

# Réussir sa migration .NET avec Delphi 2006



Microsoft  
.net

Olivier Dahan

e-n@Xos  
[www.e-naxos.com](http://www.e-naxos.com)

## Migrer ?

- Migrer son code ?
  - Langage : Delphi.NET
  - Problématique : compatibilité ascendante
- Migrer son savoir-faire ?
  - Langage : Delphi.NET ou C# ?
  - Problématique :
    - Delphi.NET : idem ci-dessus
    - C# : courbe d'apprentissage + coût pour faire ou refaire



## Migrer, plusieurs voies

- Migrer son code
  - Oui, s'il y en a...
  - Oui, pour le code métier bien écrit...
- Migrer son savoir faire
  - Indispensable : .NET est le futur de Windows
  - Choisir la méthode douce Delphi.NET ou tout réapprendre avec C# ...



## Migrer par nécessité

- Aujourd'hui 99% de votre code Win32 pourrait certainement être écrit en Delphi 1 Win16 ...
- Mieux : ce code pourrait toujours être exécuté sous Windows !
- ...Mais cela ne vous viendrait pas à l'idée de concevoir ou de vendre un logiciel 16 bits aujourd'hui... Il en va de même de .NET ...



## Migrer par choix aussi !

- .NET propose un framework objet enfin à la hauteur des langages Orientés Objet
- Il fédère les meilleures technologies des 10 dernières années qui étaient éparses, autant de « verrues » ajoutées avec le temps, en un tout cohérent
- Le code est plus fiable, plus sécurisé
- Les langages sont plus puissants (C# mais aussi les grandes avancées de Delphi.NET)
- Les EDI ont eux aussi évolués de façon très singificatives : refactoring, testing, UML... Delphi 2006 c'est aussi tout cela !



## Migrer : changer d'époque !

- Sous Win32 les EDI étaient avant tout des environnements de développement résolument orienté « codage »
- Sous .NET les EDI comme D2006 deviennent des « laboratoires de développement » de l'analyse au déploiement



## Un mot sur C#...

- Nous ne l'aborderons pas ici  
(mais lors de la 2de session nous aborderons ASP.NET avec les deux langages, C# et Delphi.NET)
- C'est un langage puissant et séduisant
- Comme tout nouveau langage il faut du temps pour acquérir de l'expertise
- La double compétence Delphi.NET/C# semble être le choix de la raison pour l'ingénieur IT, l'indépendant, moins pour l'éditeur ou le service informatique d'une entreprise.



## La migration du code Win32

- Migration de code métier
  - Juste des classes, des bibliothèques de code
- Migration de composants VCL
- Migration d'interfaces utilisateurs



## Migration de code

- Le coût de la migration des librairies de classes, les unités de code repose sur :
  - La quantité de code à migrer
  - Le % de compatibilité ascendante du compilateur
  - De la qualité d'écriture du code
  - De la quantité de code lié à la plate-forme
- Intérêt ?
  - Réutilisation du code, réduction des coûts, capitalisation d'un savoir métier, le tout en s'ouvrant aux nouvelles possibilités de .NET
  - Pouvoir fournir du code .NET exécutable à des développeurs C#, J#, VB.NET etc...
  - Pouvoir utiliser du code sans traduction de header ou autre



## Migration de composants VCL

- Même analyse de coût que pour les bibliothèques de classes
- Intérêt ?
  - Les mêmes que les bibliothèques de classes
  - Partager son code et ses composants mais dans la limite des utilisateurs de Delphi



## Migration d'IHM

- VCL.NET
  - Compatibilité ascendante excellente
  - Rentabilité maximale
- Windows Forms
  - Il faut refaire la partie interface (coût...)
  - Techniquement peu d'intérêt
  - Seul avantage : code plus facilement échangeable avec d'autres développeurs (C#, VB.NET...)



## Intérêt de la VCL.NET ?

- Vaut-il mieux tout refaire en WinForms ou conserver la VCL ?
- Même logique que « quitte à passer à .NET autant le faire en C# » : cela coûte cher !
- VCL.NET = productivité immédiate
- WinForms est temporaire (-> XAML)
- Techniquement aucune différence (P/Invoke)
- VCL.NET : nombreux composants
- VCL : Win16, Win32, Linux, .NET, XAML?... WinForms : Compact framework (avec de grosses restrictions).



## Migration de code

- Compatibilité ascendante du compilateur excellente
- Des limites imposées par .NET au langage donc quelques modifications du code à prévoir, par force
- Des adaptations à prévoir comme dans tout changement de plate-forme



## Ce qui ne change pas

- Les chaînes et les tableaux (string et array)
- Les record (mais ils supportent de nombreuses choses désormais)
- Les déclarations de classes et d'interfaces
- Les propriétés et événements
- Les ensembles
- Les fichiers texte (textfile)
- Les composants « pure VCL »
- La gestion des flux
- Les procédures et fonctions utilitaires de la VCL



## Le code unsafe

- Pour le débuser le mieux est d'utiliser Delphi 7 ou Delphi Win32 2006 avec les avertissements spécifiques .NET
  - {\$WARN UNSAFE\_TYPE ON}
  - {\$WARN UNSAFE\_CODE ON}
  - {\$WARN UNSAFE\_CAST ON}.



