Migrating apps from Delphi 5 to Delphi 2006; some notes on 2007 & 2009

Jeroen Pluimers
better office benelux
jpluimers@better-office.com

Business case:

Nationale Nederlanden
100+ apps in ≈ ½ year

Massive migration

- Overall
  - Delphi 5 —> Delphi 2006
  - BDE —> ADO
  - Conversions must be done 'in place'
- Servers
  - NT4 —> Windows 2003
  - Apps —> Services
  - Single core —> Multi-processor
  - New version of Unisys imaging servers/software
  - Development of service controller
- Clients (end-user and system maintenance)
  - New revision of XP base install
    - Yet another Acrobat Reader version
    - No more BDE
  - System maintenance tool restrictions
    - No remote-logon / event-viewer access on the servers
    - Only database access is allowed

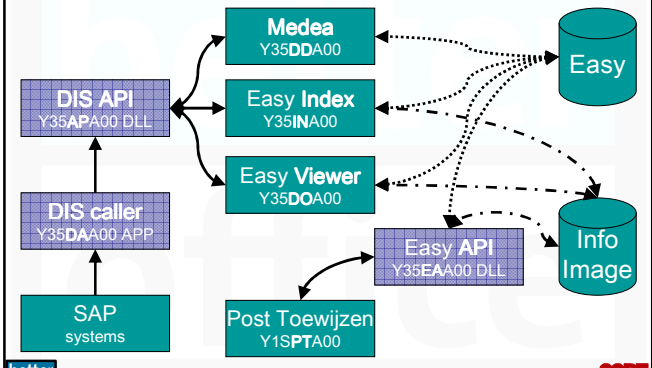
But

- Code should still be as much compatible to Delphi 5 as possible
- Reasons:
  - Some applications outside our scope depend on the library layers and were staying at Delphi 5
  - It enables potential regression testing both in Delphi 5 and Delphi 2006

Overall structure

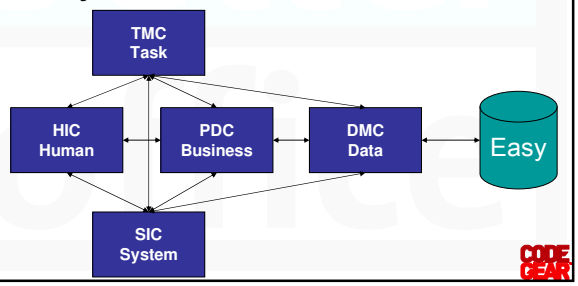
- Delphi RTL
- Delphi VCL
- 3rd party libraries (few – phew)
- NN Library
- EA Library
- Apps

EA application/DB structure



Multi-tier structure at NN

- Hardly any form-designer things
  - Mostly hand-coded



Part 1

Project structure

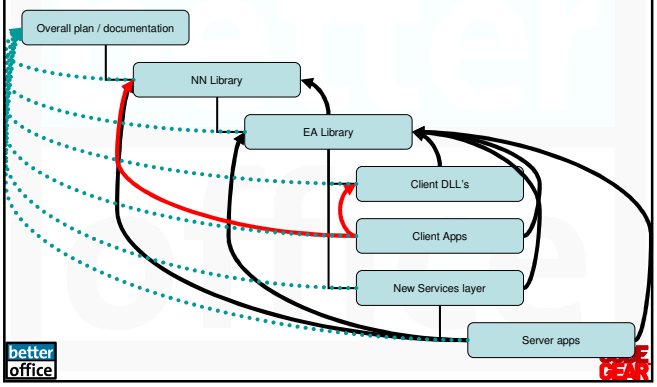
People

- 0,5 manager
- 2,5 part-time
  - Library conversion
  - Overall conversion plan
  - Technically difficult areas
  - Coaching
- 2 full-time
  - Doing grunt-work and basic testing
- X people
  - Test phase

Most important

- Managing the process
  - What to do
  - When to do it
  - Which people
- Managing the technical stuff
  - See later on…

