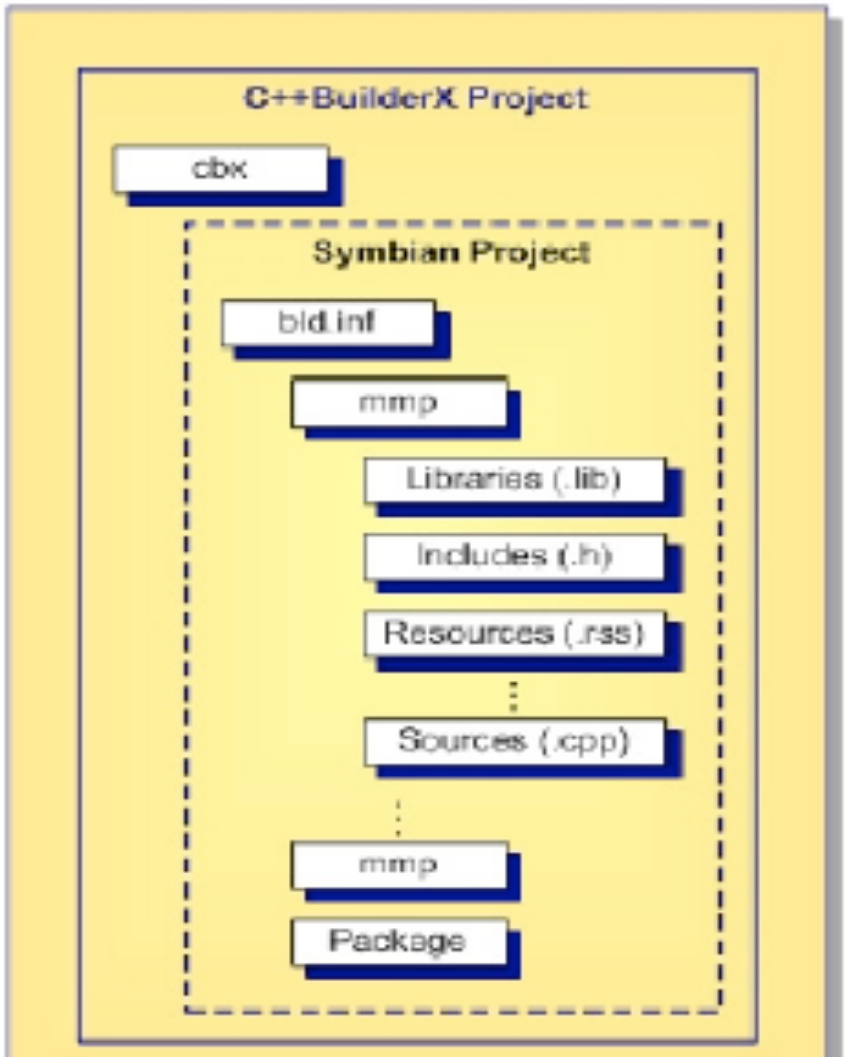
- ***The working environment for creating mobile applications is the same as for normal C++ development in C++BuilderX***
- ***The development environment is flexible because it supports programming using different SDKs***
- ***It lets you compile with different compilers, debug with debuggers from different toolsets, and deploy to different emulators or mobile devices***

Lifecycle for Mobile Development




C++BuilderX - C:/Symbian/dev/HelloWorld/src/HelloWorldcontainer.akn
File Edit Search View Project Run Team Wizards Tools Window Help
Project
HelloWorld.cbx
HelloWorld.cbx
Symbian Project (bld.inf)
HelloWorld.mmp
Libraries
Includes
Resources
Aif
Bitmaps
Sources
Designs
HelloWorldcontainer.akn
Build Targets
Project File Browser
Structure
HelloWorldcontainer.akn
Visual
HelloWorld
MainContainer
iAknDoubleStyleListBox1
ControlPane
Menu
Non-Visual
String Table
HelloWorldcontainer.akn
HelloWorldcontainer.cpp
HelloWorldcontainer.h
HelloWorldmain.cpp
HelloWorldapp.cpp
HelloWorldappui.cpp
Tools
Select
Standard
CAknSlider
CEikEdwin
CEikLabel
CAknIntegerEdwin
CEikFloatingPointEditor
CEikFixedPointEditor
CEikProgressInfo
CAknGrid
Editors
ListBoxes
Text
Text
Text2
Text2
Text3
Text3
OK
C
ABC
1 qo
2 abc
3 def
4 ghi
5 jkl
6 mno
HelloWorld
Name
Value
MenuBar
iMenuBar1
Cba
ControlPane
Properties
Events
Active designer:
Visual
Menu
Non-Visual
String Table
Design
Source
History

