Automatic migration of Gupta applications to .NET

Back to the Future

Transforming a Gupta application created by Team Developer into a .NET application at the press of a button has been the dream of many a user, especially since the takeover of Gupta by Unify. An international association of software houses calling itself "The Porting Project" provides a service that comes very close to fulfilling this ideal.

It is always difficult to part company with a good friend. The world of software is no different. Thousands of database applications involving billions of lines of code are still based on 4GL technology from Gupta, even when the underlying database is no longer SQLBase but perhaps Microsoft SQL server or Oracle. Since the mid-nineties, SAL (Scalable Application Language, previously SQLWindows Application Language) and Team Developer have been used as a toolkit for efficiently coding database access and for creating reasonable graphical interfaces. Up until recently, this was the reason why companies and their IT departments relied on Gupta's development environment.

However, change is in the air, brought about in part by the announcement in the middle of last year of Gupta's takeover by Unify. The portfolio of both companies previously included a development environment as well as a database backend, i.e. Team Developer and SQLBase, and NXj and DataServer. Unify's NXj product has a series of features that are not provided in Team Developer. As a result, Unify is promoting this product as the one with the more promising future. Team Developer 2006 has now been released, but under the name of 5.0. Discussions in the relevant newsgroups have generally been critical about its quality.

Regardless of the above, many companies have been looking – some for many years – for more future-proof technologies with which to maintain and develop their applications, especially since it is becoming increasingly difficult to find good 4GL developers. The Unify deal is just one more reason to assess the situation more quickly and to find a short-term route out of this technological dead-end.

An assortment of business-critical applications with a history of change and expansion cannot be migrated overnight but instead poses a huge challenge to company management and its IT department. The biggest challenge of all is to minimize any possible risk. There is something to the saying "if it works, don't fix it". Nevertheless, touching the code cannot always be avoided if the system is to run in the future. Not only must a suitable target platform be chosen but there is also the thought-provoking question about how to tackle migration.

Reconstruction or renovation?

Should all of the applications be developed from scratch on the target platform or is an automatic conversion of existing code preferable? There is no clear answer to this difficult question, which requires prior analysis.

Small organizations with 10 to 15 applications and 10 to 30 thousand lines of code often decide for the first option. Freed from all inherited burdens and able to make use of current technology, a system can be created that can be understood and maintained by the existing development team. On the other hand, all mistakes can be repeated in in the process, which give rise to a corresponding increase in cost for the test and acceptance phases. Moreover, in a "live system" the new and the old product must run in parallel during migration to limit any

The Porting Project (PPJ)

The Porting Project is a custom development that adopts a predominately automated approach for the reliable migration of Team Developer applications (Gupta, SQLWindows) to C# or VB.NET 2.0 and Visual Studio .NET.

It is a blend of software technologies, expertise, support and a worldwide active network of associated professional organizations. The number of companies involved has deliberately been kept low. Fecher, a company based in Hesse, is responsible for German speaking countries and Eastern Europe. It has subsidiaries in Berlin, Munich, Paderborn, Vienna and Zurich as well as an offshore development centre in Oradea / Romania. Fecher became known through its Team Developer add-ons such as buildingBLOCKS and is now the first German delivery partner for Microsoft's NXT campaign.

Contact: **www.fecher.eu**

Summary

Author

Dr. Bernhard Röhrig is a freelance consultant for databases, networks and operating systems, as well as the author of numerous specialist books. He can be contacted via www.roehrig.com.

dotnetpro.code
A0706Gupta.NET

Languages SAL, C#

Prerequisites Team Developer, VS.NET

Technology Migration of Gupta 4GL applications to .NET

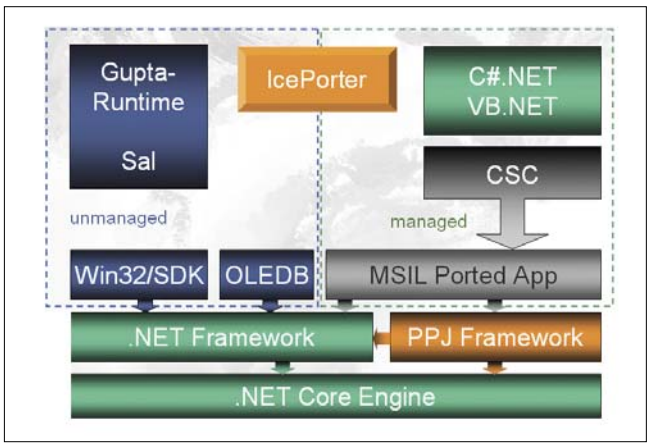

Figure 1 The PPJ/FW porting framework supports automatic migration of Gupta applications to .NET.

possible damage caused by faults in the new system. This can give rise to additional risk factors and sources of error.