## Les types unsafe

- Pchar, PWideChar, PAnsiChar
- Tous les pointeurs non typés
- Tous les paramètres out et var non typés
- Les « File of <type> »
- Les vieux réels TP (real48)
- Les records avec partie variable



## Les constructions non portées

- Les variables absolute
- Tout ce qui indique une adresse mémoire (Addr, Ptr, Hi, Lo, Swap)
- BlockRead et BlockWrite
- Fail dans les constructeurs
- Allocation: GetMem, Freemem, ReallocMem



## Les constructions non portées

- Remplissage ou déplacement mémoire (Fillchar, Move)
- L'ancien type Object de TP
- Les routines asm
- Les unités utilisant ExitProc
- L'opérateur d'adresse @



## Les transtypages interdits

- Transtyper un objet dans un type qui n'est pas un ancêtre
- Transtyper un record en n'importe quoi d'autre
- Transtyper sous .NET : équivalent à AS, la transtypage « sauvage » est interdit par le CLR, Delphi n'y peut rien...



## Les erreurs plus sournoises

- Sous Win32 les valeurs flottantes de moins de 4 chiffres après la virgule sont transformées en Currency dans une affectation à un Variant
- Sous .NET ce type est « Double »
- Si le type de la donnée est testée dans le code cela ne fonctionnera donc plus



## Autre exemple

- Sous .NET les destructeurs ne sont pas forcément appelés...
- Tout code qui relâche des ressources autre que de la mémoire sur un destructeur (par exemple le OnDestroy d'un TForm...) doit être revu.



## Équivalences

| Delphi Win32 | Delphi .NET                                |
|--------------|--------------------------------------------|
| TObject      | System.Object                              |
| Variant      | Variant défini comme System.ValueType      |
| Exception    | System.Exception                           |
| TComponent   | System.ComponentModel.Component            |
| String       | System.String                              |
| Record       | System.ValueType (nouvelle syntaxe)        |
| AnsiChar     | Simulé (Borland.Delphi.System.AnsiChar)    |
| ShortString  | Simulé (Borland.Delphi.System.ShortString) |
| AnsiString   | Simulé (Borland.Delphi.System.AnsiString)  |
| WideString   | System.String                              |
| IHM          | System.Windows.Forms et la VCL .NET        |



## String

- Le type String de .NET est un type non modifiable ...
- Les parties de code utilisant du travail intensif sur les chaînes devront être modifiées pour utiliser `StringBuilder`



## Équivalences de type

| Caractères |             |
|------------|-------------|
| Char       | System.Char |
| WideChar   | System.Char |

| Logiques |                |
|----------|----------------|
| Boolean  | System.Boolean |

| Entiers  |               |
|----------|---------------|
| Byte     | System.Byte   |
| Shortint | System.SByte  |
| Smallint | System.Int16  |
| Integer  | System.Int32  |
| Word     | System.UInt16 |
| LongWord | System.UInt32 |
| Cardinal | System.UInt32 |
| Int64    | System.Int64  |

| Flottants |                                        |
|-----------|----------------------------------------|
| Single    | System.Single                          |
| Double    | System.Double                          |
| Real      | System.Double                          |
| Decimal   | System.Decimal                         |
| Currency  | Simulés                                |
| Extended  | Alias vers<br>System.Double            |
| Comp      | Deprecated, alias vers<br>System.Int64 |



## Le code qui « ne passe pas »

- Que faire si un code ne « passe pas » et que sa réécriture est trop longue ou trop coûteuse ?
  - Transformation en DLL Win32
  - Appel de la DLL Win32 depuis le code .NET (P/Invoke)



## Migration mixte transitoire ?

- Le code .NET peut appeler du code Win32 (P/Invoke)
- ...Mais le code Win32 peut aussi appeler du code .NET !
  - De nouveaux modules peuvent être développés sous .NET et être appelés à la fois par du code .NET et de l'ancien code Win32.



## Portage d'applications VCL

- Faut-il tout porter ?
  - NON
  - Par défaut, le code existant qui fonctionne n'a aucune raison d'être migrer !
  - Le code métier est migré s'il y a un projet de réutilisation (lapalissade...)
  - Les applications sont à migrer si elles doivent évoluer dans le futur (idem...)



## Migration « automatique »

- Delphi.NET sait ouvrir les DPR Win32
- Il ajoute et modifie ce qui permet d'en faire un projet Delphi.NET
- Grâce à la VCL.NET compatible VCL Win32 D7, grâce à la compatibilité ascendante du compilateur tout est automatique
- ... Aux différences près déjà évoquées...