Main Project structure

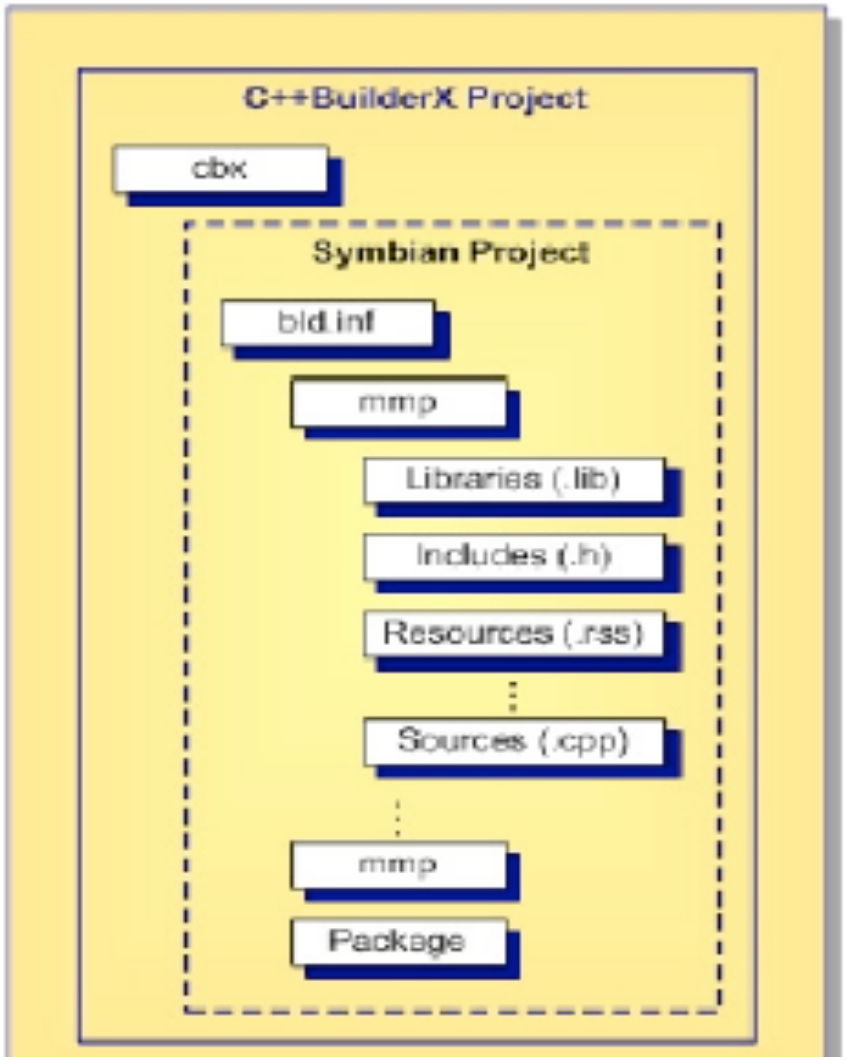


According to the Builder products any Series 60 project defines several files and folders

Several project files are used for building and deploying Symbian applications to the mobile device or simulator

In addition to the .cbx files used by BuilderX product to store own information, we need to define Symbian OS files adopted for building the final application

Main Project structure



Thw .cbx file is a C++BuilderX project file that organizes the mobile project

The bld.inf file is the Symbian project file that specifies one or more .mmp files and a package

Each project must have one bld.inf that lists all of the targets contained in the project

It is used by bldmake to create the makefiles to build the app. It

Main Project structure

```
TARGET          HelloWorld.app
TARGETTYPE      app
UID             0x100039CE 0x1000005
TARGETPATH      \system\apps\HelloWorld

LANG 01

SOURCEPATH      ..\src
SOURCE          HelloWorldmain.cpp
SOURCE          HelloWorldapp.cpp
SOURCE          HelloWorlddocument.cpp
SOURCE          HelloWorldappui.cpp
SOURCE          HelloWorldview.cpp
SOURCE          HelloWorldcontainer.cpp

RESOURCE        HelloWorld.rss

USERINCLUDE     ..\inc

SYSTEMINCLUDE   \epoc32\include
SYSTEMINCLUDE \epoc32\include\borland

LIBRARY         euser.lib
```