This option is out of the question for most large projects. The companies affected usually look for an automated solution with the following criteria in mind:

- *Costs*: a completely new development is an order of magnitude more costly than an automated migration
- *Time factor*: developing and implementing new software takes considerably longer than using a porting tool that works reliably
- *Risks*: in addition to these two factors, there is the risk of design and coding mistakes that always accompany any new development

Suppliers of automatic code porting tools promise to minimize the above factors with their products and therefore make the migration process more manageable and less daunting.

Anyone for iced tea?

An international association of experienced software houses called the Ice Tea Group (ITG), has been involved in porting SAL applications to .NET for some years. ITG has developed tools such as IntelliSal, XSal2 and IntelliDoc. Fecher is its business partner for Germany, Austria, Switzerland and Eastern Europe.

The main component provided by the Porting Project is the Ice Porter tool, which performs a fully automatic conversion of existing SQLWindows code into C# or VB.NET 2.0 based on the PPJ/FW porting framework (see figure 1). The conversion produces a result that accurately reflects the source code. This is very important if the client's existing development team is to continue to maintain the ported program code. In addition to the source code, reports produced with the Gupta Report Builder can be ported to either Crystal Reports or List&Label. Fecher and the other Porting Project partners use the Ice Porter in their porting projects. The manual processing that is required means that it does not make sense for the customer to purchase the tool and use it on-site.

One step at a time

An automatic porting project is carried out in phases, in much the same way as normal software has phases of development:

1. Project setup
2. Code generation
3. Code completion
4. Code finalization
5. Test
6. Commissioning
7. Maintenance

During the setup phase, an inventory is made of existing SAL applications, their resources, bitmaps, cursor and configuration files, the SAL libraries, external DLLs and reports. A free preliminary analysis providing a project outline, a risk assessment and an initial estimate of the expected costs can be used to decide what further action should be taken. The subsequent detailed analysis provides information about source code improvement and enables a detailed project plan and final cost to be produced. The end result is a fixed-price porting quotation, for which the customer only has to give the go-ahead.

The following project phases assume that the SAL code has been processed in the project setup phase and partitioned into libraries. A purpose-built inventory tool (see figure 2) is used to help with this

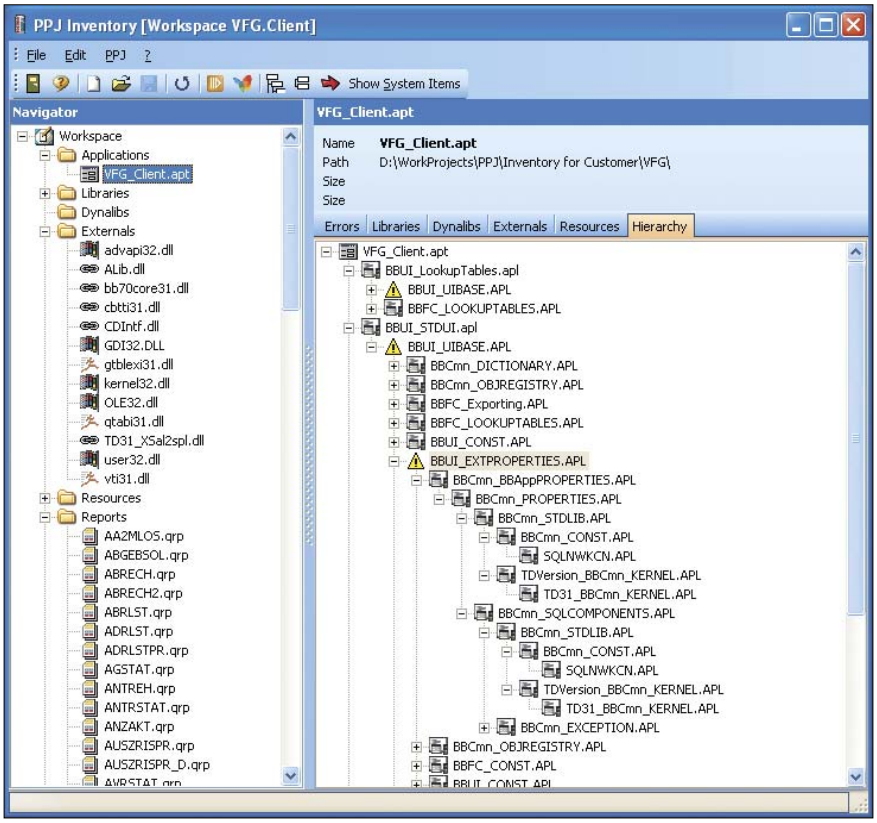

Figure 2 PPJ-Inventory prepares the SAL code for conversion to C#.

