【セッションNo. 4】

Delphi/400 テクニック公開

Windows7に最適化した アプリ開発・運用テクニック

株式会社ミガロ.
システム事業部　プロジェクト推進室
尾崎　浩司

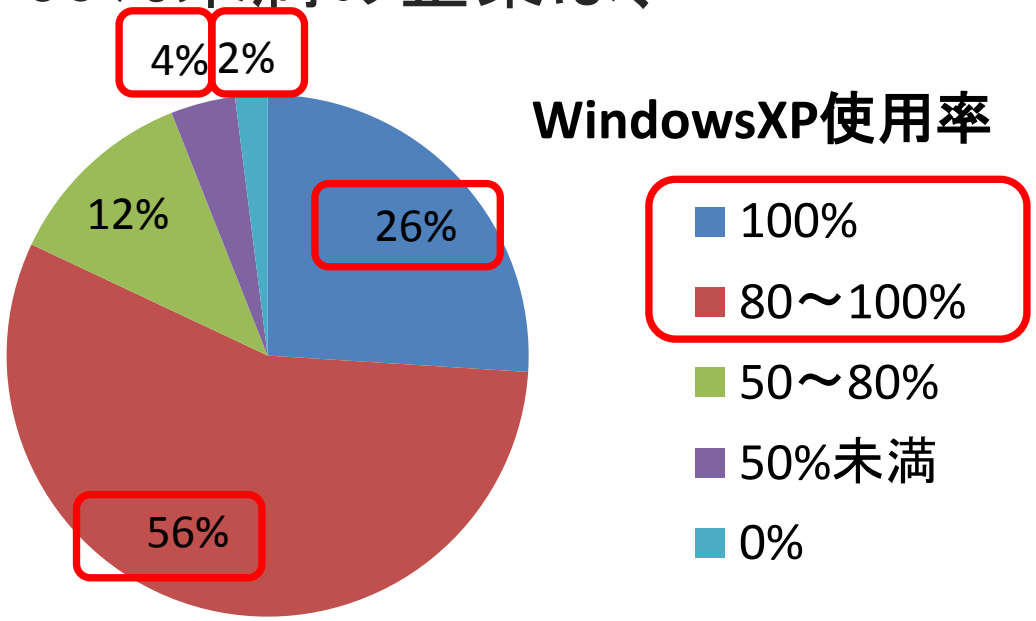
【アジェンダ】

1. はじめに
2. Windows7は、ここが変わった！
3. Windows7に対応させる開発ノウハウ
4. Windows7対応版Delphi/400 versionXEを使用するメリット
5. まとめ

1. はじめに

企業におけるWindowsXP利用状況

- 企業ユーザーの**約82%**が未だにWindowsXPをメインに使用
- WindowsXP使用率が50%未満の企業は、わずか**約6%**

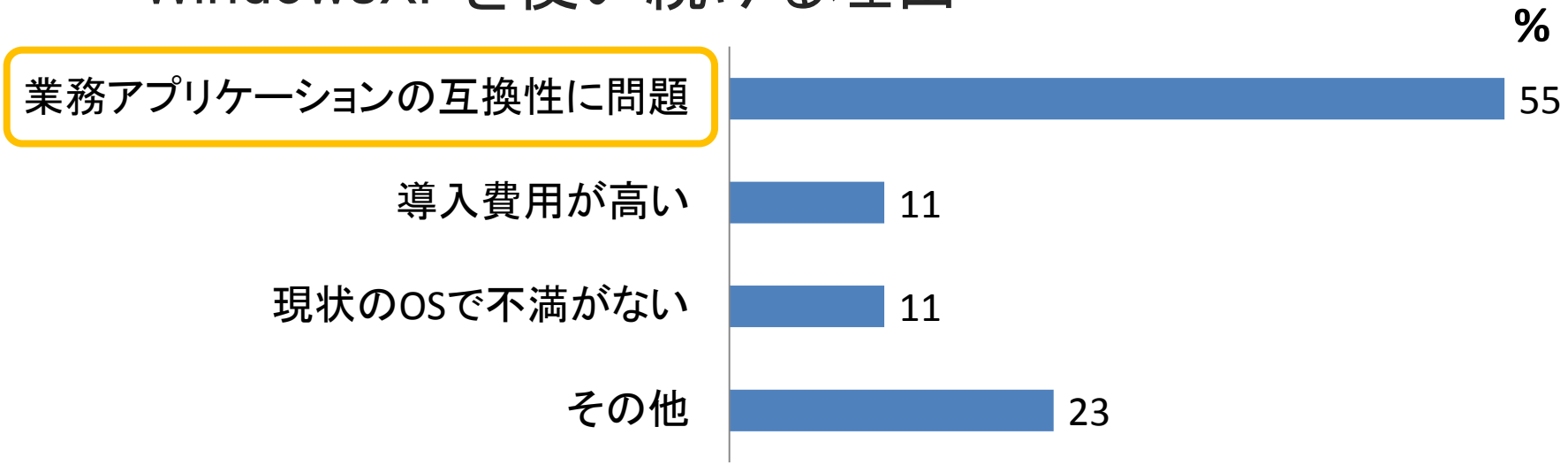


社団法人日本情報システム・ユーザー協会（JUAS）
「企業のIT投資動向に関する調査研究」（企業IT動向調査2011）より

Windows7導入状況

- 2012年度迄に**約75%**の企業が導入を開始
  - 2014年4月にWindowsXPのサポートが遂に終了

- WindowsXPを使い続ける理由



社団法人日本情報システム・ユーザー協会（JUAS）
「企業のIT投資動向に関する調査研究」（企業IT動向調査2011）より

Delphi/400のWindows7対応

- Version2010にて正式対応

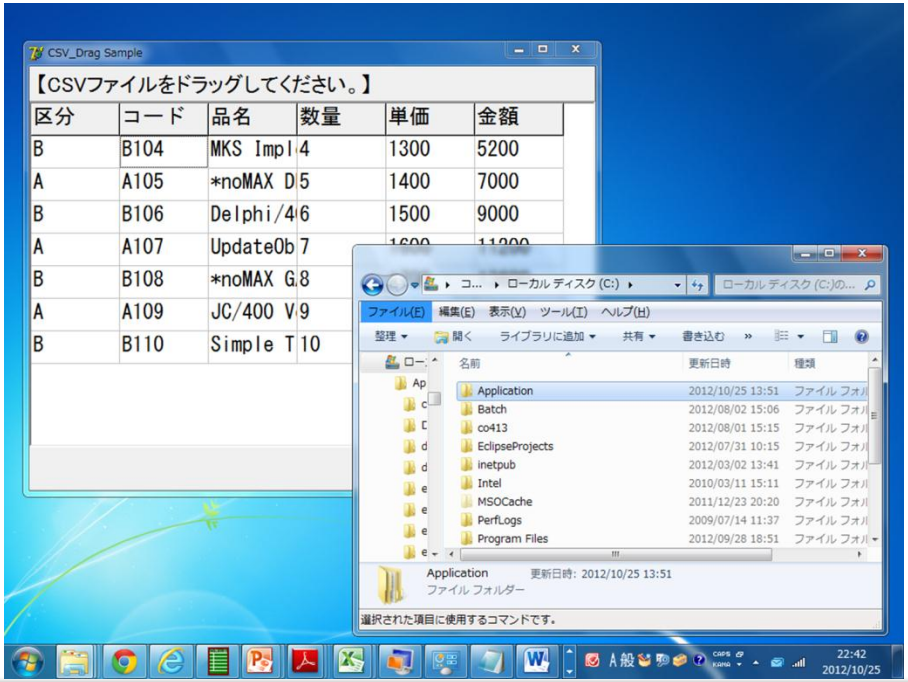
	Delphiバージョン	7	2005	2006	2007	2009	2010	XE
動作環境	Windows3.1	－	－	－	－	－	－	－
	WindowsNT3.51	－	－	－	－	－	－	－
	Windows95	－	－	－	－	－	－	－
	WindowsNT4	－	－	－	－	－	－	－
	Windows98	○	－	－	－	－	－	－
	WindowsMe	－	－	－	－	－	－	－
	Windows2000	○	○	○	○	○	○	○
	WindowsXP	○	○	○	○	○	○	○
	WindowsServer2003	－	○	○	○	○	○	○
	WindowsServer2008	－	－	－	－	○	○	○
	Windows Vista	－	－	－	○	○	○	○
	Windows 7	－	－	－	－	－	○	○

- 旧バージョンであっても、Windows7動作保証は無いが、アプリケーションを稼働させることは可能！
  - 但し、注意しなければいけない点もある
  - 今回はWindwos7で使用する際のノウハウを紹介

2. Windows7は、ここが変わった！

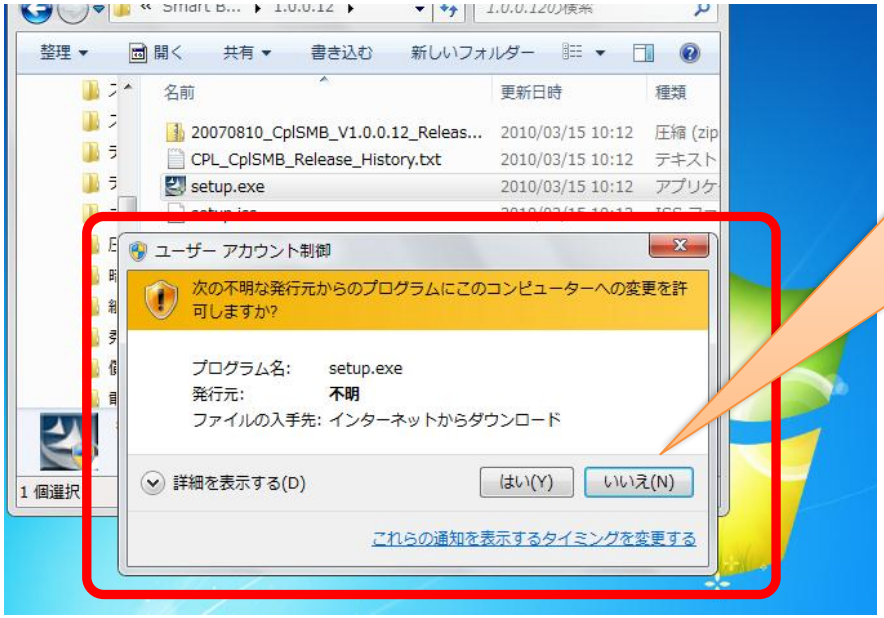
WindowsXP と Windows 7

- 画面は、繊細なアニメーションと半透明のグラスウィンドウが特徴のAero対応により見た目のイメージは異なるが、基本的にDelphi/400で作成されたWin32アプリケーションはそのまま実行可能。



Windows7はここが変わった①

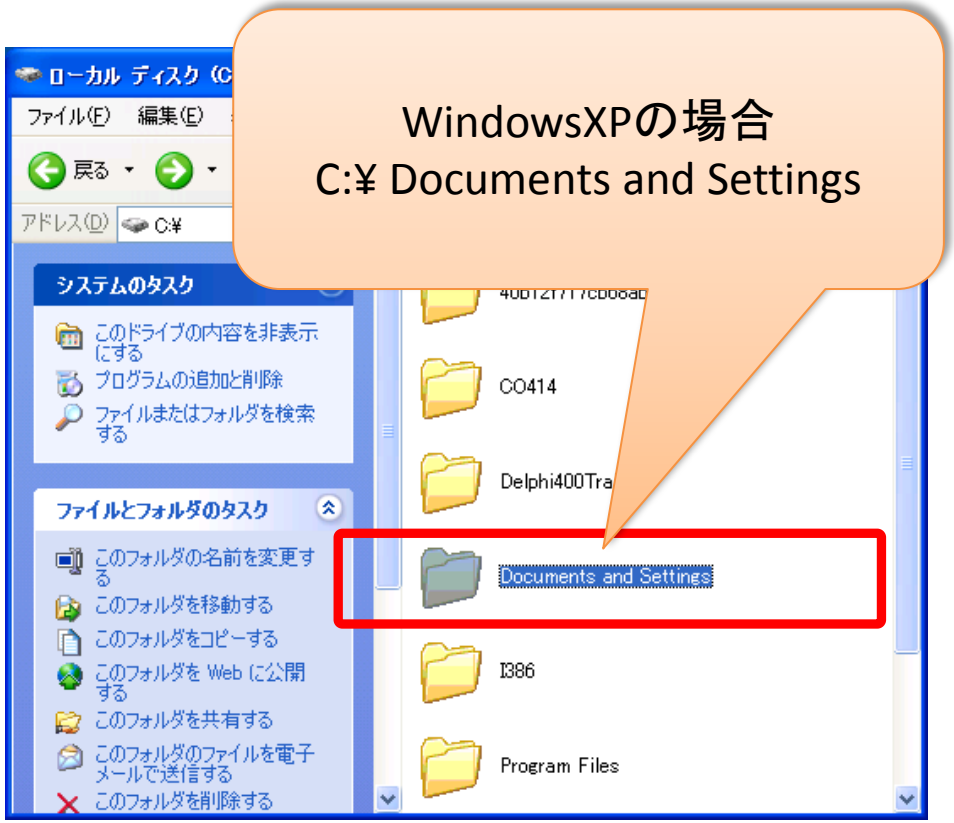
- UAC(User Account Control)
  - 管理者権限(Administratorsグループ)のユーザーでも、通常は一般ユーザーと同じ権限レベルとなり、権限がないとできない操作は必ず確認画面がでてくる。