- *An `.mmp` file defines a mobile target specifying the properties of a project in a platform and compiler-independent way*
- *The information in this file is converted into a makefile for the project*
- *The package description file, `.pkg`, is a text file that contains installation information for an application*

Preparation – step 0

Download the following components:

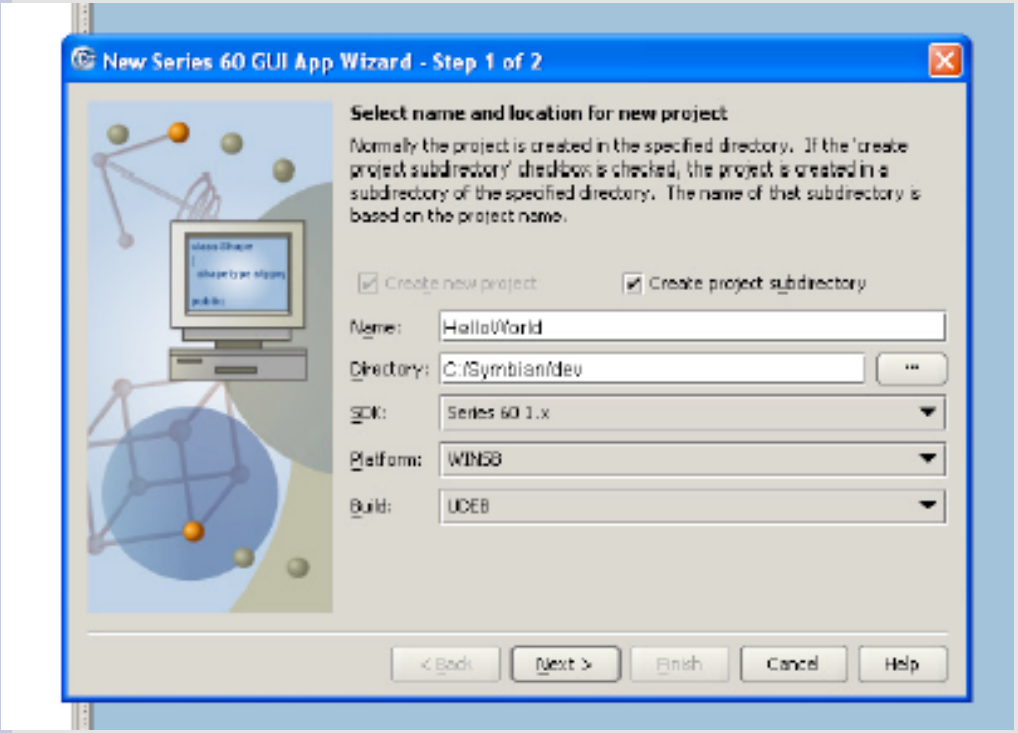
- ***ActivePerl - http://www.activetest.com/***
- ***Symbian OS SDK – we refer to the Series 60 SDK 1.2 for Symbian OS, Nokia Edition, http://www.symbian.com/developer/sdks_series60.asp***
- ***Microsoft Debugging Tools - http://www.microsoft.com/whdc/devtools/debuggin/***
- ***Visual C++ Toolkit - http://www.microsoft.com/downloads/***
- ***Borland C++ BuilderX Mobile Edition (v1.5) – get it for free by filling out the short survey, http://info.borland.com/survey/cbx15_mobile_edition.html***

Note. From http://www.forum.nokia.com/ it is free an overall zip package including all you need (except Borland)

HelloWorld Example – goal

- ***`HelloWorld` C++ example with Series 60 for Symbian OS v. 1.2***
- ***We refer to the EPOC Emulator as shown on the right***
- ***It displays the `HelloWorld` application and other pre-built menus***