Planning



Part 2

Technical port
Libraries

Biggest problems (1)

- Support both Delphi 5 and Delphi 2006
  - Use conditional defines {$ifdef ...}
- Delphi 2006 compiler is more strict, for instance with pointers

```
procedure CreateDatabase(DBName, User, Pswd: string);
var
  status: array[1..19] of longint;
  StatusVector: PLongint;
begin
//jpl: [Pascal Error] NNCRDBIB.pas(71): E2010
// Incompatible types: 'PLongint' and 'Pointer', dus [1] subscript toegevoegd
  StatusVector := @Status[1];
//isc_dsql_execute_immediate(@status, //...
  isc_dsql_execute_immediate(StatusVector, //...
end;
```


Biggest problems (2)

- Compiler continued...
  - Delphi does the String –> PChar automatically, no @ needed

```
var
  ClassNaam: array[0..255] of char;
begin
//GetClassName(ChildHandle, @ClassNaam, SizeOf(ClassNaam));
  GetClassName(ChildHandle, ClassNaam, SizeOf(ClassNaam));
```

- FastMM finds more bugs
  - Dynamically enable logging
- Consts.pas was changed to translate resourcestrings into Dutch
  - Not compatible with Delphi 2006
  - Solved with dynamic resource substitution (demo!)

Easier issues (1)

- Variants require new Variants unit
- Classes need to implement Interfaces more strictly
  - Use methods/properties from interface in exactly the same way (i.e. use const, var, etc)
- TControl.SetAutoSize introduced in Delphi 7
  - procedure SetAutoSize(Value: Boolean); {$ifdef d7up} override; {$else} virtual; {$endif}

Easier issues (2)

- Some consts moved over to other units:
  - RTLConsts, VDBConsts (Delphi 7+)
- Sometimes, TypeLibrary editor now generates consts parameters:
  - procedure CombineWith(aLijst: IUnknown); safecall;
  - procedure CombineWith(**const** aLijst: IUnknown); safecall;

Easier issues (3)

- Since Delphi 7, MakeObjectInstance is now in Classes.pas
  - In Forms.pas it is now deprecated.
- Since Delphi 7, TModalResult is now in Controls.pas
  - It was in Forms.pas
- Since Delphi 6, DirectoryExists is now in SysUtils.pas
  - so you don't need FileCtrl.pas any more (which is platform dependent)
- Since Delphi 6, TVarData.VError is now a HRESULT

```
{$ifdef d6up}
  TVarData(FEmptyVar).VError := HRESULT($80020004);
    {DISP_E_PARAMNOTFOUND}
{$else}
  TVarData(FEmptyVar).VError := LongWord($80020004);
    {DISP_E_PARAMNOTFOUND}
{$endif}
```


Easier issues (4)

- Since Delphi 2006, a lot of Database things now use WideStrings (new unit)

```
type
{$ifdef d10up}
  TValueList = TWideStringList;
  TAbstractValueList = TWideStrings;
{$else}
  TValueList = TStringList;
  TAbstractValueList = TStrings;
{$endif d10up}
```

Easier issues (5)

- Various Delphi versions introduce new TTypeKind and TOrdType enumerations
  - Only interesting if you do stuff at TField descendent levels (NNPrpFld)
- Various Delphi versions introduce new RTTI level things
  - Only interesting if you do RTTI reflection (NNRTTI)

Easier issues (6)

- Components icons
  - On Tool Palette:
    - 16 x 16 pixels
    - 24 x 24 pixels
    - 32 x 32 pixels
  - On Forms/Datamodules
    - 24 x 24 pixels
- Solution
  - Keep the old 24 x 24 bitmap in your .DRC or RES file
  - Add a new 16 x 16 bitmap with the same name, but with 16 as a suffix
  - Add a new 32 x 32 bitmap with the same name, but with 32 as a suffix
- Example: DCLMID120.bpl
  - TCLIENTDATASET – old 24 x 24 pixels bitmap
  - TCLIENTDATASET16 – new 16 x 16 pixels bitmap
  - TCLIENTDATASET32 – new 32 x 32 pixels bitmap