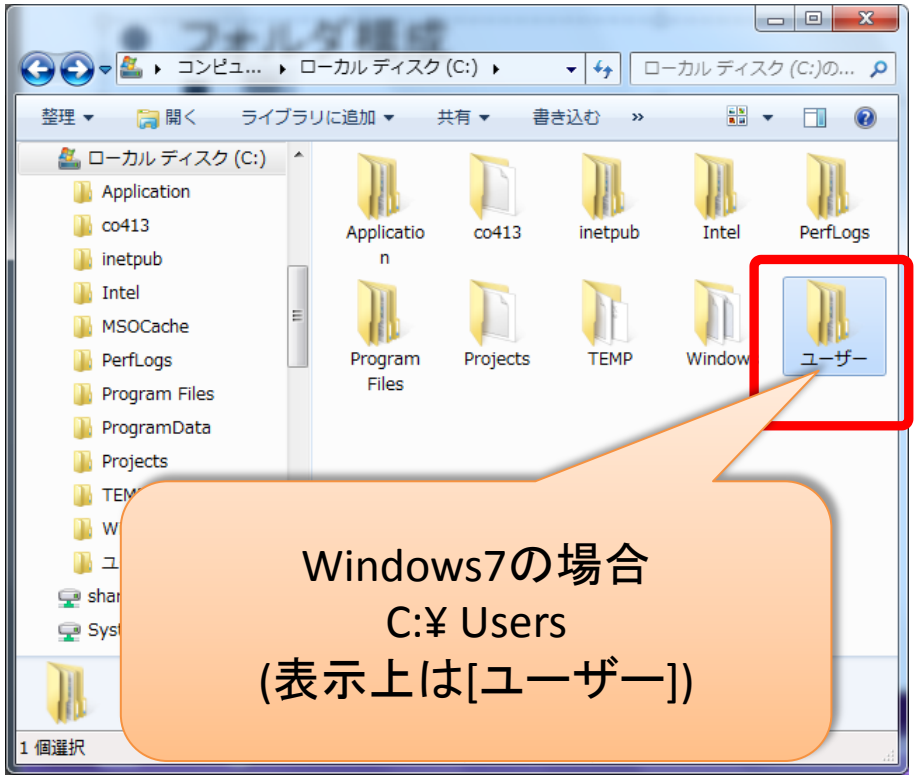


プログラムのインストール等システム設定を変えるような物は、管理者権限でログインしていても、変更を許可しない限り実行不可。

Windows7はここが変わった②

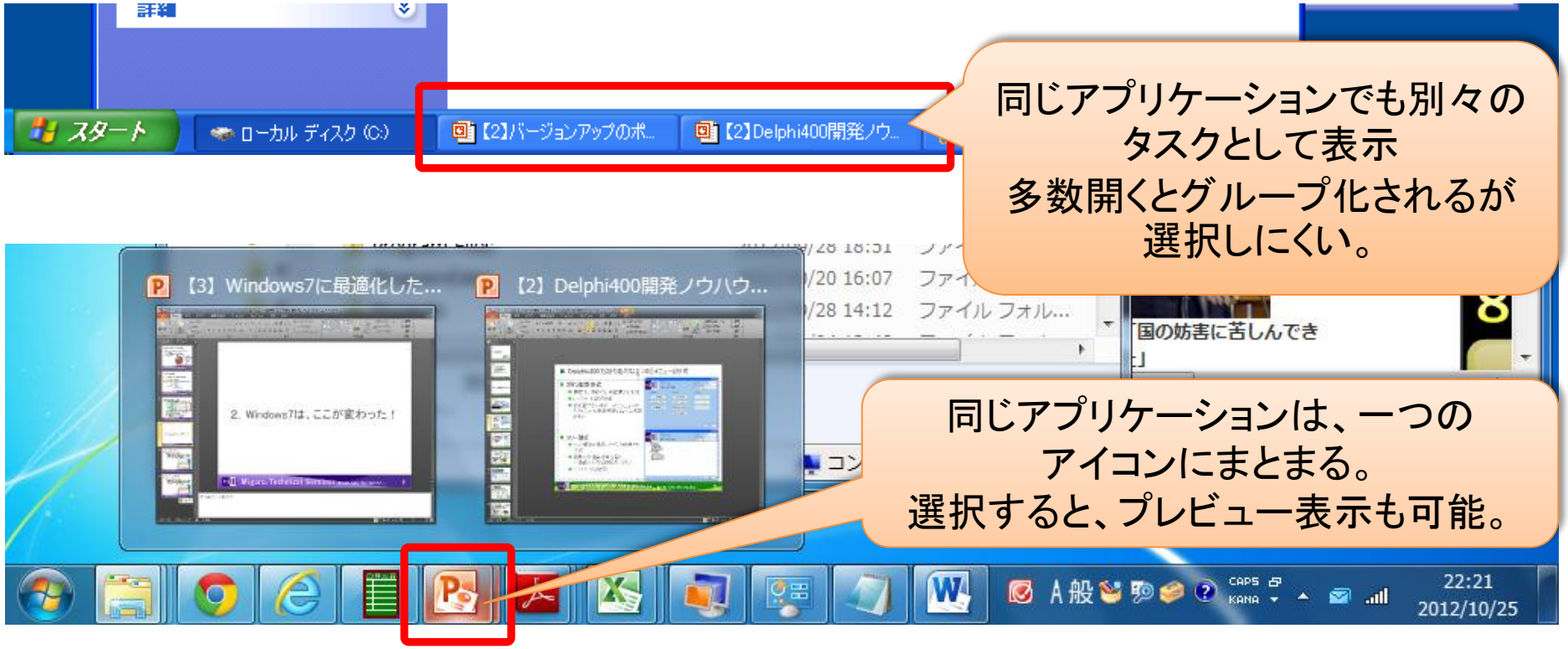
- フォルダ構成
  - ユーザーの個人情報が格納されるフォルダが、WindwosXPとWindows7とでは場所が異なる。





Windows7はここが変わった③

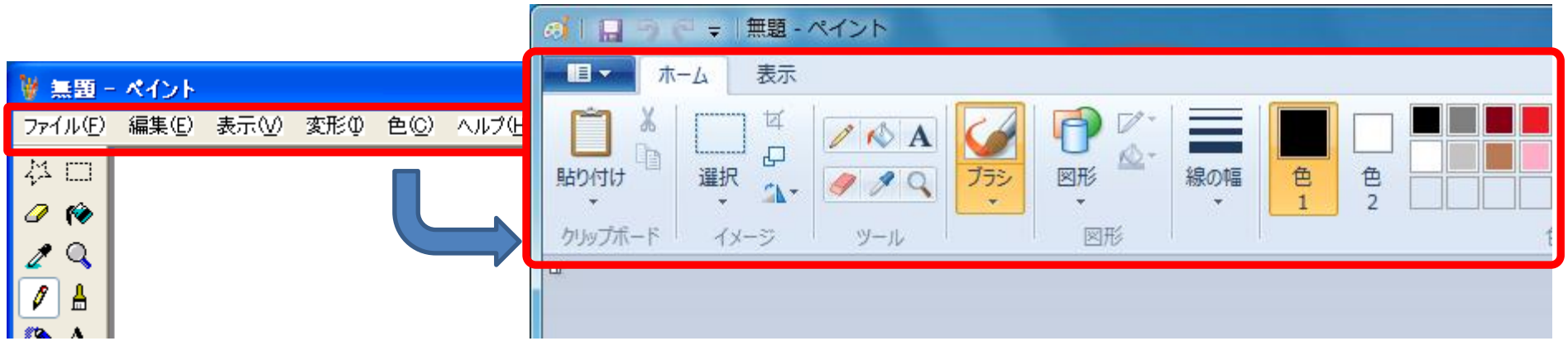
- タスクバー
  - Windows7では、アプリケーション毎にアイコン表示され
    タスクがまとまるため、把握しやすい。



Windows7はここが変わった④

- ジェスチャー/マルチタッチ
  - 従来は、基本的なタッチ操作（クリック）のみ
  - フリック動作やマルチタッチがOSレベルで可能に

- リボン
  - 従来のメニューバーの代替。
  - Office2007で初採用。 Windows7からはOS付属アプリケーションでも使用。（例：ペイント）

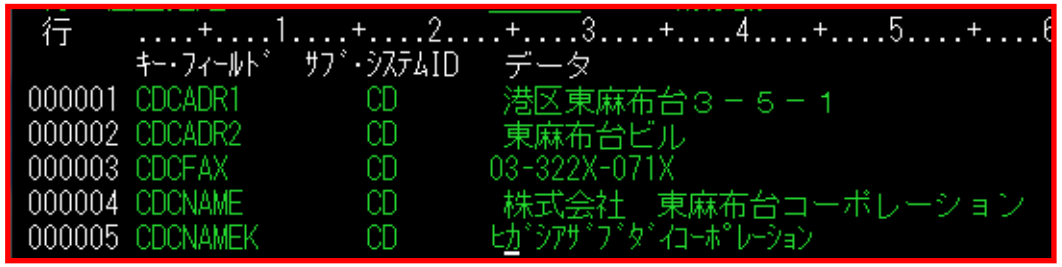


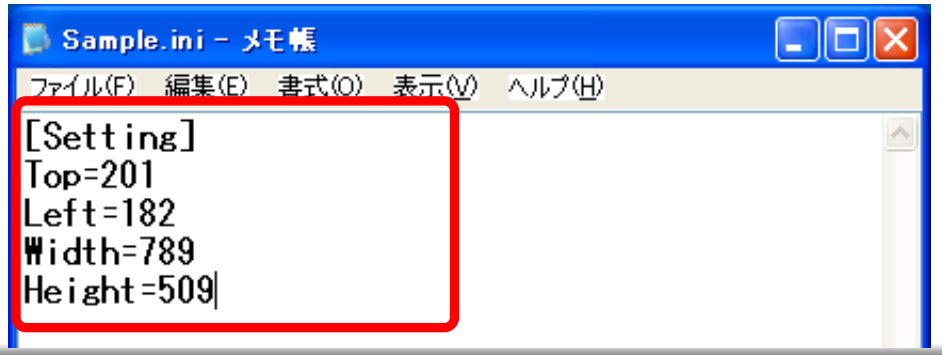
3. Windows7に対応させる開発ノウハウ

（1）UAC対応

アプリケーションの設定情報

- アプリケーション固有の設定情報を保持するには？
  - DB
  - レジストリ
  - INIファイル





INIファイルの使用例

- アプリケーション終了時に、フォームの位置・サイズを INIファイルに書き出す

```
procedure TForm1.FormDestroy(Sender: TObject);
var
  sPath: String;       // Path
  iniFile: TIniFile;   // INI情報
begin
  //アプリケーションファイルPath取得
  sPath := ExtractFilePath(Application.ExeName);
  //INIファイルアクセスオブジェクト生成
  iniFile := TIniFile.Create(sPath + 'Sample.ini');
  try
    //INIファイルにフォームの位置サイズを保存
    iniFile.WriteInteger('Setting', 'Top', Self.Top);
    iniFile.WriteInteger('Setting', 'Left', Self.Left);
    iniFile.WriteInteger('Setting', 'Width', Self.Width);
    iniFile.WriteInteger('Setting', 'Height', Self.Height);
  finally
    //INIファイルアクセスオブジェクト破棄
    iniFile.Free;
  end;
end;
```