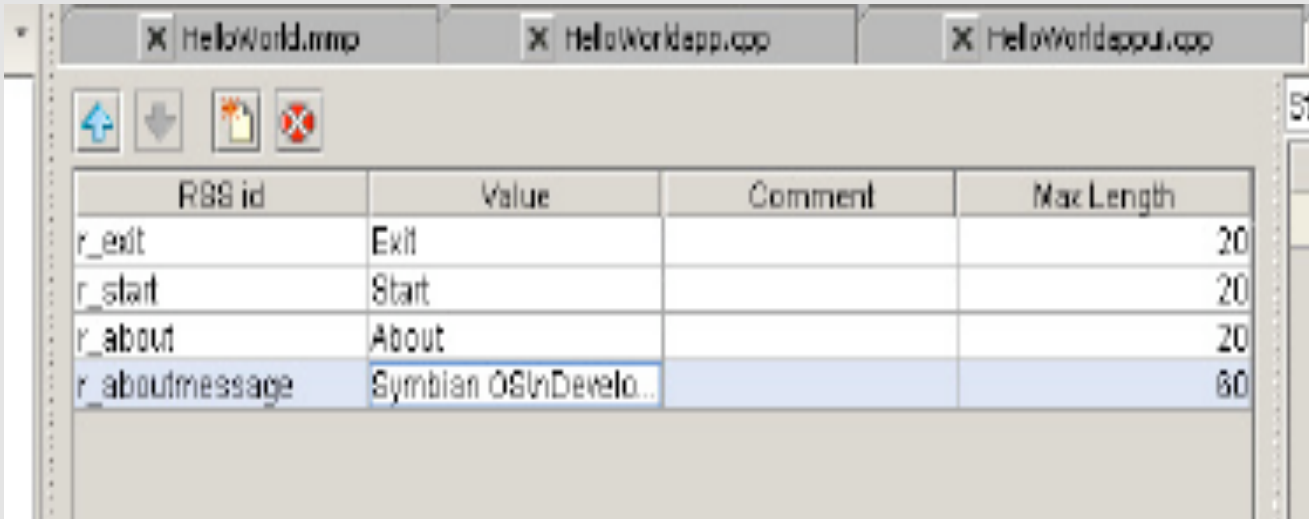
HelloWorld Example – step 1



- Open Borland's C++ BuilderX Mobile Edition
- Let us create a new SDK configuration for Symbian OS under Tools and Symbian SDK Configuration
- Set the path with C:\Symbian\Series60_1_2_B\ and call it Symbian Series 60 1.x
- Create a new project and choose Series 60 and then New Series 60 GUI Application
- Take a look at the snapshot on the left...

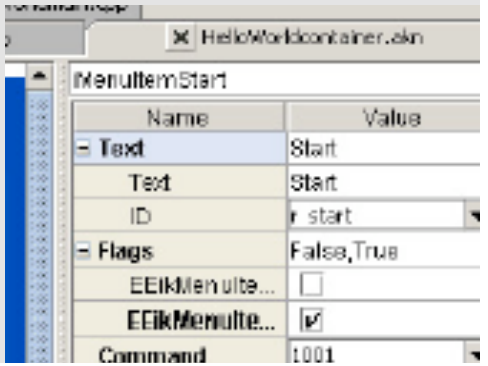
HelloWorld Example – step 2

- ***Next step is to customize our own menu items***
- ***We first have to define the text that we want to display: for doing this we need to define some resource files***
- ***Switch to the `String Table` designer and then add the following lines***



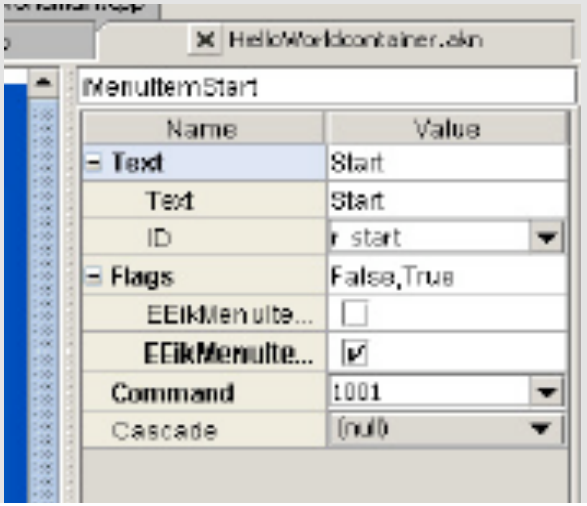
HelloWorld Example – step 3

- ***Switch to the `Menu` designer and select the menu item tool on the left panel and click into the menu to add a new item***
- ***On the right panel select `iMenuItem1` and change its name in `iMenuItemStart`***
- ***Customize the other menu in the same way...***