Time consuming (1)

- Unit filenames must be the same case as the unit names
  - File MyUnit.pas is not compatible with "unit myunit;"
- Separate packages in design-time and run-time
  - DsgnIntf was split up in multiple units in Delphi 6
  - DesignIntf is now in a design-time only package

```
{$ifdef d6up}
  VCLEditors, // TMPFileNameEditor
  DesignIntf, // DsgnIntf renamed to DesignIntf in D6
{$else}
  DsgnIntf,
{$endif d6up}
```

Time consuming (2)

- Delphi 7 introduced new warnings
  - To prepare for .NET
  - Decide where to apply these:
    - {$warn Symbol_Platform off}
    - {$warn Symbol_Deprecated off}
    - {$warn Unit_Platform off}

The process we took

For both Libraries and Apps

Process steps – include files

- {$i NNdefine.inc} in every unit
  - Contains global defines

```
{$define d5}        {$define d5up}
{$define d10}       {$define d10up}
```

  - Contains global compiler directives

```
{$ifdef d7up}
  {$A4}     { dword-align }
{$else}
  {$A+}     { align on a machine-word boundary }
{$endif d7up}
```

  - Includes Project.inc 2 times:
    - Prolog
    - Epilog

Process steps – include files

- Project.inc (required per project)
  - Prolog sets up NNdefine.inc parameters
    - {$define develop} /
    - {$define test} /
    - {$define ship}
  - Epolig reacts on NNdefine.inc settings
    - {$warn Symbol_Platform off}
    - {$warn Symbol_Deprecated off}
    - {$warn Unit_Platform off}
  - Can include ProjectGroup.inc for more global stuff
- ProjectGroup.inc (optional)
  - Arranges stuff on a more global level
    - {$define PdfViaAcrobat}
    - {$define PdfViaIExplore}
    - {$define FastMM} { test voor heap leaks }

Part 3

Technical port Applications

Hardest issue

- The DLL's were causing havoc
  - Access violations
    - {$A4} was required to get them to work
  - Wrong versions of the DLL's
    - Watch your path
  - Exported functions not found
    - These are CASE SENSITIVE
- In the end, we refactored the DLL's away

Easier issues

- NetMasters was removed in Delphi 6
  - Use Indy instead
  - In this case
    - rewrite usage of TNMFinger with TIdFinger
    - Accommodate differences between the two
- TModalResult moved from unit Forms to unit Controls in Delphi 7
- TDataModule moved from unit Forms to unit Classes in Delphi 7

Easier issues (2)

- Since Delphi 2005, virtual methods are case sensitive
  - Especially the Destroy method of TObject is case sensitive
  - This is because of .NET issues, and package DLL exports
    - actually, similar to DLL export issues – they are case sensitive
    - Similar to the fact since Delphi 1, the "procedure Register;" has been case sensitive this impacted component writers, now it impacts more people
- [Pascal Hint] Y1SAAP022.pas(176): H2365 Override method TPdcProcesQueryList.**destroy** should match case of ancestor TEAObject.**Destroy**

FastMM

- Put all application logic in a separate unit
- Project.dpr
  - Uses all referenced units to:
    - Make project manager happy
    - Make form-inheritance work
- MainUnit.pas
  - First unit used in Project.dpr
  - Contains the actual main block
  - Initializes FastMM and other stuff

FastMM – Project.dpr

```
{$i nndefine.inc}
program Project;
uses
  MainUnit in 'MainUnit.pas',
  Y35AAD00 in '..\shared\Y35AAD00.pas' {DmcDataBase: TDataModule},
  // ...
  Y35AAD18 in '..\shared\Y35AAD18.pas' {DmcEasy: TDataModule};
{$R *.RES}
begin
  Main;
  //Application.Terminate;
end.
```