Exeファイルが実行されている Pathを取得

取得Path名＋Sample.iniに アクセス

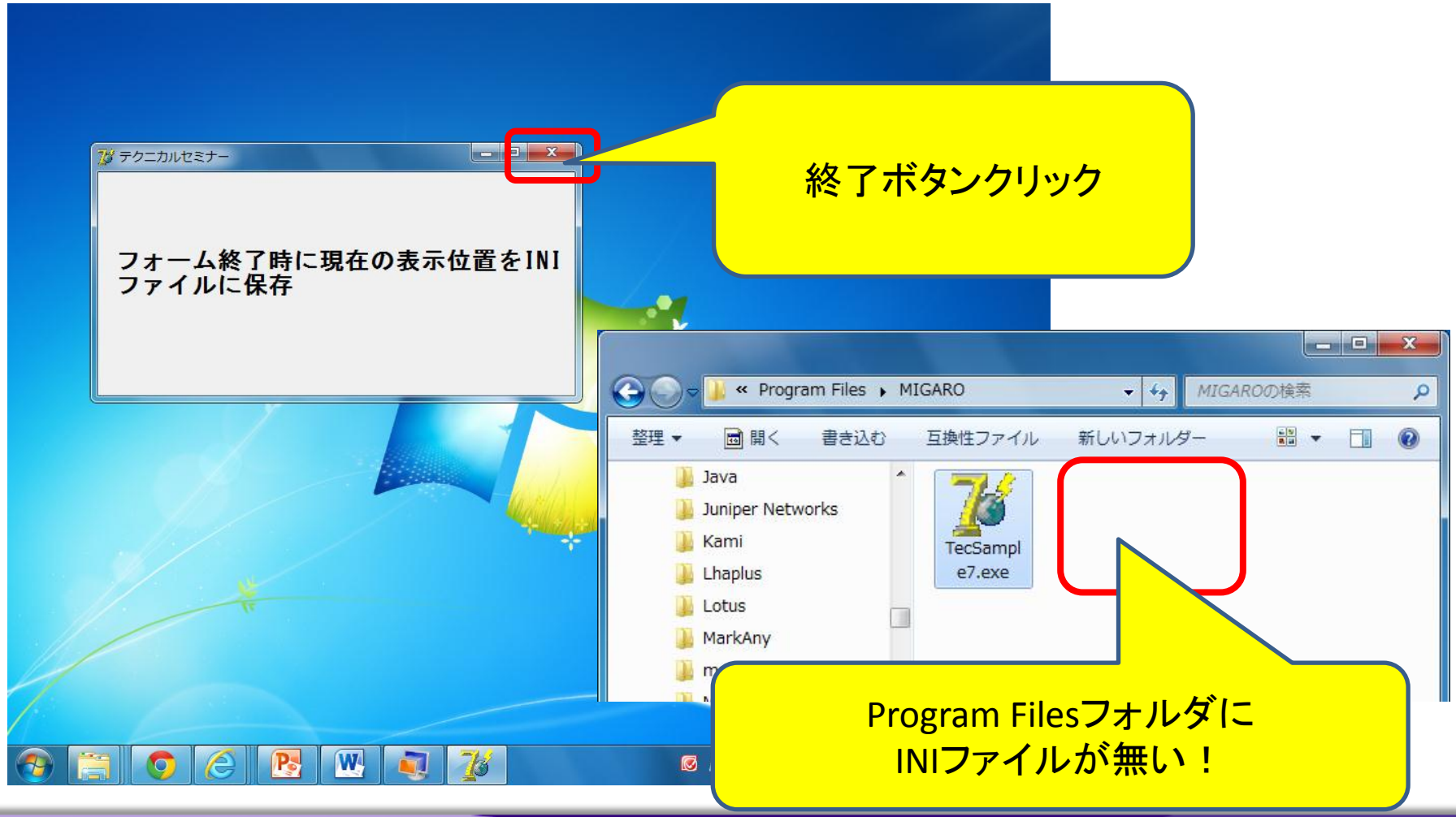
INIファイルに情報書込み

実行イメージ（WindowsXP：Delphi/400 ver.7）

- C:¥Program Files¥MIGARO¥TecSample7.exe

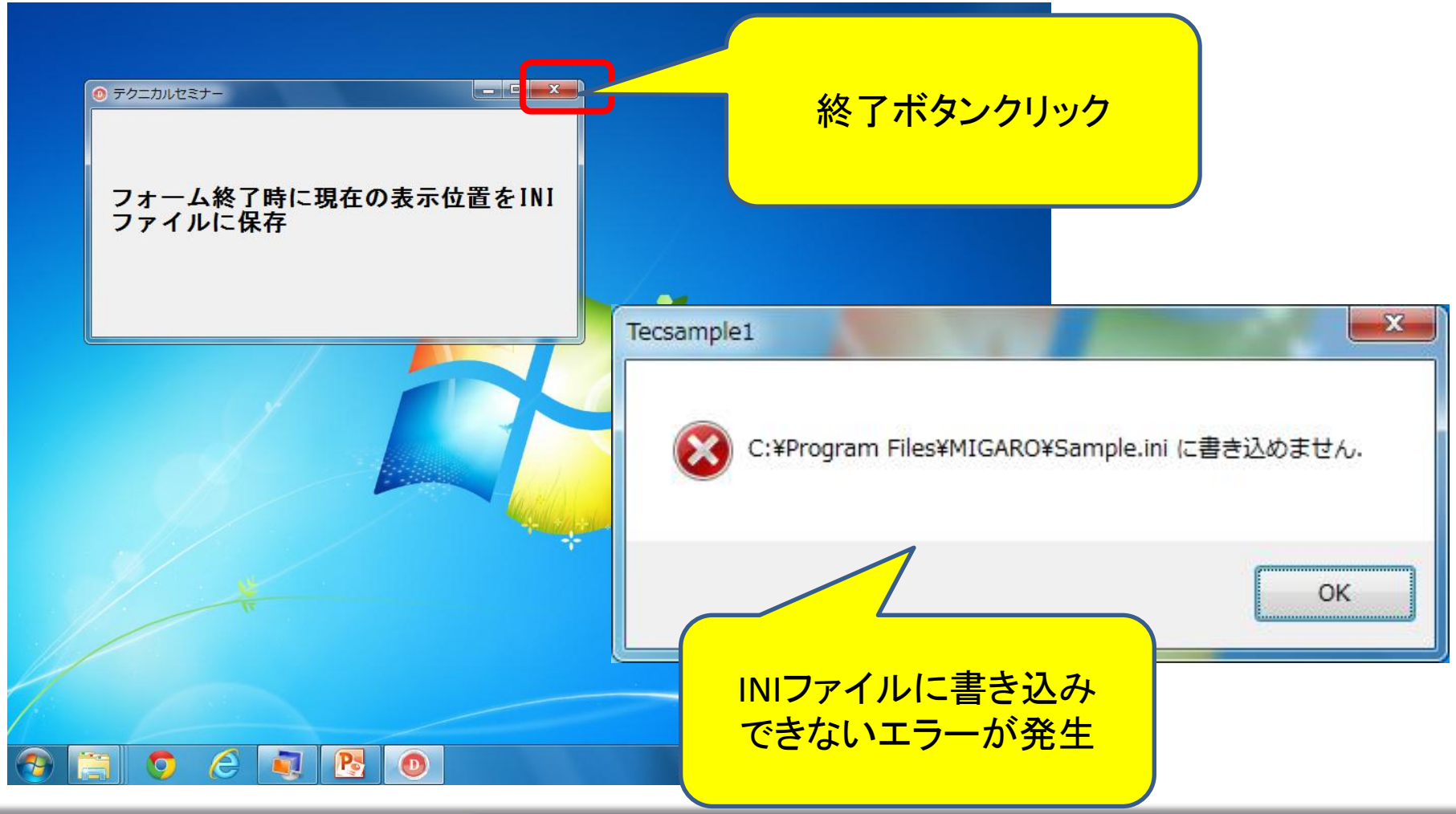
実行イメージ（Windows7：Delphi/400 ver.7）

- C:¥Program Files¥MIGARO¥TecSample7.exe



実行イメージ（Windows7： Delphi/400 ver.XE）

- C:¥Program Files¥MIGARO¥TecSample1.exe



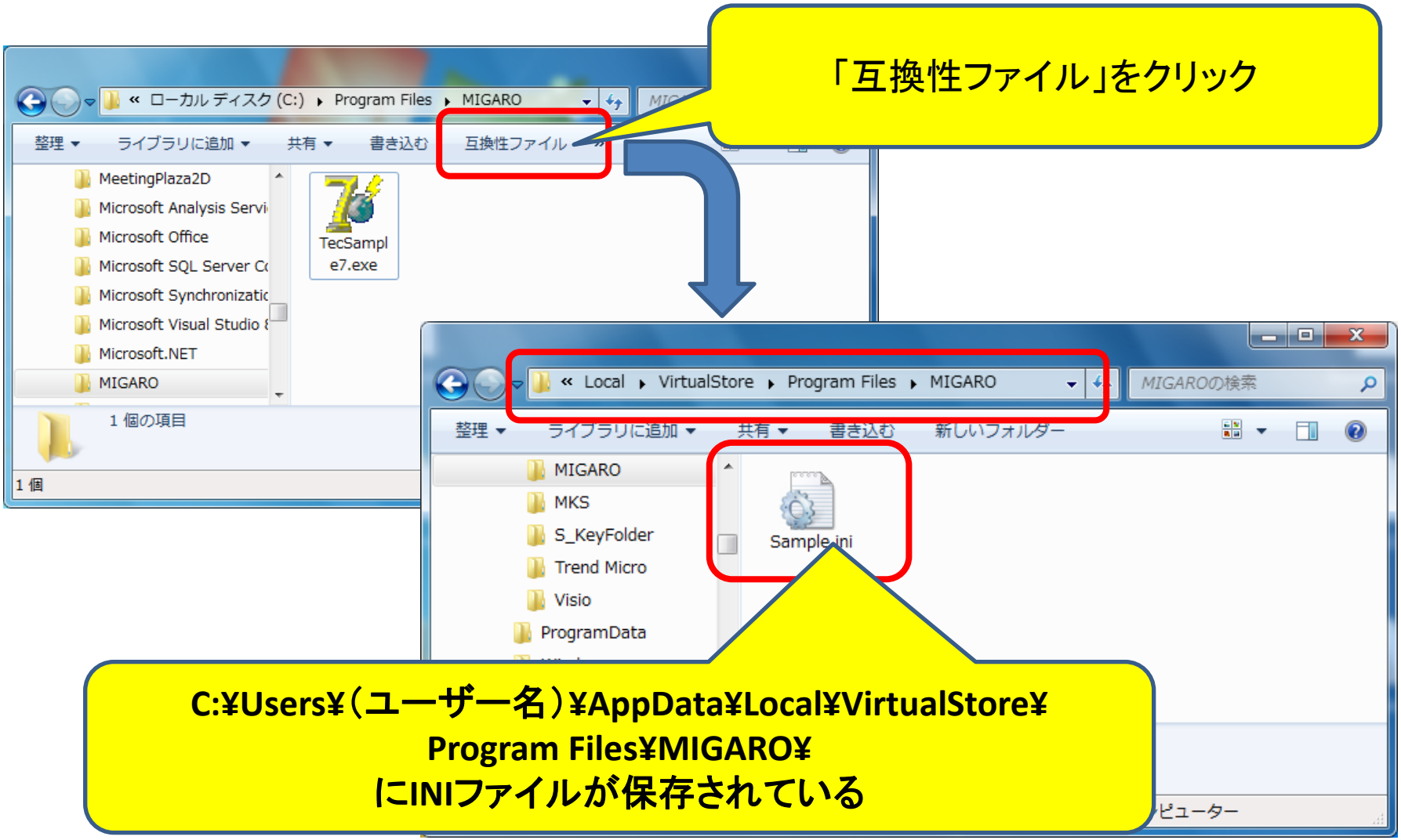
- UAC(User Account Control)
  - Windows7でUACが有効な場合、管理者権限ユーザーでも通常は一般権限しか有しない。
  - 下記に対する書き込みは、都度管理者権限を取得しないと不可。
    - Cドライブ直下
    - フォルダ
      - Program Files フォルダ
      - Windows フォルダ
      - System32 フォルダ
    - レジストリ
      - HKEY_LOCAL_MACHINE¥SoftWare キー

旧バージョンで作成されたExe

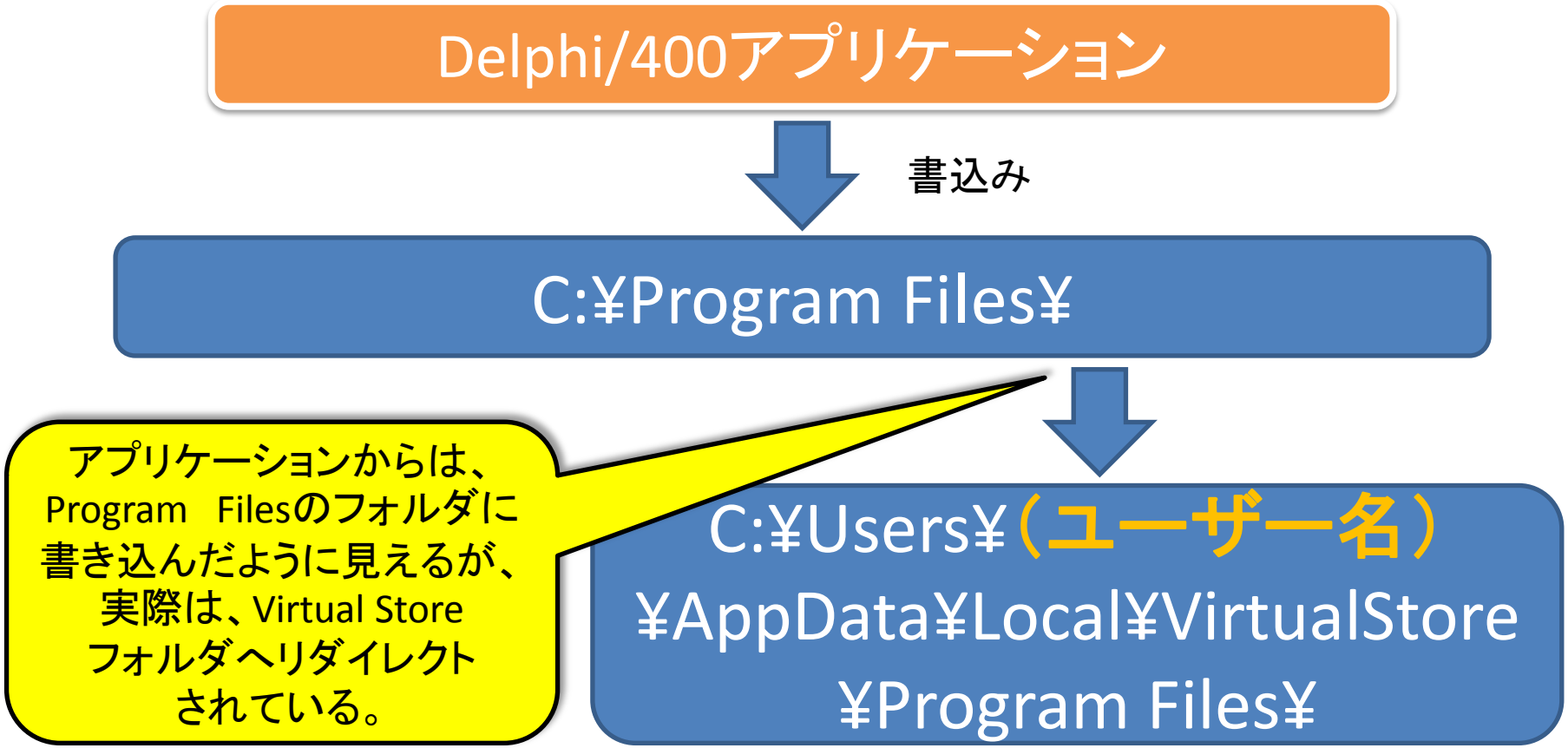
- Delphi/400 ver.7でコンパイルされたExeは、エラーが発生しない。
- しかし、生成されたはずのINIファイルが無い。

- 旧バージョンで作成されたExeの場合、内部に権限に関する情報を持っていない。
- 権限情報が無いExeをWindows7で実行すると、互換性の為に仮想化が適用される。