HelloWorld Example – step 4

- *Let us attach an event to `Start` menu*
- *Switch to `Non-Visual` menu and add `CaknInformationNote` (it is a small window that will contain text and predefined icon) calling it `Start` and select as ID `r_start` on the right*
- *Go back to menu and click on `Start`: on the right, under `Events`, double click in the empty text box next to `OnViewCommand`*
- *C++ BuilderX automatically creates the function `OniMenuItemStartViewCommand()`*
- *Basically, add `L` at the end and write in `ExecuteiAknInformationNote1L();`*



HelloWorld Example – step 5

- *Normally the emulator works fine for debugging*
- *When you have prepared your mobile application and you want to move it to the mobile phone, you must perform the following steps:*
  - *Switch the target platform of the compilation process to ARMI (the name of the Series 60 phone is ARM)*
  - *Switch the type to UREL: it means Release Version against UDEB, debugging version*
  - *Build the project means to produce a file with extension `.sis`*
  - *Now, you are ready to transfer the `.sis` file to your phone and install it via e.g. IrDA or Bluetooth*

Leaves caveats (1)

- Safe:

```
void FunctionMayLeaveL() {
  CTestClass* aObject = CTestClass::NewL();
  aObject->FunctionDoesNotLeave();
  delete aObject;
}
```

- Unsafe:

```
void UnsafeFunctionL() {
  CTestClass* aObject = CTestClass::NewL();
  aObject->FunctionMayLeaveL();
  delete aObject;
}
```

Leaving caveats (2)

- Safe:

```
void CTestClass::SafeFunctionL() {
   // member variable
   iMember = COtherClass::NewL();
   FunctionMayLeaveL();
}
```

- Safe:

```
void AnotherSafeFunctionL() {
   TInt locInt = 1; // leave-safe (built-in)
   FunctionMayLeaveL(locInt);
   TMyClass locObject(locInt); // leave safe: T
   AnotherFunctionMayLeaveL(locObject);
}
```

Cleanup stack: API

- Push object on the stack:

```
static void PushL(TAny* aPtr);
static void PushL(CBase* aPtr);
static void PushL(TCleanupItem anItem);
```

- Pop objects from the stack

```
static void Pop();
static void Pop(TInt aCount);
```

- Pop and destroy objects

```
static void PopAndDestroy();

static void PopAndDestroy(TInt aCount);
```

Cleanup stack and two-phase construction

```
CExample* CExample::NewLC() {
 // first phase
   CExample* me = new (ELeave) CExample();
   CleanupStack::PushL(me);
 // second phase
   me->ConstructL();
   return me;
}

CExample* CExample::NewL() {
   CExample* me = Cexample::NewLC();
   CleanupStack::Pop(me);
   return me;
}
```

Cleanup caveats

- Attention to exit points!

```
void functionL() {
  CExample* aObject = CExample::NewL();
  CleanupStack::PushL(aObject);
  TBool result = NonLeavingFunction();
   if (! result)
     return; // Ooops: aObject still on the stack

   LeavingFunctionL();
   CleanupStack::PopAndDestroy(aObject);
}
```

Example

- As an example, we will build an active object `CDelayedHelloWorld,` whose request function sets a timer. When the timer expires, the active object posts a message saying “Hello, world!”
- Asynchronous request method: `SetDelay(TTimeIntervalMicroSeconds32)` requests the message post
- Service provider here is simulated through the use of `RTimer`
- This mimicks more realistic situations, e.g., retrieval of files from the net, file I/O, etc.

Example: CDelayedHelloWorld

```
class CDelayedHelloWorld : public CActive
{
  public:
    CDelayedHelloWorld();
    void ConstructL(CEikonEnv* aEikEnv);
    ~CDelayedHelloWorld();
    void SetDelay(TTimeIntervalMicroSeconds32
   aDelay)
  private:
    virtual void DoCancel();
    virtual void RunL();
  private:
    CEikonEnv* iEikEnv; // our environment
    RTimer iTimer;
}
```

Example: construct active object (two-phase construction)

```
// first phase
CDelayedHelloWorld::CDelayedHelloWorld() :
  CActive(0) // standard priority
{}

// second phase
void
  CDelayedHelloWorld::ConstructL(CEikonEnv*
  aEikEnv)
{
  iEikEnv = aEikEnv;
  User::LeaveIfError(iTimer.CreateLocal());
  // add to scheduler
  CActiveScheduler::Add(this);
}
```