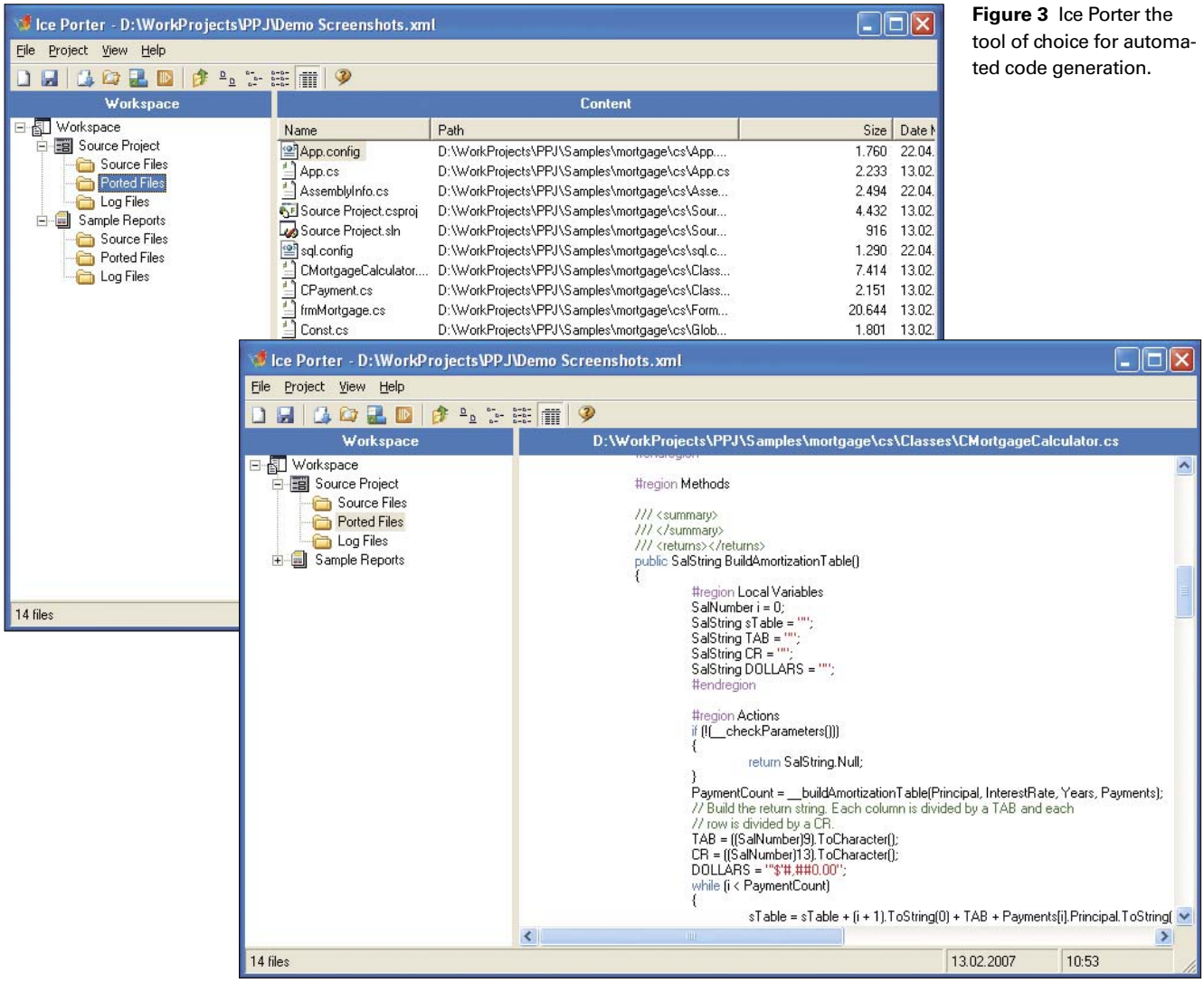

Figure 3 Ice Porter the tool of choice for automated code generation.

work. At the end of this phase the options for the actual porting tool, the Ice Porter, will have been defined. Furthermore, specially adapted Ice Porter filters are developed in order to incorporate individual customizations into the porting.