旧バージョンで作成されたExe



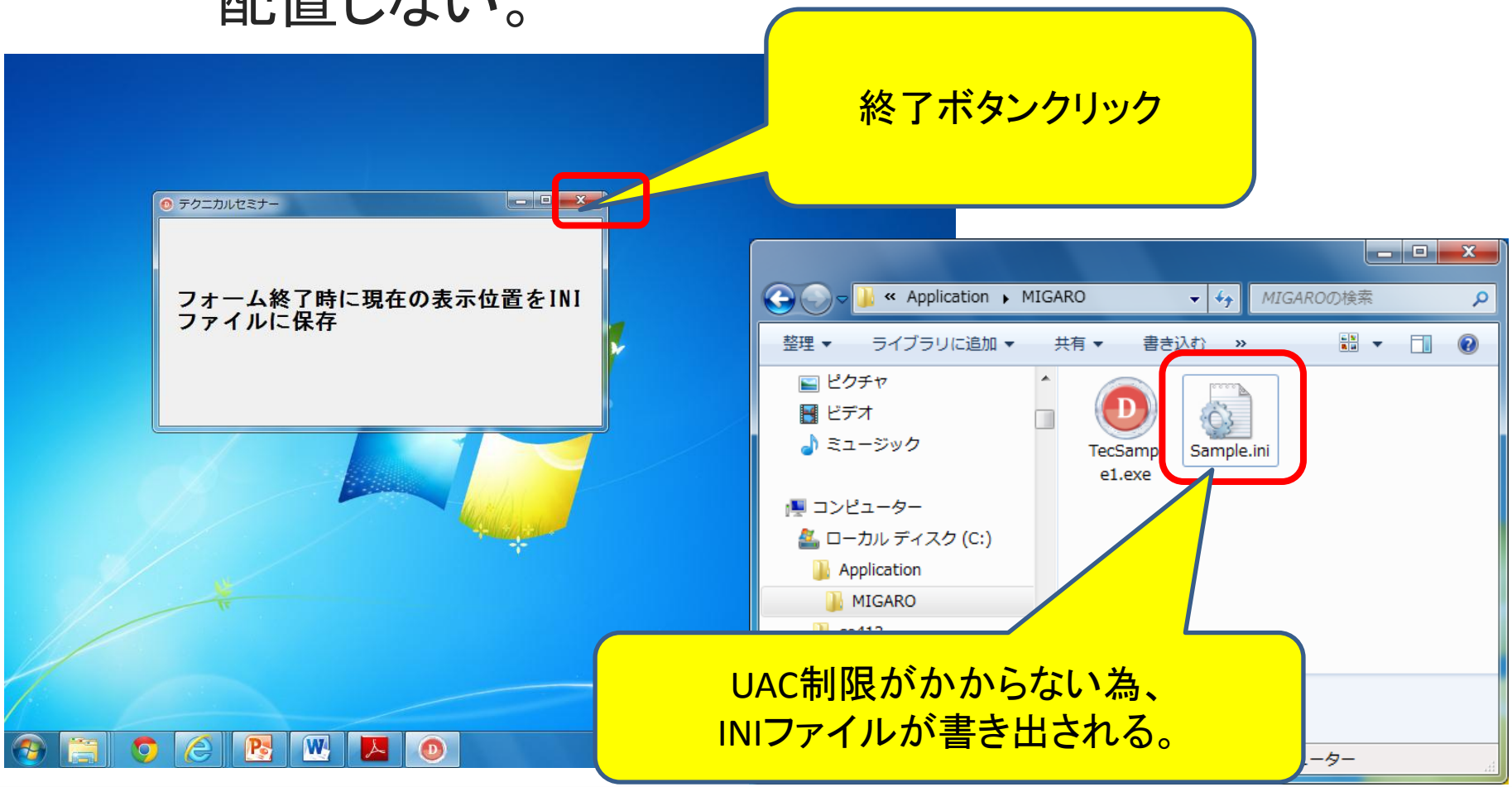
互換性ファイル(VirtualStore)



- 一見便利そうだが、互換性ファイルは
  ユーザー毎の登録となる為トラブルの元！

UAC　対応①

- アプリケーションをUAC対象となるフォルダに配置しない。



UAC　対応②

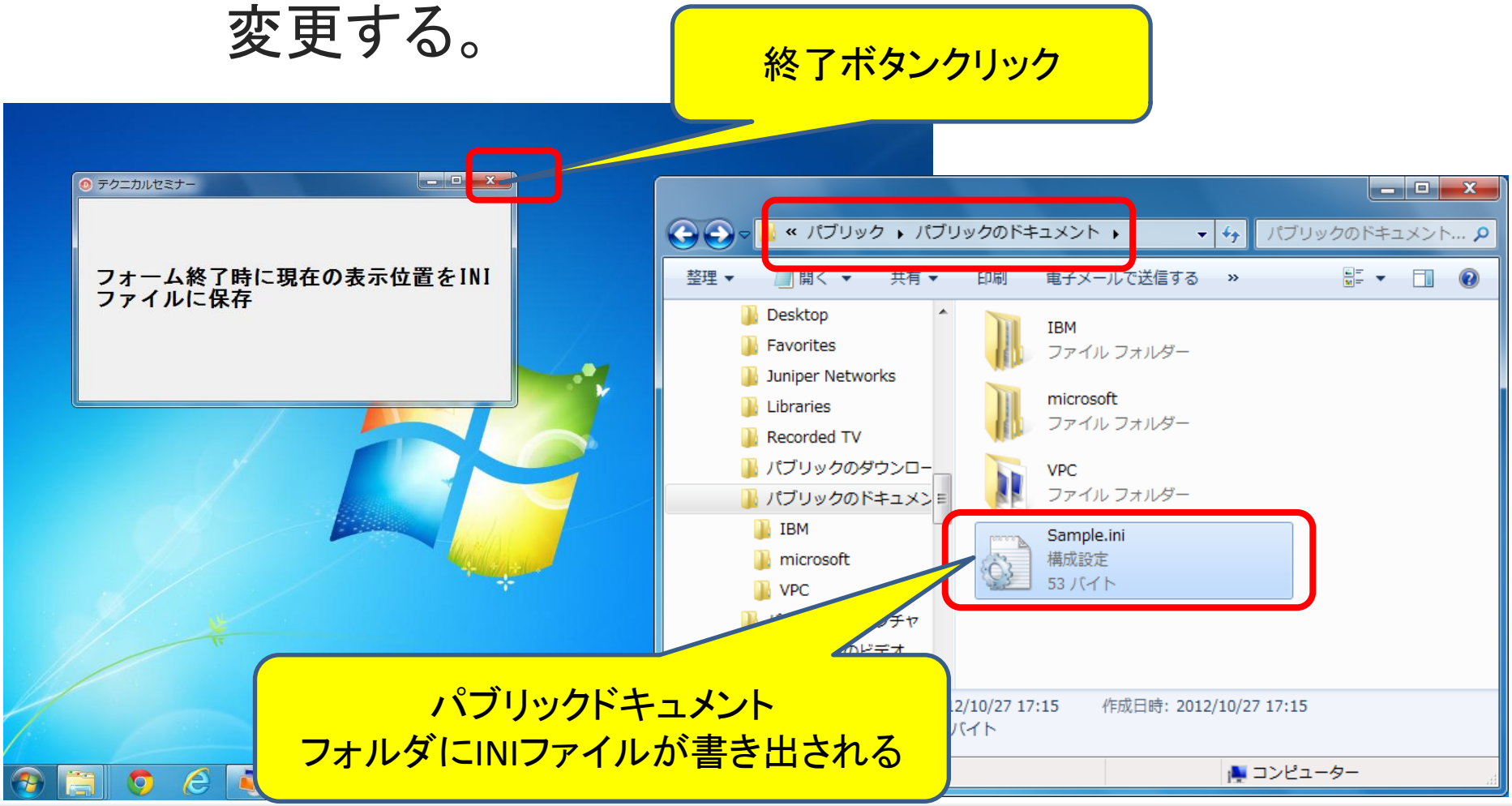
- INIファイルの書き込み先をユーザーフォルダに変更する。

```
procedure TForm1.FormDestroy(Sender: TObject);
var
  sPath: String;       // Path
  iniFile: TIniFile;   // INI情報
begin
  //保存先にパブリックドキュメントフォルダを指定
  sPath := 'C:¥Users¥Public¥Documents¥';
  //INIファイルアクセスオブジェクト生成
  iniFile := TIniFile.Create(sPath + 'Sample.ini');
  try
    //INIファイルにフォームの位置サイズを保存
    iniFile.WriteInteger('Setting', 'Top', Self.Top);
    iniFile.WriteInteger('Setting', 'Left', Self.Left);
    iniFile.WriteInteger('Setting', 'Width', Self.Width);
    iniFile.WriteInteger('Setting', 'Height', Self.Height);
  finally
    //INIファイルアクセスオブジェクト破棄
    iniFile.Free;
  end;
end;
```

全てのユーザーが使用できるパブリックフォルダを指定（パブリックドキュメント）

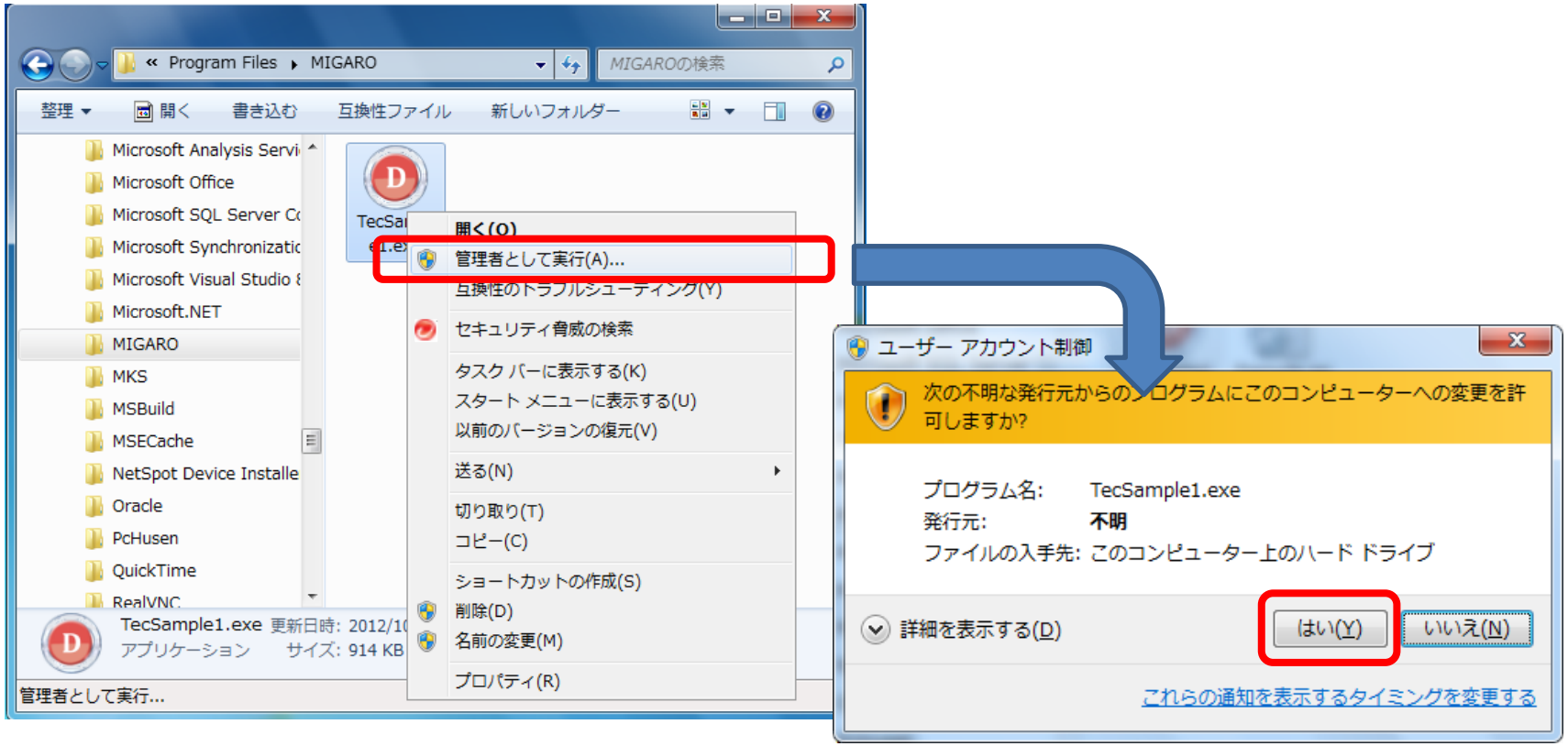
UAC　対応②

- INIファイルの書き込み先をユーザーフォルダに変更する。



UAC　対応③

- アプリケーションを管理者に昇格して実行する。



- 明示的に「管理者として実行」を指示しないといけないか？

Delphi/400 ver.XE で作成されたExe

- アプリケーションの実行権限レベルを宣言した情報がExeに含まれる(マニフェスト)
- **レベルは、”asInvoker”が設定されている**

レベル	概要
asInvoker (デフォルト)	アプリケーションは、アプリケーションを開始したプロセスと同じアクセス許可で実行される。 [管理者として実行] を選択すると、アプリケーションをより高いアクセス許可に昇格させることが可能。
highestAvailable	アプリケーションは、可能な限り高いアクセス許可レベルで実行される。 使用可能な最も高いアクセス許可レベルが、開始したプロセスのレベルより高い場合は、資格情報の入力が求められる。
requireAdministrator	アプリケーションは管理者のアクセス許可で実行される。 アプリケーションを開始するユーザーは、管理者グループのメンバーである必要がある。 開始したプロセスが管理者のアクセス許可で実行されない場合は、資格情報の入力が求められる。