Example: request service

```
void
CDelayedHelloWorld::
SetDelay(TTimeIntervalMicroSeconds32 aDelay)
{
  // cancel previous requests
  Cancel();
  /* request the service to the service
provider */
  iTimer.After(iStatus, aDelay);
  // mark this active object as active
  SetActive();
}
```

Example: handling completion

```
/* handles completion of asynchronous service
  request */
void CDelayedHelloWorld::RunL()
{
  iEikEnv->InfoMsg(_L("Hello, world!"));
}
```

Example: cancellation (1)

```
/* CActive wrapper: provides basic framework
   for correct cancellation of asynchronous
   requests */
EXPORT_C void CActive::Cancel()
{
   if ( ! iActive )
      return;
   // let derived classes do the cancellation
   DoCancel();
   // wait for request to complete
   User::WaitForRequest(iStatus);
   iActive = EFalse;
}
```

Example: cancellation (2)

```
// actual cancellation
void CDelayedHelloWorld::DoCancel()
{
  iTimer.Cancel();
}
```

Example: destroying

```
CDelayedHelloWorld::~CDelayedHelloWorld()
{
  // cancel outstanding requests
  Cancel();
}
```

Note:

- The `CActive` destructor removes the object from the active scheduler before being destroyed
- The `CActive` destructor cannot be responsible for calling `Cancel()` because the destruction may require derived-class information not available at the time the base-class

Example: using the active object

```
void CExampleAppUi::ConstructL()
{
  ...
  iHelloWorld = new (ELeave) CDelayedHelloWorld;
  iHelloWorld->ConstructL(iEikonEnv);
}
CExampleAppUi::~CExampleAppUi
{
  delete iHelloWorld;
}

...
iHelloWorld->SetDelay(3000000); // 3 seconds
  delay
...
```

A more complex example

- Active object that performs web page fetching via HTTP requests
- Steps needed to complete a request:
  - Resolve the IP address of a given DNS name
  - Open a socket connection
  - Write HTTP GET request
  - Read response from web server
- Each step is done with asynchronous method calls
- Exposes to clients the `FetchFileL()` method

Construction - destruction (1)

```
CAOExampleEngine* CAOExampleEngine::NewLC()
{
    CAOExampleEngine* result = new (ELeave)
  CAOExampleEngine();
    CleanupStack::PushL( result );
    result->ConstructL();
    return result;
}

CAOExampleEngine::CAOExampleEngine()
: CActive( CActive::EPriorityStandard )
{ }
```

Construction - destruction (2)

```
void CAOExampleEngine::ConstructL() {
    User::LeaveIfError( iSocketServ.Connect() );
    // connect to host resolver server
    User::LeaveIfError(iHostResolver.Open(
  iSocketServ,       KafInet, KProtocolInetUdp ) );
    CActiveScheduler::Add(this); // add to AS queue
}

CAOExampleEngine::~CAOExampleEngine()
{
    if( IsAdded() ) {
        Deque(); // calls also Cancel()
    }
    iHostResolver.Close();
    iSocketServ.Close();
}
```

Request method

```
void CAOExampleEngine::FetchFileL( const TDesC& anURL,
        CBufBase* aBuffer, MEngineObserver* anObserver )
{
    if( iState != ENotConnected ) {
        _LIT( KError, "Engine is already fetching a
  file (it is now cancelled)");
        iObserver->Cancelled(KError);
        Reset();
        User::Leave(EAlreadyFetching);
    }
    iObserver = anObserver;
    iOutputBuffer = aBuffer;

    ParseURLL( anURL, *anObserver, iServerName, iPort,
  iDocument );
    ResolveIP();
}
```

```
void CAOExampleEngine::ParseURLL( const TDesC& anURL,
  MEngineObserver& anObserver, TDes& aServerName,
  TInt& aPort, TDes& aDocument )
{
    /*
       breaks up the URL provided by the user in its
       server name, port and document components.
     */
}
```

Looking up the IP

```
void CAOExampleEngine::ResolveIP()
{
    iState = ELookingUpIP;
    iObserver->ToState( ELookingUpIP );

    /* make an asynchronous call to get the IP of the
  host */
    iHostResolver.GetByName(iServerName, iNameEntry,
  iStatus);
    SetActive();
}
```