The iceman cometh

As soon as all conditions have been satisfied in the initial project phase, code generation can begin. The Ice Porter now starts the task of compiling the available SAL applications into (as perfect as possible) C# code, whilst taking into account the resources required – normally between several hundred thousand and millions of lines of code. At the same time, it also generates the required VS.NET project files (see figure 3). Experience has shown that manual rework is needed in both of the subsequent processing phases to ensure that code compiles and executes perfectly. However, thanks to the efficient performance of the Ice Porter and by thorough preparation in the setup phase, this effort can be minimized.

The goal of code completion is a compilable .NET project, no more and no less. This calls for expertise in the source and target languages, the conversion algorithms used by the Ice Porter and the libraries of the support framework. This is an area where fecher can demonstrate its unique strengths. An iterative process is used so that the Ice Porter, for example, by using filters, achieves an optimum translation outcome. If necessary, the SAL code is modified (without affecting its functionality), it is then translated by the Ice Porter, and the output from the Ice Porter is compiled under .NET. This procedure is repeated until the .NET compilation is error-free.

In the subsequent phase of code finalization, the compilable project must be transformed into an application that can be executed and that demonstrates behavior identical to the original application. The combination of specialized porting know-how and experience in both language environments (which is also required in the initial phase), is needed here too. Analysis of Ice Porter warnings at the start of the finalization phase may indicate that some SAL constructs cannot be adequately transformed. Components such as these, which cannot or should not be compiled automatically, will be created manually. At this time the first functional software tests and resulting actions needed to resolve any problems are carried out.

The proof of the pudding ...

Once the ported software is basically working , the subsequent test phase is one of exhaustive functional software-

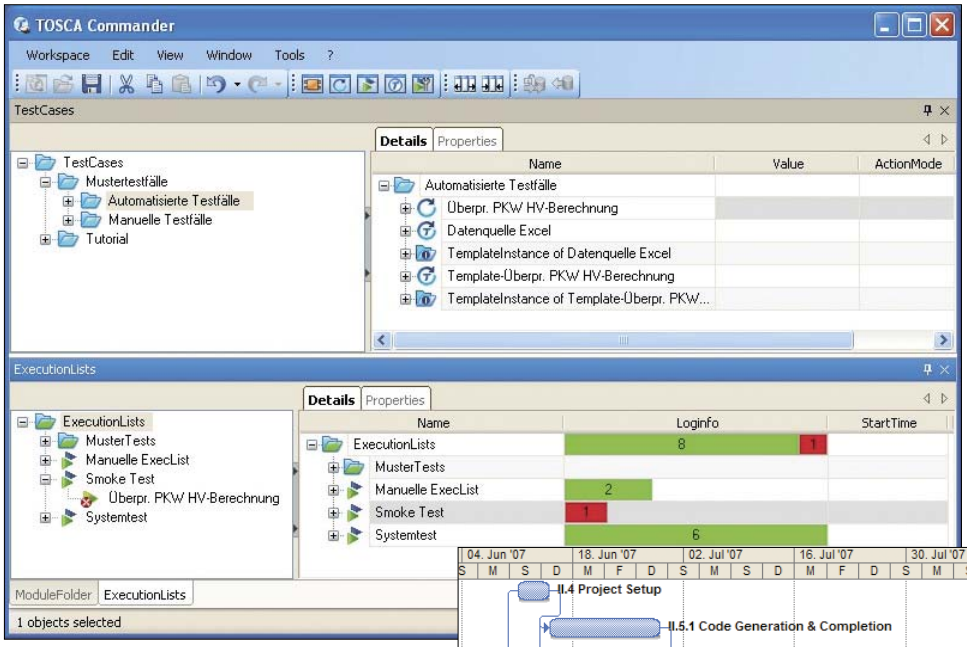

Figure 4 Triton Tosca Commander supports automated application testing.