実行権限レベル変更（Delphi/400 ver.XE）

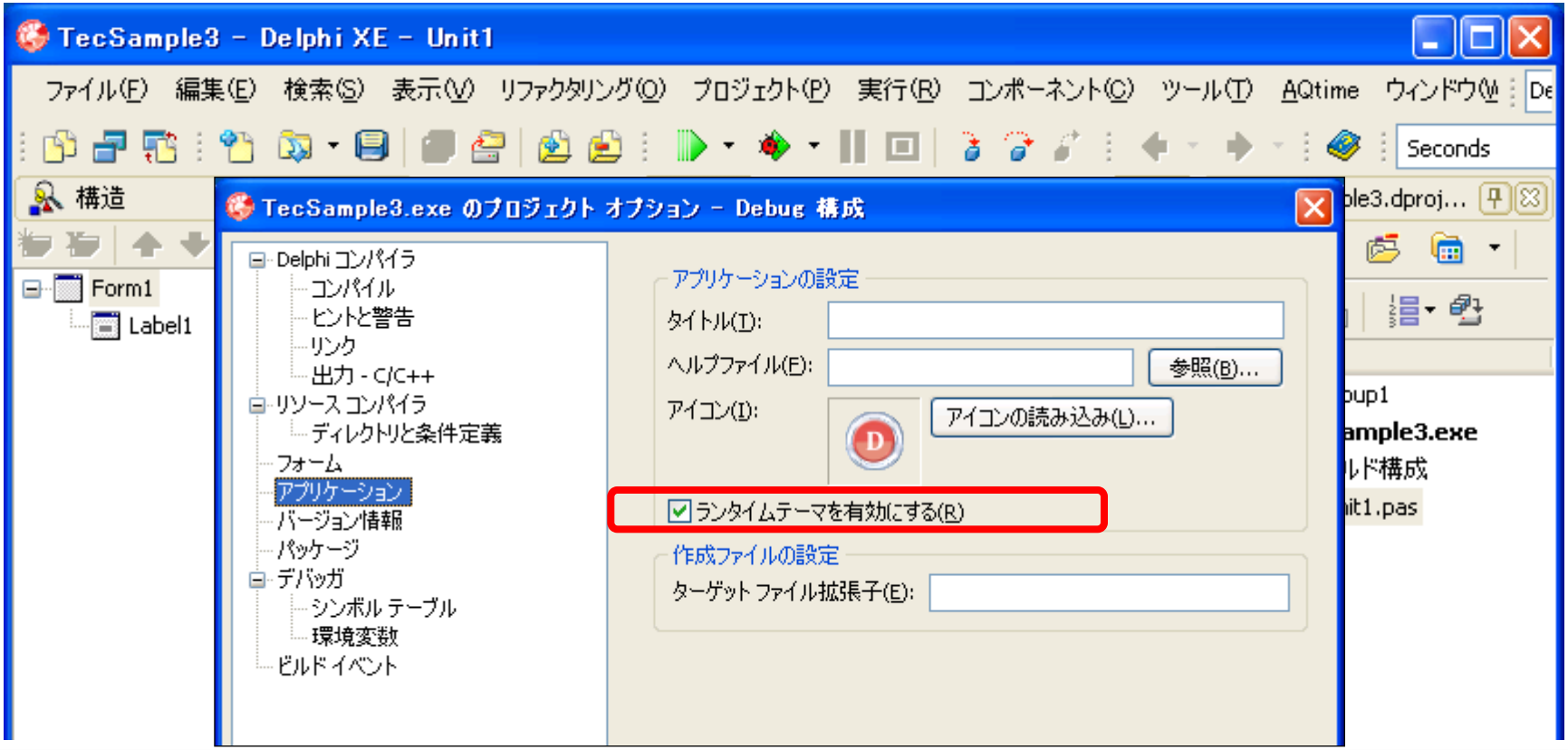
- 権限レベルを**requireAdministrator**へ変更する。
- 作業する際にリソースエディタを使用。
  - XN Resource Editor
    http://cc.embarcadero.com/item/25783

- 変更手順

STEP1	リソースエディタを使用して、Exeファイルからマニフェストを抽出する
STEP2	マニフェストをテキストエディタで編集し権限レベルを変更する
STEP3	編集したマニフェストをプロジェクトに取込み、再構築する

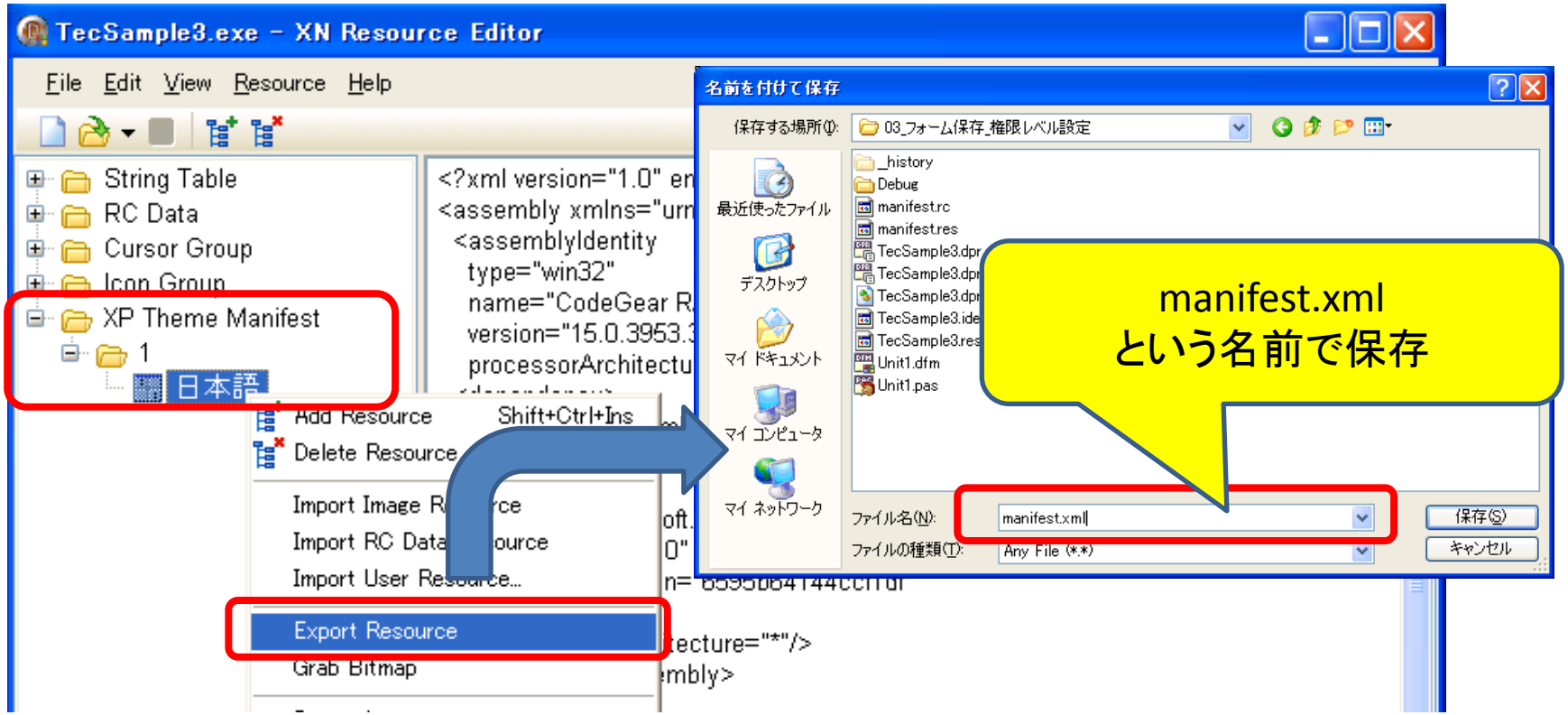
STEP1： マニフェストを抽出

- プロジェクト⇒オプションより、アプリケーションの『ランタイムテーマを有効にする』にチェックを付けて、プロジェクトをコンパイル。



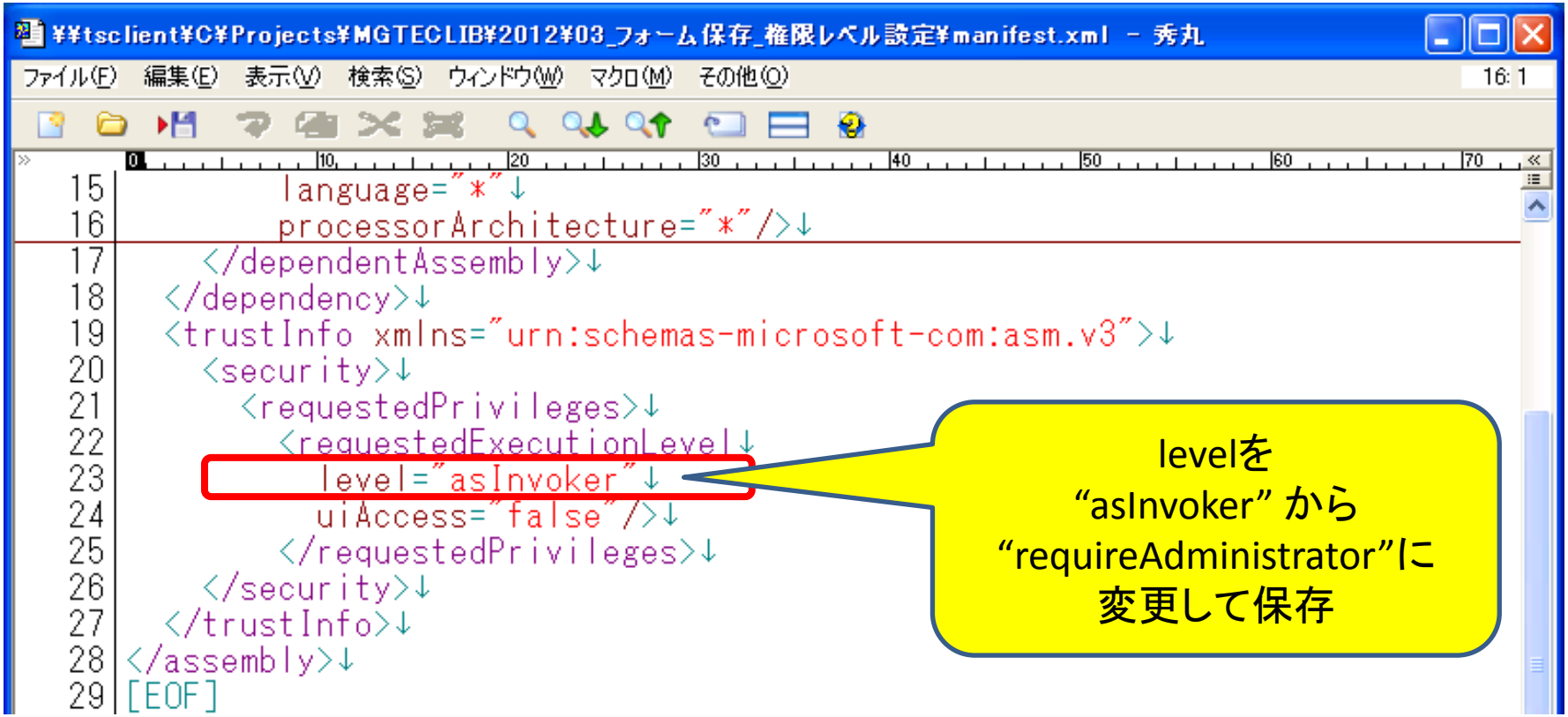
STEP1： マニフェストを抽出

- 作成されたExeファイルを「XN Resource Editor」で開き、
  [XP Theme Manifest]⇒[1]⇒[日本語]を選択。
  右クリックし、[Export Resource]で「manifest.xml」を出力。



STEP2： マニフェストを編集

- UTF-8形式で保存できるテキストエディタで、「manifest.xml」を開き、[requestedExectionLevel]タグのlevel値を”asInvoker”から”requireAdministrator”に変更。

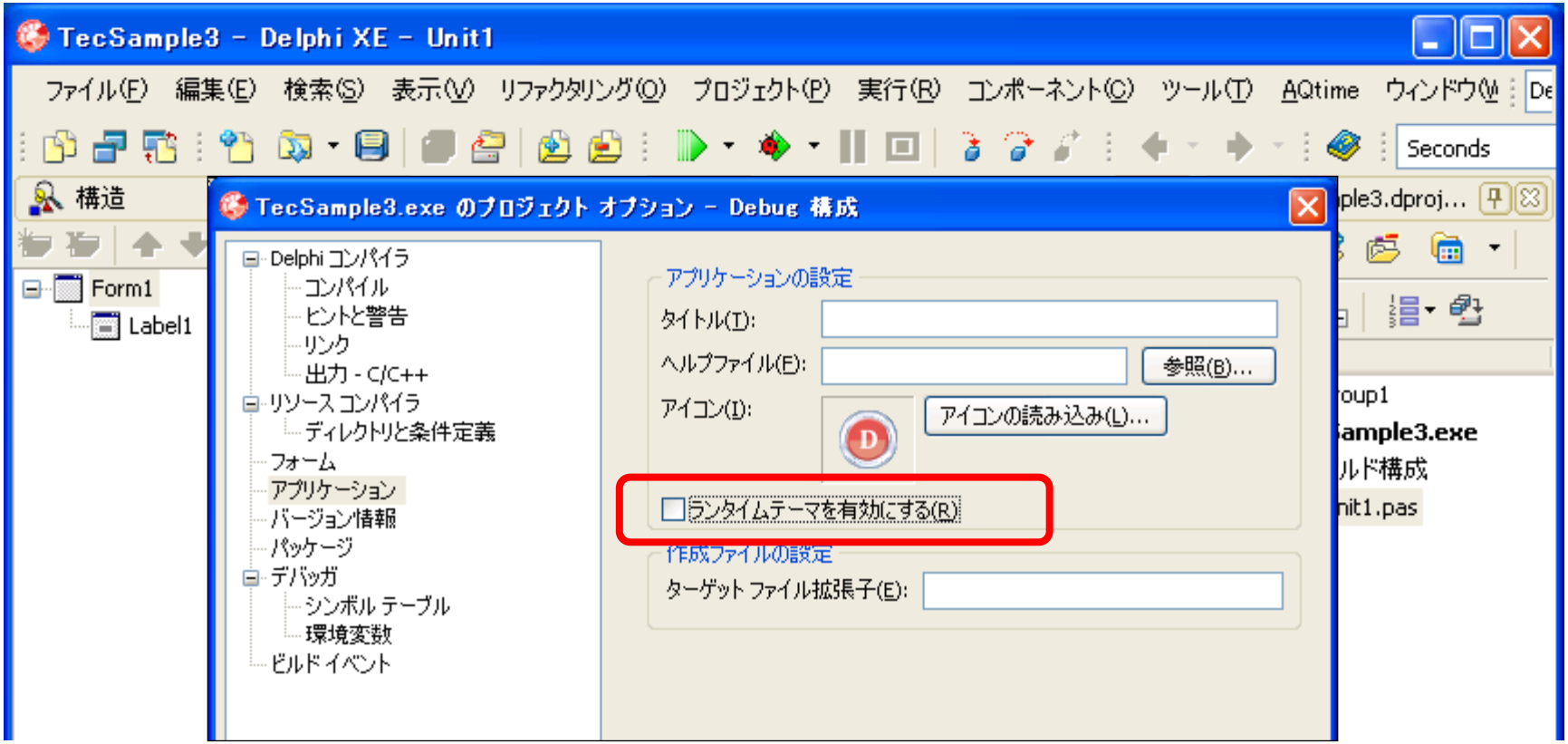


STEP2： マニフェストを編集

- テキストエディタで、下記入力を行い、「manifest.rc」と名前を付けてUTF-8形式で保存。(リソースファイル作成)