FastMM – MainUnit.pas

```
{$I nndefine.inc}
unit MainUnit;
interface
uses
{$ifdef FastMM}  FastMM4,{$endif FastMM}
  NNNLResourceStrings, (*...*)  Forms,  SysUtils;
procedure Main;
implementation
procedure Main;
begin
{$ifdef FastMM}  ReportMemoryLeaksOnShutdown := true;{$endif FastMM}
  Application.Initialize;
  // Application.CreateForm(.....);
  Application.Title := 'Easy Autoindex';
  Application.Run;
end;
end.
```

Part 4

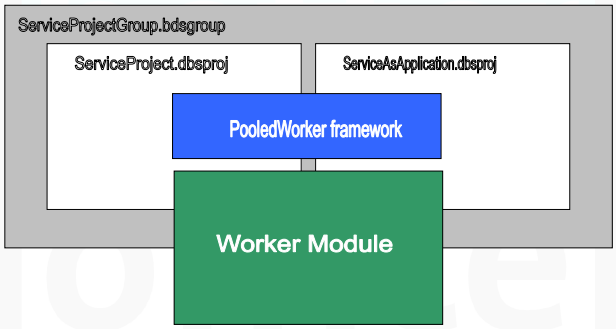
Refactorings

A bit out of scope, but still

Refactoring

- You should use this often
- Current corporate shortsighted bureaucrazy usually shoots this down
- But it makes your life a lot easier
  - So do it anyway
  - It will pay off in future cycles

Apps to services



BDE —> ADO (1)

- For BDE with SQL Server, you need this DLL in your EXE subdir
  - NTWDBLIB.DLL
    - Throw this away when converted to ADO
      - If your app now fails, it still contains some BDE stuff

BDE —> ADO (2)

- Delphi units that use the BDE
  - BdeMts
  - DbExcept
  - DBTables
  - dbLookup
  - drTable
  - The Decision Cube units
- TDatabase —>
  - TAdoConnection
- TTable —>
  - TAdoDataSet
- TStoredProc —>
  - TAdoDataSet
- TQuery —>
  - TAdoDataSet
  - TAdoCommand

BDE —> ADO (3)

- TAdoDataSet
  - far better design-time support than TAdoQuery and TAdoTable
- TAdoCommand
  - For readonly stuff, far better performance than TAdoDataSet

Get rid of DLL's

- They were our biggest application issue
- Alternatives
  - Go monolithic
    - Embed everything in your EXE's
  - Go for packages
    - Far better version mismatchs errors
    - Far better type-safety

Exception refactoring

- Assess Exception usage
  - EAbort is silent
  - Empty "except...end" are vicious
  - Reraising a different exception hides the original
    - Use a kind of "InnerException" construct
  - Object creation must be OUTSIDE the try:

```
myObject := TObject.Create;
try
//    myObject := TObject.Create; // this is BAD!
finally
    myObject.Free;
end;
```

Misc refactoring

- When returning objects from functions, make sure who the owner is
  - Alternatively, use Interfaces
    - They are reference counted, so freeing is automagical
- Duplicate code
  - Sometimes, whole units were copied
    - Merged back into one, and cross-applied bugfixes
- Consolidating Database connections
  - Your apps become faster
  - Usually less transaction problems

From Delphi 2006 to Delphi 2007 to Delphi 2009

Delphi 2007

- Is VCL compatible with Delphi 2006
- Introduces Vista support
- You only need changes if you actively want Vista support
  - Glass UI
  - Shield support
  - Vista directory structures

Delphi 2009

- Breaking change
  - Unicode strings
- Compiler progression
  - Generics
  - Anonymous methods
- Some examples in the slides *after* the Q&A

Discussion time

What conversions are you going to do

What are you afraid of

Q & A

Jeroen Pluimers
better office benelux
jpluimers@better-office.com
If you have questions after the session, please mail me