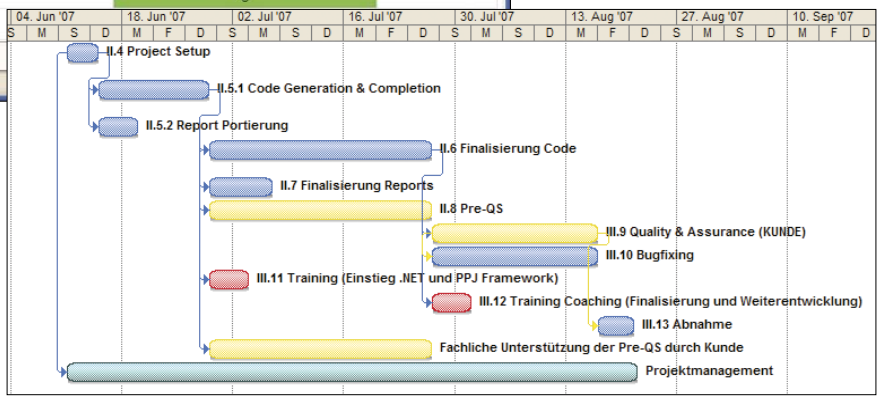

Figure 5 Training measures to support a porting project.

testing and checking whether the actual results match the expected results.

Fecher uses a computer-based software-testing tool to automate the test phase. The company will supply this tool, the Triton Tosca Commander, to customers upon request (see figure 4). This software stores technical information in a neutral form in XML-GUI Maps. Adaptors for SQLWindows and .NET enable the test cases to be used later in .NET after porting is finished.

The following, seventh phase is concerned with maintenance of the completed .NET project. Respecting the code structure during conversion ensures that the user's development team will be able to maintain the code and carry out continued development. Of course, this will necessitate additional training in this area and a certain period of familiarization with .NET technology and tools. Support is provided via fecher's ongoing training program (see figure 5).

The creators of the Porting Project have paid considerable attention to achieving the highest degree of automation for the core process of code transformation. This means staying as true to the source code as possible and not introducing any new constructs during the porting process. This, of course, assumes that the .NET platform totally supports all of the SAL 4GL features. Unfortunately, this support is never 100%. This gives rise to problems during the conversion and a number of challenges that the porting team must overcome.

A goal of the Ice Tea Group is to minimize any risks that may arise. The migration software supports:

- nearly all SAL functions including *SalCompileAndEvaluate()*, a function for executing dynamically created source code
- bind-into variable (100%),
- TableWindows (100%),
- the VT library (100%),
- QO with the exception of QuickGraph
- XSal
- and much more

Together with the ported application the Porting Project Framework is provided – a comprehensive library of .NET functions. It enhances .NET with 4GL features, Grid functions and numerous SAL commands, which cannot be directly mapped into C# or VB.NET. The porting process replaces components such as QuickGraph with alternative libraries under .NET. Report generation with Report Builder, including all features such as formulas and BLOB processing, are modeled using Crystal Reports – without needing to access external libraries, DLLs or COM

Treatment of Windows byRef parameters

The SAL code

```
Set hWndCancel = SalCreateWindow( dlgCancel, hWndForm, bCancel)
```

is compiled automatically to:

```
// TODO: Assign back the unsupported receive window parameter.
hWndCancel = Sal.CreateWindow(typeof(dlgCancel), Sys.hWndForm, bCancel);
```

and must be changed manually to:

```
hWndCancel = Sal.CreateWindow(typeof(dlgCancel), Sys.hWndForm, bCancel);
bCancel = App.dlgCancel.bCancel;
```

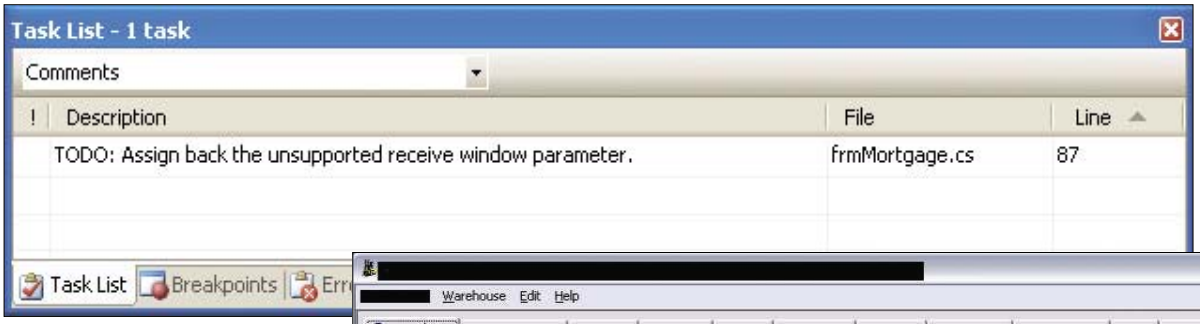

Figure 6 Manual work is also supported.