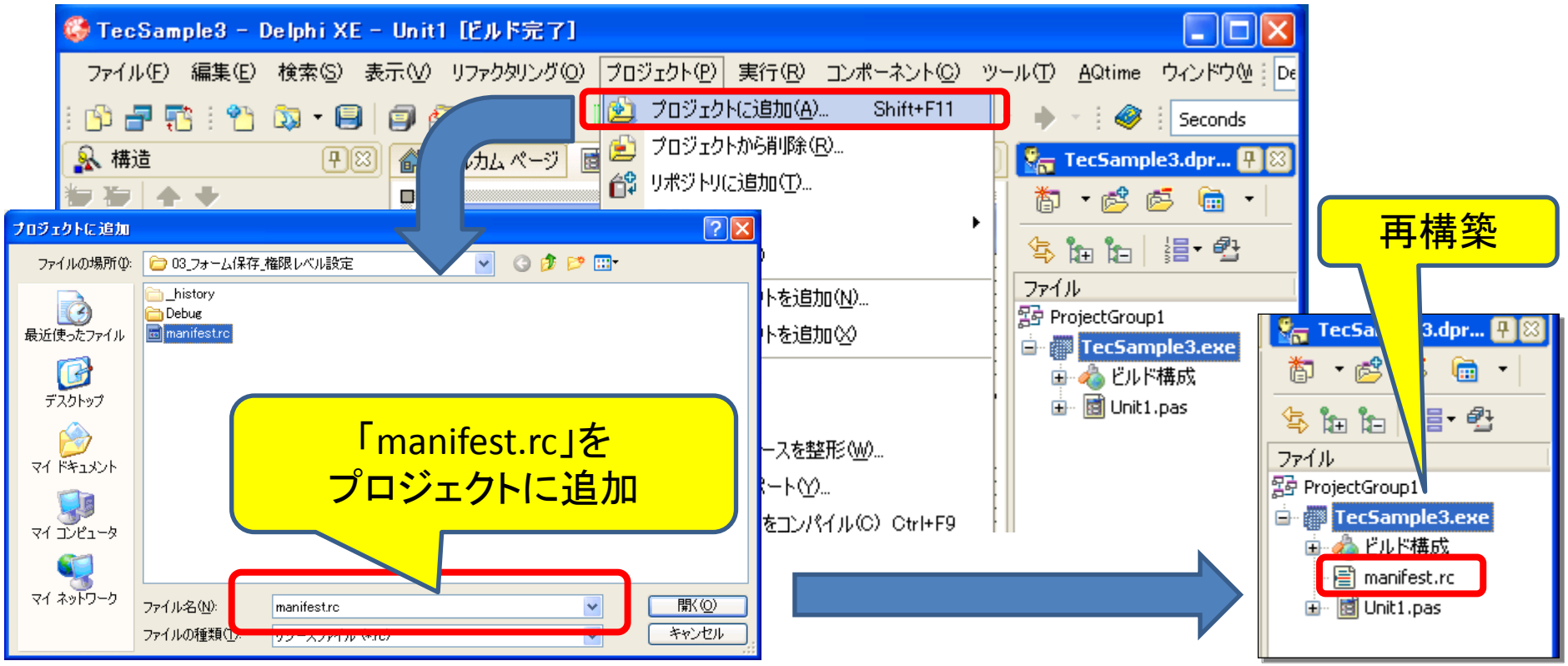
STEP3： マニフェストの取込、再構築

- プロジェクト⇒オプションより、アプリケーションの
『ランタイムテーマを有効にする』のチェックをOFFに変更。



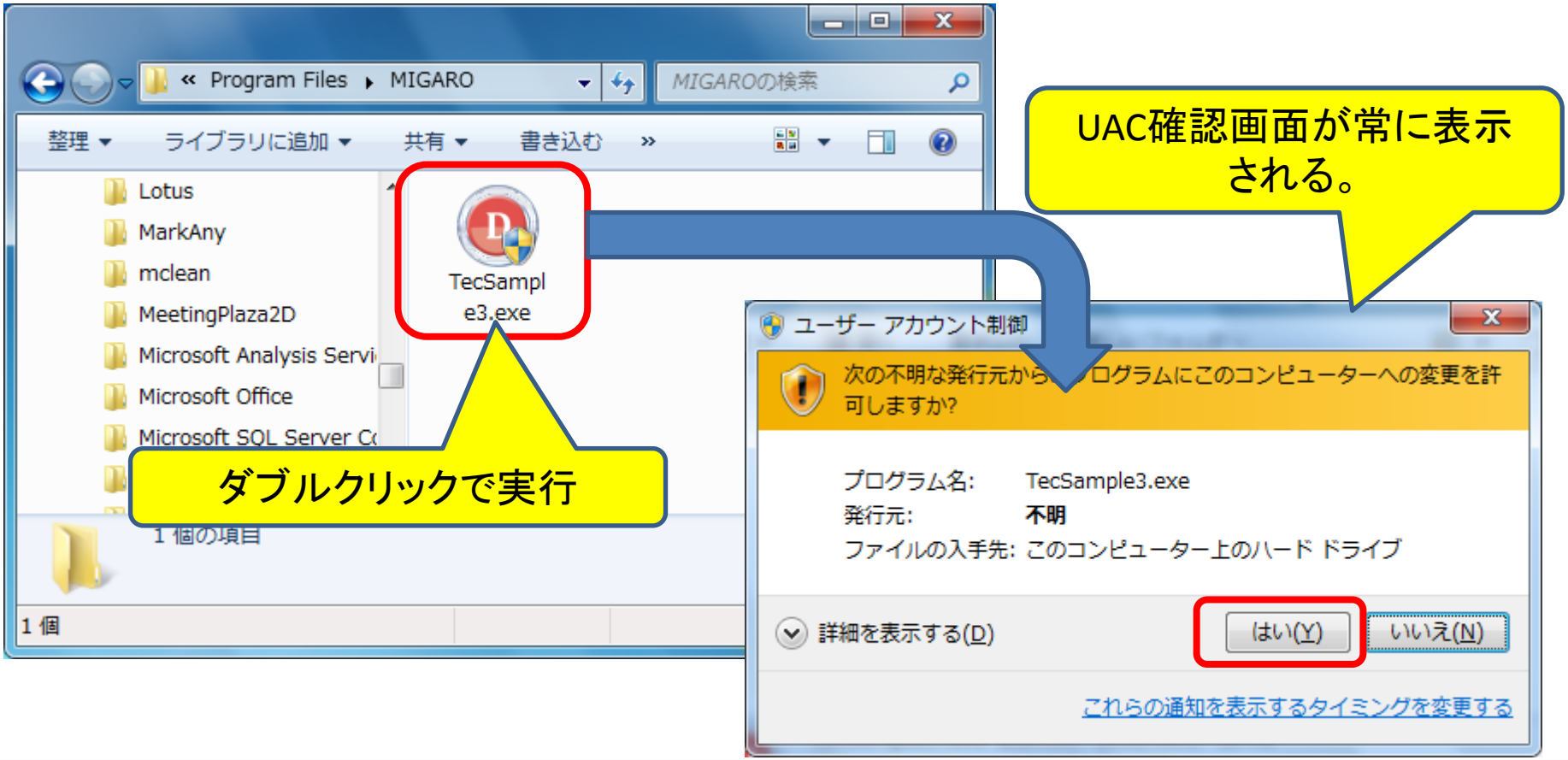
STEP3： マニフェストの取込、再構築

- プロジェクト⇒プロジェクトに追加より、「manifest.rc」をプロジェクトに追加し、プロジェクトを再構築。



管理者権限要求アプリケーションの完成

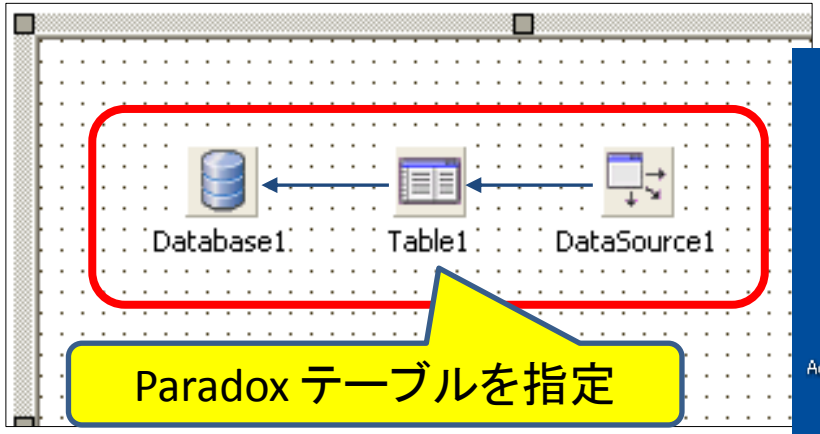
- Exeファイルに管理者権限要求を表す「シールド」アイコンが付与。



（2）BDEアプリケーション

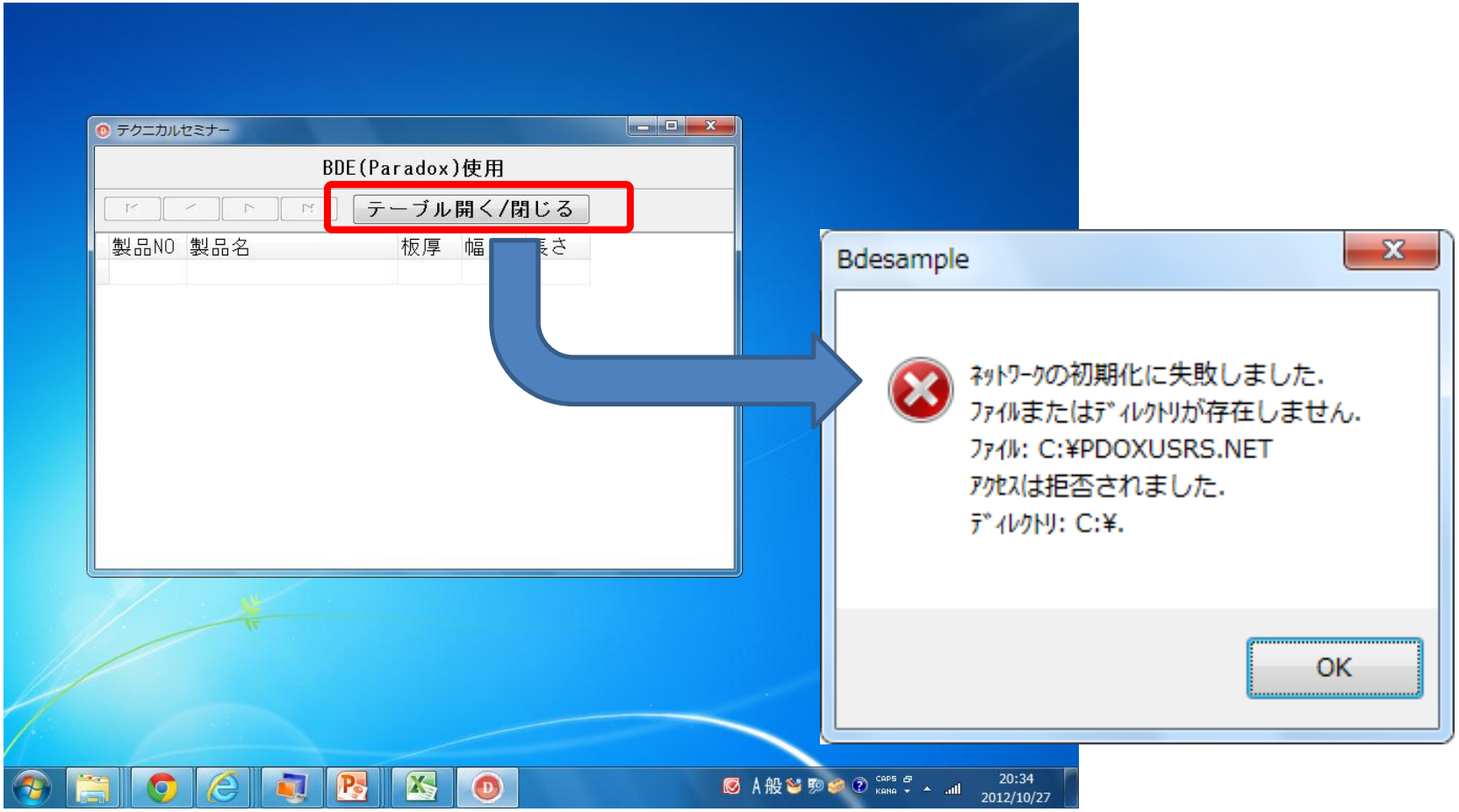
BDEアプリケーション

- DB2/400へのアクセスのみならず、OracleやSQLServer等へのアクセスでも頻繁に利用。
- Paradox(ローカルデータベース)は、簡易に使用でき便利である。



BDEアプリケーション

- Windows7で実行すると、エラーが発生。



BDEアプリケーション

- Windows7では、C:¥直下にファイルを書くことができない。
- Paradoxは、通常C:¥直下に”PROXUSRS.NET”（BDEネットワークコントロールファイル）を作成する。