## Code Win32 / .NET

- En raison de la très grande compatibilité du langage très peu de choses sont modifiées dans un projet DPR Win32.
- Il est donc possible de concevoir un code compilable Win32 et .NET (merci la VCL...) :  
`uses Windows, Classes, Graphics, Forms, Controls,  
{ $IFDEF CLR } System.ComponentModel, { $ENDIF } StdCtrls;`
- Le code compilé (dcuIL) ne peut se mélanger au code Win32 (DCU) seul le nom de l'exécutable (.EXE) est à contrôler (le même)



## Compatibilité cross platform

- Le code conditionnel permet la compilation d'un même code sous Win32, VisualCLX et VCL.NET

```
project MultiPlatform;  
{ $APPTYPE CONSOLE }  
Var s:string;  
begin  
  { $IFDEF CLR } // Delphi .NET  
  S:='.NET !';  
  { $ENDIF }  
  { $IFDEF WIN32 } // Delphi Win32  
  S:='Win32';  
  { $ENDIF }  
  { $IFDEF LINUX } // Kylix  
  S:='Linux';  
  { $ENDIF }  
  writeln('Bienvenue sous '+s);  
end.
```



## Les composants tiers

- Ce sont eux qui posent le plus de problèmes:
  - Existent-ils sous VCL.NET ?
  - Possédez-vous le code source ?
  - Celui-ci est-il migrable facilement ?
  - Peuvent-ils être facilement remplacés par un composant VCL proche ?
- Des tests doivent être faits sur tous les composants tiers avant toute migration d'applications !



## Les étapes d'une migration réussie

- Étape 0 : Sauvegarde du projet ...
- Étape 1 : Contrôles sous Delphi Win32
- Étape 2 : Modifications sous Delphi Win32 jusqu'à ce que tout soit ok
- Étape 3 : passage sous Delphi.NET, contrôles et corrections

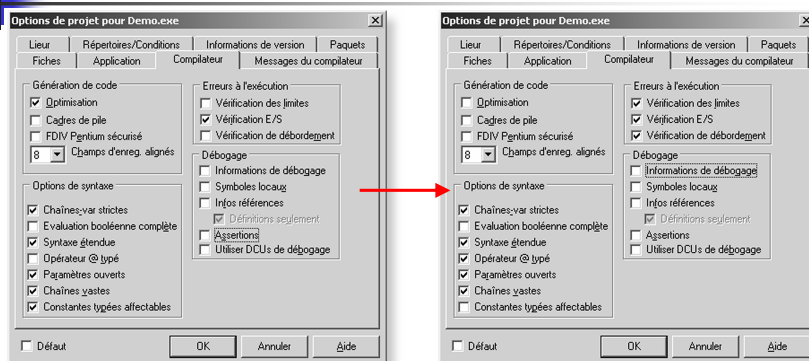


## Les contrôles sous Win32

- L'essentiel d'une migration sous .NET se passe.. Sous Delphi Win32 !
- Commutateurs de compilation (warnings)
- Correction du code
- Suppression et remplacement du code tiers qui ne passe pas
- Passer les fiches DFM en mode texte



## Options de compilation Win32



Suppression optimisation, Typage fort pour @, Suppression des constantes typées modifiables, Mise en route des erreurs d'exécution.



## Migration des accès données

- Sous VCL.NET
  - On retrouve les mêmes composants que sous D7 (dbGo, IB express, DB express...)
  - Portabilité immédiate du code existant
  - Reste des techniques Win32 majoritairement
  - ADO VCL.NET correspond à ADO 32b et non à ADO.NET



## Migration sous WinForms

- Sous WinForms
  - ADO.NET et ses classes spécialisées par type de serveur
  - Borland Data Provider, sorte de db Express pour ADO.NET
  - Plus natif, portable Compact Framework, mais impose réécriture IHM



## Différences VCL / VCL.NET

- BDE local est frozen depuis longtemps, il est repris en VCL.NET
- SQL Links, deprecated, plus fourni (mais si vous avez D7, les pilotes fonctionnent encore)
- DB express : repris en natif .NET
- IBX : repris



## Supports étendus VCL.NET

- DataSnap déploiement gratuit
- dbExpress : IB 7.5, MS Server 2000 et 2005, Informix 9.x, DB2 8.x, MySQL 4.0.24, SQL Anywhere 9, Sybase 12.5
- Support Unicode pour SQL server et l'ensemble des compos DB
- Séparateur décimal Oracle, passage de paramètres de connections, MS Server valeurs de retour des SP,



## Migration des états

- Rave Report est fourni sous VCL et VCL.NET comme sous D2005, D8 et D7
- Quick Report 4.06 VCL.NET est sorti et peut être acqui auprès de QuSoft
- De nombreux outils spécifiques .NET existent sur le Web



## Migration des applis Web

- Web service totalement repris
- WebSnap, WebBroker en partie
- IntraWeb Win32 et version VCL.NET
- Intérêt des anciennes technologies ?
  - Si l'existant fonctionne, le laisser tel quel...
  - Les nouveaux projets doivent être fait sous ASP.NET, bien plus puissant et plus simple



## Pour conclure

---

- Avec Delphi 2006 :
  - Productivité immédiate
  - Écriture de solutions natives
  - VCL.NET ou WinForms : choix évolutif
  - Environnement plus puissant améliorant la productivité
  - Maintenir le code Win32, migrer sous Delphi.NET, se former à C# : 1 seul produit



## Fin...

---

- Merci de votre attention...
- Slide avec exemple de code téléchargeable prochainement sur **WWW.E-NAXOS.COM**