Handling service completions (1)

```
void CAOExampleEngine::RunL() {
  switch( iState ) {
      case ELookingUpIP:
          if( iStatus == KErrNone ) {
           /* convert socket address to inet address */
              TInetAddr addr( iNameEntry().iAddr );
              addr.SetPort( iPort );
              Connect( addr );
          }
          else {
              _LIT( KFormat, "Failed to obtain IP for
  \"%S\" (error %d)." );
              iMsgBuf.Format( KFormat, &iServerName,
  iStatus );
              iObserver->Cancelled( iMsgBuf );
              Reset();
          }
      break;
```

Handling service completions (2)

```
    case EConnecting:
    {
            if( iStatus == KErrNone ) {
                MakeHTTPRequest();
            } else
                {
                _LIT( KFormat,
                        "Failed to connect to port %d on
\"%S\" (error %d)." );
                iMsgBuf.Format( KFormat, iPort,
&iServerName, iStatus );
                iObserver->Cancelled( iMsgBuf );
                Reset();
                }
            break;
    }
```

Handling service completions (3)

```
   case EWriting:
          if( iStatus == KErrNone )
              {
              iSocketBuf.Zero(); //reset buffer
(remove http request).
              ReadHTTPResponse(TRUE);
              }
          else
              {
              _LIT( KFormat, "Failed to write HTTP
request (error %d)." );
              iMsgBuf.Format( KFormat, iStatus );
              iObserver->Cancelled( iMsgBuf );
              Reset();
              }
          break;
```

Handling service completions (4)

```
case EReading:
            if( iStatus == KErrNone ) {
                ReadHTTPResponse(TRUE);
            } else if( iStatus == KErrEof ) {
                ReadHTTPResponse(FALSE);
                iObserver->FileFetched( iOutputBuffer );
                Reset();
            } else
                {
                _LIT( KFormat, "Failed to read HTTP
 response (error %d)." );
                iMsgBuf.Format( KFormat, iStatus );
                iObserver->Cancelled( iMsgBuf );
                Reset();
                }
            break;
        }
    }
```

Connecting the socket (asynchronous call)

```
void CAOExampleEngine::Connect(TInetAddr& aInetAddr)
{
    iState = EConnecting;
    iObserver->ToState( EConnecting );
    TInt err = iSocket.Open( iSocketServ, KAfInet,
  KsockStream, KProtocolInetTcp );
    if( err != KErrNone )
        {
        _LIT( KMsg, "Failed to open socket: %d." );
        iMsgBuf.Format( KMsg, err );
        iObserver->Cancelled( iMsgBuf );
        Reset();
        }
        iSocket.Connect(aInetAddr, iStatus);
        SetActive();
}
```

Request document (asynchronous call)

```
void CAOExampleEngine::MakeHTTPRequest()
{
    iState = EWriting;
    iObserver->ToState( EWriting );
    // Construct an HTTP request, e.g. "GET
  /docs/index.html HTTP/1.0\n\n"
    _LIT(K1, "GET ");
    _LIT(K2, " HTTP/1.0\n\n");
    iSocketBuf.Copy( K1 );
    iSocketBuf.Append( iDocument );
    iSocketBuf.Append( K2 );

        iSocket.Write( iSocketBuf, iStatus );
    SetActive();
}
```

Read response (asynchronous call)

```
void CAOExampleEngine::ReadHTTPResponse(TBool
  aContinue) {
    iState = EReading;
    iObserver->ToState( EReading );
   /* append data read from the socket to the end of
  the output buffer. When data finished, iStatus set
  to KErrEof */
    iOutputBuffer->InsertL( iOutputBuffer->Size(),
  iSocketBuf );
    iSocketBuf.Zero();

    if( aContinue ) {
        iSocket.RecvOneOrMore( iSocketBuf, 0, iStatus,
  iStupid );
        SetActive();
    }
}
```

Shutting down

```
void CAOExampleEngine::Reset() {
    switch (iState) {
        case EWriting:
        case EReading:
        case EConnecting:
            iSocket.CancelAll();
            iSocket.Close();
            break;
        case ELookingUpIP:
            iHostResolver.Cancel();
            break;
    }
    iState = ENotConnected;
}
void CAOExampleEngine::DoCancel() {
    Reset();
}
```

Bibliography

- *J. Stichbury, Effective C++ Programming for Smartphones, 2005, Symbian Press*

- *M. P. Tasker, Active Objects, 1999, http://www.symbian.com/developer/techlib/papers/tp_active_objects/active.htm*

- *Symbian OS: Active Objects and the Active Scheduler, 2004, dal sito: http://forum.nokia.com/*