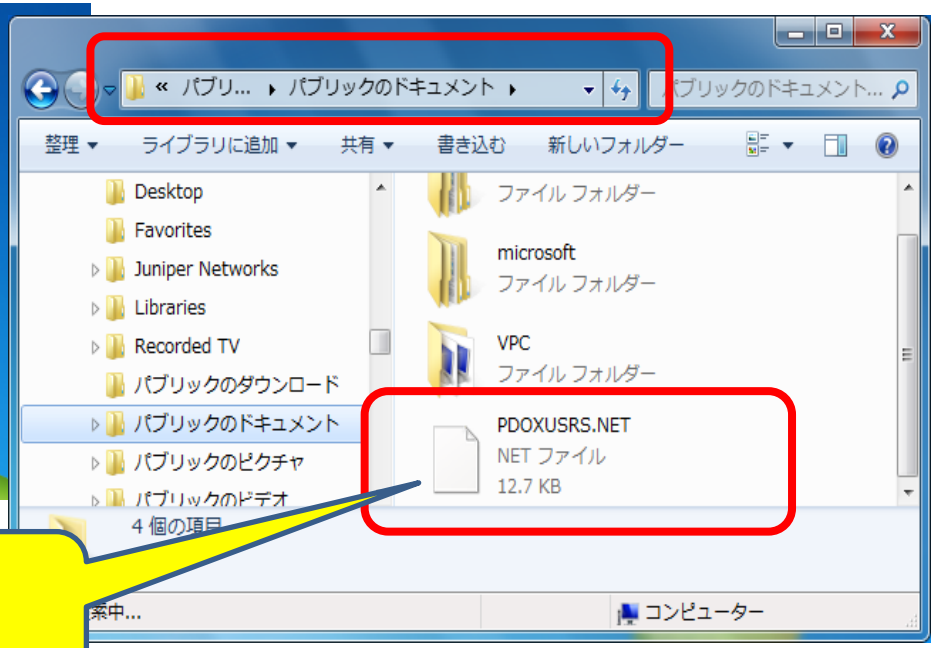
- **Session変数のNetFileDirプロパティに書き込み可能なフォルダを指定すれば良い。**

BDEアプリケーション

実装例

```
procedure TdmMain.DataModuleCreate(Sender: TObject);
begin
  //BDEネットワークコントロール保管場所指定
  Session.NetFileDir := 'C:¥Users¥Public¥Documents¥';
end;
```

全てのユーザーが使用できる
パブリックフォルダを指定
（パブリックドキュメント）



パブリックドキュメント
フォルダにファイルが書き出される

（３）特殊フォルダの取扱い

■ 特殊フォルダの取扱い

● P4-25のソース

```
procedure TForm1.FormDestroy(Sender: TObject);
var
  sPath: String;       // Path
  iniFile: TIniFile;   // INI情報
begin
  //保存先にパブリックドキュメントフォルダを指定
  sPath := 'C:¥Users¥Public¥Documents¥';
  //INIファイルアクセスオブジェクト生成
  iniFile := TIniFile.Create(sPath + 'Sample.ini');

  （（（以下省略））））
```

全てのユーザーが使用できる
パブリックフォルダを指定
（パブリックドキュメント）

- Windows7での、[パブリックドキュメント]は確かに
  ”C:¥Users¥Public¥Documents”だが、
  WindowsXPでの、[共有ドキュメント]は、
  “C:¥Documents and Settings¥All Users¥Documents”である。

  ⇒ フォルダの決め打ちは、危険！

SHGetSpecialFolderLocation API

- 特殊フォルダをPathを取得するAPI
- ShlObj ユニットに定義
- 特殊フォルダは、CSIDL値という定数で定義

```
function GetCommonDocFolder: String;
var
  PID   : PItemIDList;
  pPath: PChar;
begin
  //SHGetSpecialFolderLocation APIより情報取得
  SHGetSpecialFolderLocation(Application.Handle,
    CSIDL_COMMON_DOCUMENTS  PID);

  GetMem(pPath, MAX_PATH);
  try
    //フォルダ名の取得（末尾に"¥"を付加）
    SHGetPathFromIDList(PID, pPath);
    Result := IncludeTrailingPathDelimiter(Trim(pPath));
  finally
    FreeMem(pPath);
  end;
end;
```

共有（パブリック）フォルダの
CSIDL値（定数）は、
CSIDL_COMMON_DOCUMENTS

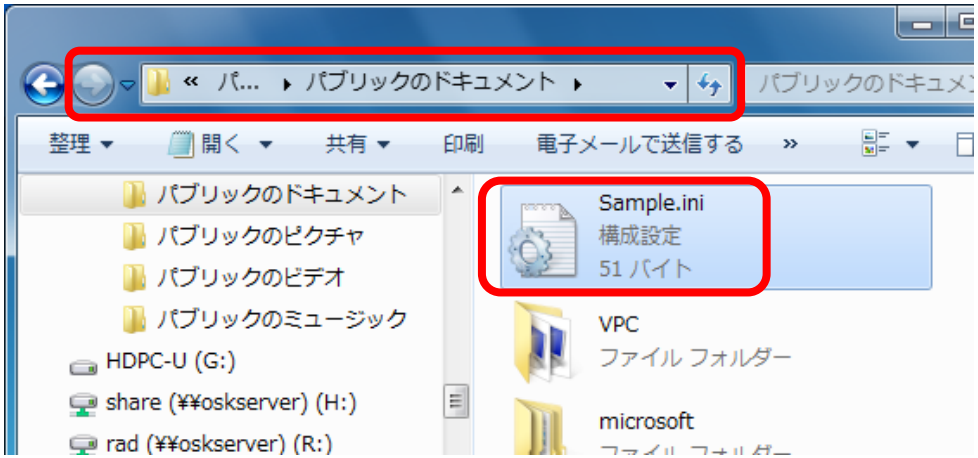
■ 特殊フォルダ取得の使用例

- P4-25のソースを改良

```
procedure TForm1.FormDestroy(Sender: TObject);
var
  sPath: String;       // Path
  iniFile: TIniFile;   // INI情報
begin
  //保存先にパブリックドキュメントフォルダを指定
  sPath := GetCommonDocFolder;
  //INIファイルアクセスオブジェクト生成
  iniFile := TIniFile.Create(sPath + 'Sample.ini');

  （（（以下省略））
```

全てのユーザーが使用できる
パブリックフォルダを指定
（パブリックドキュメント）

GetCommonDocFolder
関数（P4-44）を使用



4. Windows7対応版 Delphi/400 versionXEを使用するメリット

- Delphi/400　Version XEのメリット
  - Windows2000～Windows7　と幅広いバージョンでのアプリケーションの動作を正式サポート
  - Windows7の新機能がフル活用可能
    - Aero
    - ジェスチャー
    - リボン
    - タスクバー
  - RADツールとしての進化
    - VCL for the Web
      - Ajaxに対応し、動的な画面更新が容易に可能
      - コネクションプールを使用したDBセッションが作成可能
    - DataSnap
      - 3層アプリケーションがウィザードで容易に作成可能
    - テクニカルセミナー＆テクニカルレポートにて随時　情報公開！

Aero

Aero効果の使用

Applicaton.MainFormOnTaskBar := True;

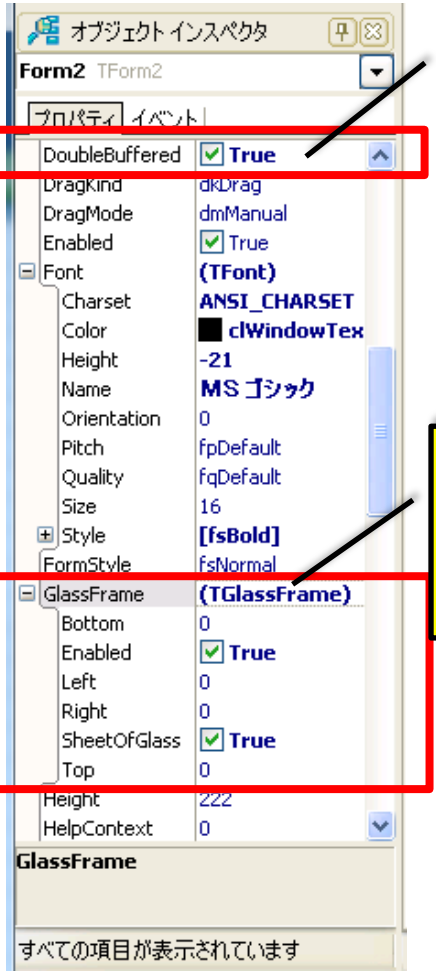
```
program Project1;

uses
  Forms,
  Unit1 in 'Unit1.pas' {Form1};

{$R *.res}

begin
  Application.Initialize;
  Application.MainFormOnTaskbar := True;
  Application.CreateForm(TForm1, Form1);
  Application.Run;
end.
```

新規プロジェクトの場合、
デフォルトでTrueがセット
旧バージョンから移行する場合、この行を追加する。

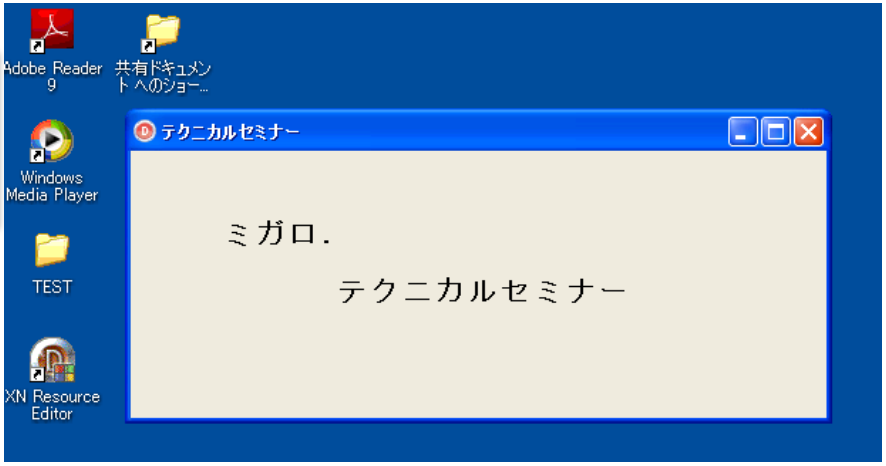
- ライブ タスクバーサムネイル、動的ウィンドウ、Windows フリップ 等　が利用可能