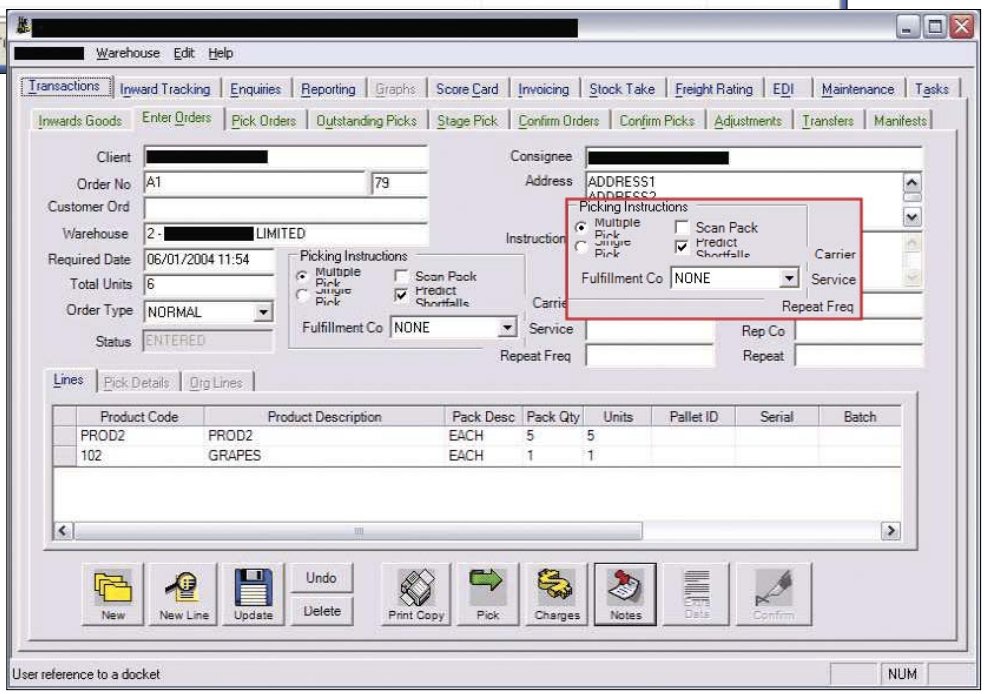

Figure 7 Small corrections sometimes cannot be avoided.

objects. The Ice Porter provides special support for converting SAL functions into an object-oriented syntax. The porting tool provides the corresponding options needed during porting.

Embedded SQL commands are converted by the Porting Project framework into native ADO.NET calls. The Framework proves very useful here since it contains a complete .NET implementation of 4GL features. This means that all of the convenient features provided by embedded SQL can still be used in future. The connection to the physical database can be configured externally. sqlTRANSLATOR is a plugin developed by fecher for the Porting Project framework allowing alternative back ends such as Oracle and Microsoft SQL Server to be used without altering the source code.

Not yet perfect

There are still a few small but significant things remaining that prevent conversion from SAL to .NET at the press of a button. It is precisely here that a programmer must manually intervene, especially in the finalization phase of the project. Even here, the porting tool does not let the team down. Comments about these critical areas are displayed in the Ice Porter log files and also in the generated source code. They are highlighted in the Visual Studio task panel, as shown in the small example in figure 6.

This example illustrates the fact that parameters may be passed by reference in SAL, but not in C#. The box displays the manual alteration needed to return the parameter for a created window.

There are other language elements for which conversion cannot be fully automated and which therefore require rework: unlike most other external libraries, the Team Developer runtime DLLs are not supported in .NET. This means that their content must also be ported. The Ice Porter generates empty external wrapper functions for this purpose, which must be filled out with code implemented during the finalisation phase.

Although the porting tool virtually replicates even the most complex of formulas, discrepancies may arise in some cases because of the different way that SAL and .NET implement control elements. It is therefore recommended that each window and dialog be opened as a test to detect and resolve any spurious discrepancies. Refer to figure 7 for an example.

SAL allows multiple inheritance with method overloading and late binding, leading to very complex constructs that can sometimes cause problems when converting to C#. The porting tool therefore attaches to-do comments to the appropriate pieces of code to prompt the developer to check the porting result.

Conclusion

The Porting Project from the Ice Tea Group provides an efficient collection of software and know-how, which is able to port even the most extensive Gupta project to Visual Studio 2005 and all within a calculable budget. The local porting partners have a business model – offering a conversion to .NET which runs and has been tested – that dispels the user's concerns about escalating cost factors and residual risks, especially for the manual rework.