Aeroの例（グラスウィンドウ）

- Windowを半透明表示

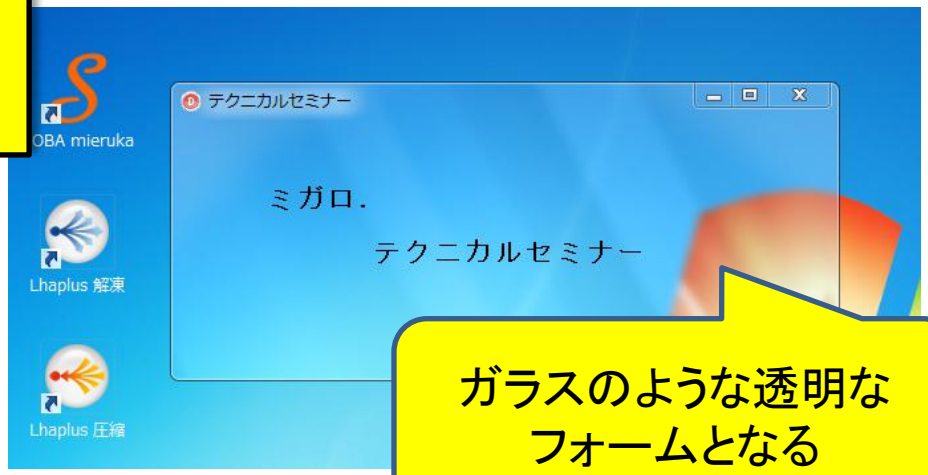


DoubleBufferedを
Trueに設定



WindowsXPで実行

GlassFrame
Enbaled := True
SheetOfGlass := True

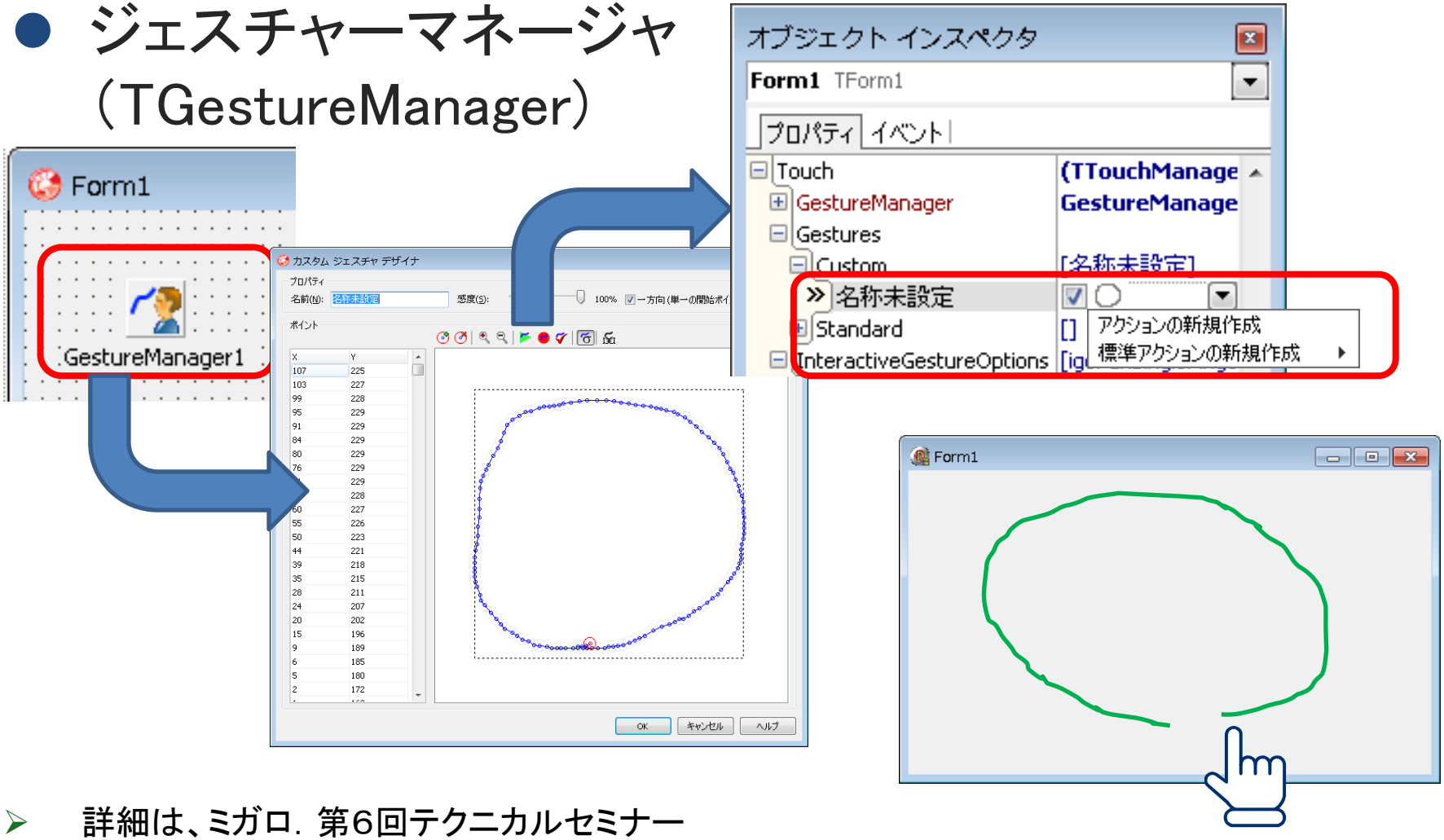


Windows7で実行

ガラスのような透明な
フォームとなる

ジェスチャー

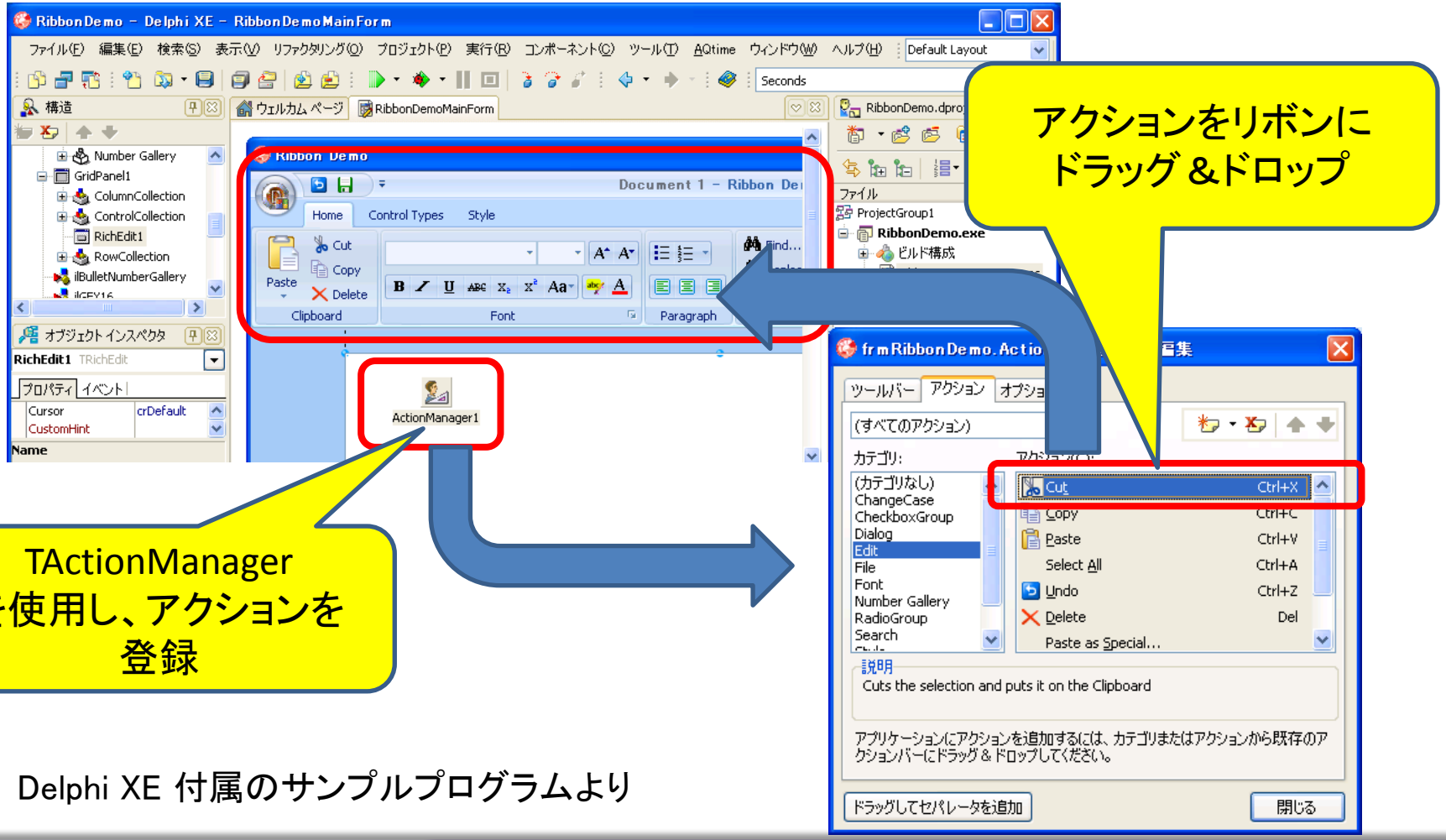
- ジェスチャーマネージャ（TGestureManager）



- 詳細は、ミガロ．第6回テクニカルセミナー
  『知って得する！現役ヘルプデスクが答えるDelphiテクニカルエッセンス6.0』 にて紹介

リボン

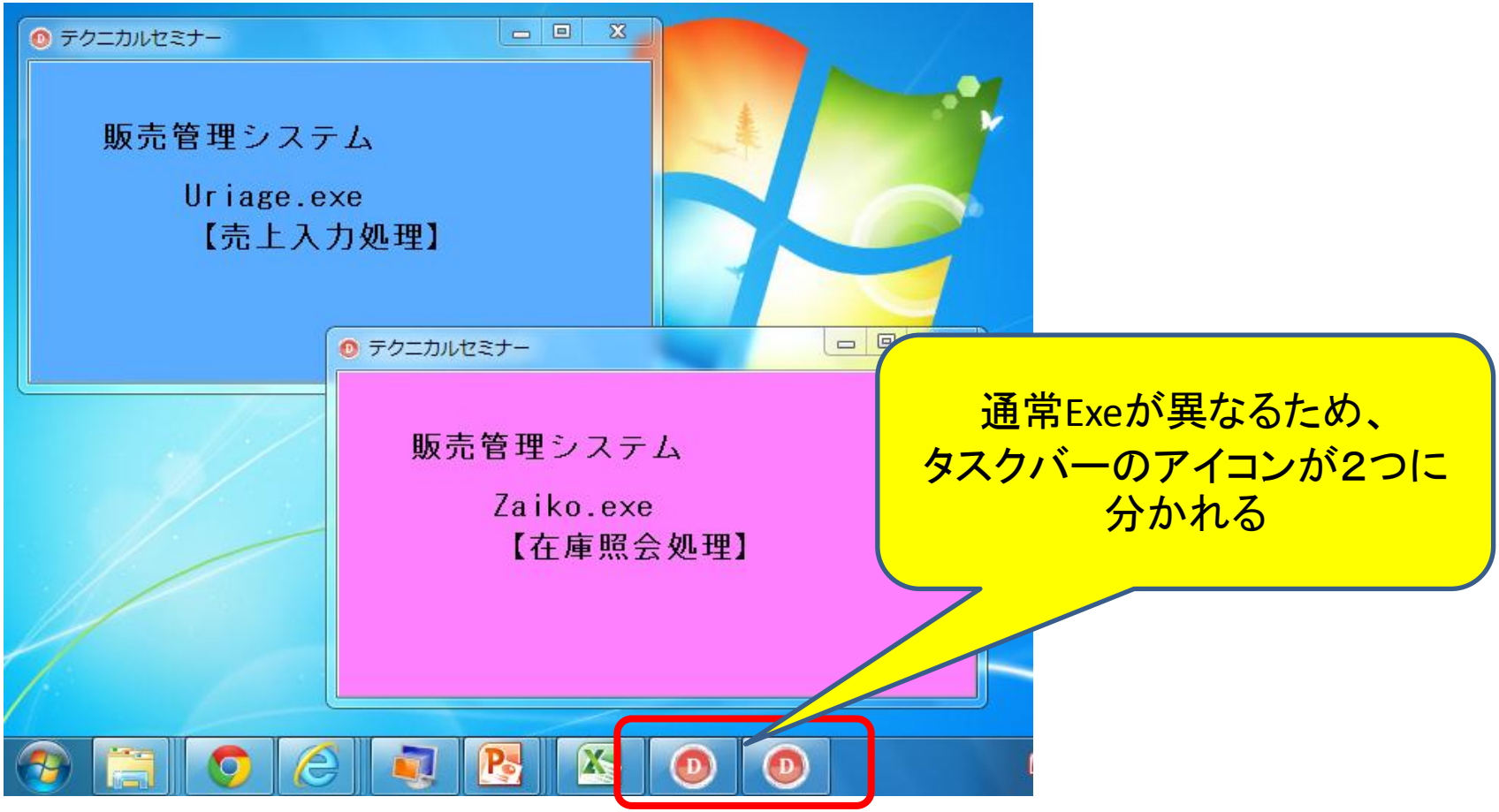
リボンコントロール（TRibbon）



- Delphi XE 付属のサンプルプログラムより

タスクバー

- 同じ業務アプリケーションが複数EXEの場合



タスクバーのグループ化

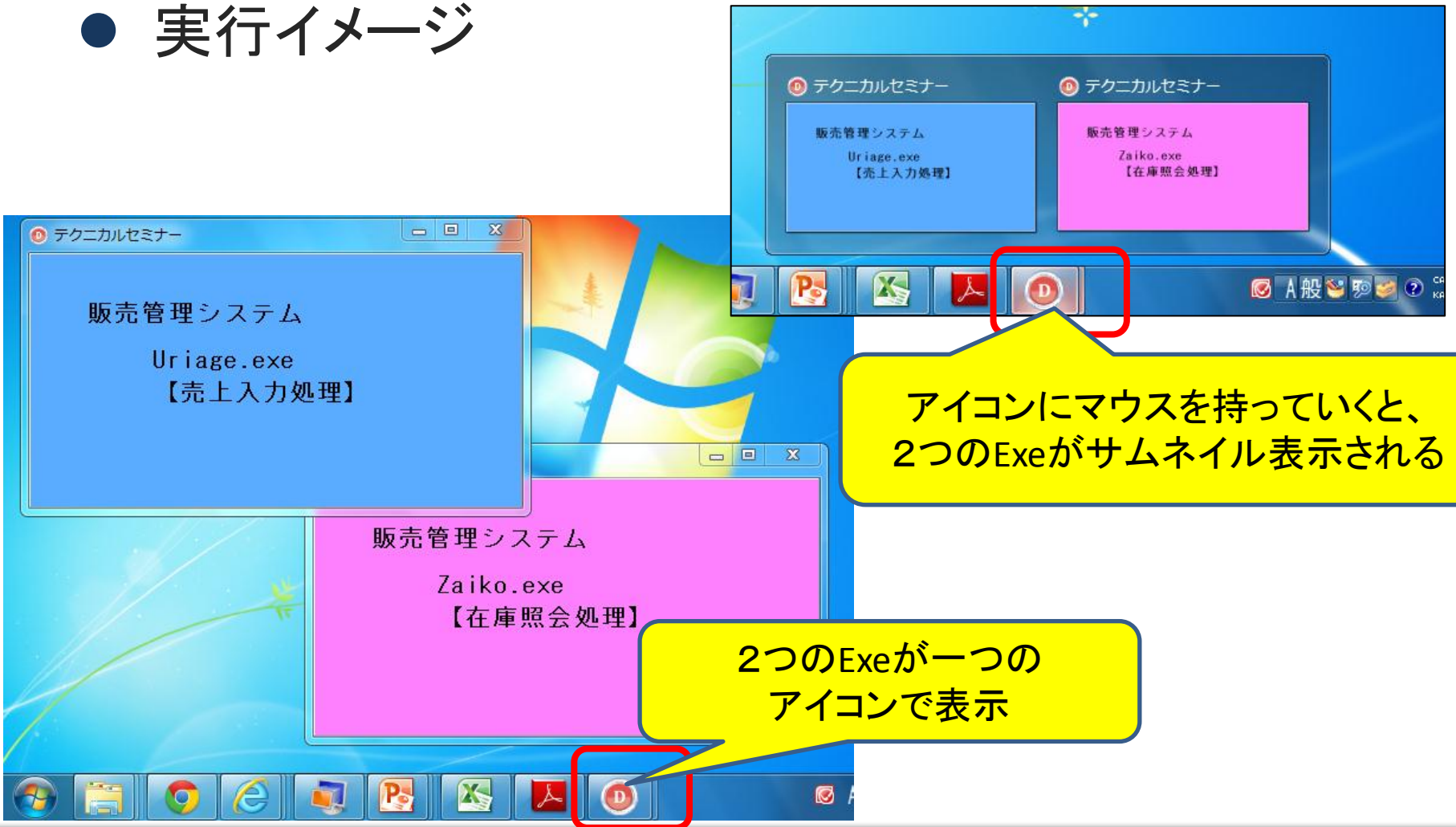
- SetCurrentProcessExplicitAppUserModelID API
  - タスクをグループ化する為の一意なIDをセット
    “CompanyName．ProductName．(SubProduct)．(VersionInformation)”形式
  - 同じIDのプログラムは、全て同じグループとなる
  - ShlObj ユニットに定義

```
uses ShlObj;
  …
procedure TForm1.FormCreate(Sender: TObject);
begin
  //アプリケーションユーザーモデルIDの指定
  SetCurrentProcessExplicitAppUserModelID('MIGARO.HANBAI');
end;
```

同じグループのプログラムに全て”MIGARO.HANBAI”というIDを付与

タスクバーのグループ化

- 実行イメージ



5. まとめ

まとめ

- UACへの対応方法
  - アプリケーション配置フォルダの工夫
  - ユーザーフォルダへのファイル書き出し
  - マニフェストの作成
- BDEの対応
  - NetFileDirプロパティの設定方法
- 特殊フォルダ
  - 特殊フォルダ取得方法
- Delphi/400　ver.XEのメリット
  - Windows7新機能のフル活用（Aeroの例）
  - タスクバーのグループ化方法

ご清聴ありがとうございました