



PORTABLE GUI DEVELOPMENT

Alan Bell

TESSELLA SUPPORT SERVICES PLC

Issue V1.R4.M0

June 2003



Introduction

Portability of applications across different platforms is a subject that has attracted a lot of attention for some time. Until recently, it has been quite difficult or expensive for an application with a graphical user interface (GUI) to be truly portable. Portability requires a certain amount of conformance of operating systems, programming languages and tools. This means that in most cases a GUI is limited to a certain environment, for example, to the Windows platform. In addition, a new version of any of the underlying ingredients may break the original GUI.

The situation becomes more difficult if the application must be portable across very different platforms, such as UNIX and Microsoft Windows. One approach is to make the interface layer as ‘thin’ as possible. The vast majority of the code is in the fully portable ‘engine’ and the interface code is an easily separable module, with a version for each of the supported platforms. This does require separate source code for each supported platform, with the risk that all code will not be maintained in step, leading to divergence of versions.

A more appealing and effective approach is being offered by Java. Java has gained enormous popularity as a programming language for the worldwide web but also offers similar advantages for GUI applications. In a sense, Java is a variation of the above interface layer approach, in that the Java engine already provides the interface to the underlying platform. However, the consequences are quite dramatic. The latest Java release, with its rich set of GUI tools, is being ported to virtually all platforms, and true to its “write once, run everywhere” slogan requires no special maintenance for differing platforms, which is reflected in reduced costs.

This report outlines the Java approach and some other methods and tools that can be used to create truly portable interfaces and can be maintained as a single application. Java has been ported to a large number of platforms and so everything said here should apply to all of those platforms. For the alternative approaches, only X/Motif and MS Windows are explicitly covered here, although some of the tools described can also be used to create Apple Macintosh applications.

In addition, this report also describes a couple of in-progress initiatives that aim to provide cross-platform operability and development of GUIs.

Java

Java has gained a wide and rapidly growing popularity and acceptance as the programming language of the world wide web since its first release in 1995 by Sun Microsystems. Java is an object-oriented language that is secure, robust, portable, simple, multi-threaded and distributable. Java comes with a large library of tools useful for many areas of distributed and Internet computing.

The most common appearance of Java is as an applet. Applets are small programs that are downloaded from a server and executed in a client browser, independent of the server's and client's platforms. Java achieves this portability by using a virtual machine, which runs on a particular computer. Virtual machines are available for many different platforms and their maintenance is provided by various vendors. Standard Java code is compiled into platform-independent bytecode, which can run on any Java virtual machine (JVM).

Java is, however, more than just a language for writing applets. It is a general programming language whose library contains a rich set of tools to create GUIs. The first versions of Java provided only basic GUI tools like lists, radio buttons and the like, but the latest release of the Java libraries has a much larger set of tools that now include tables and trees, various forms of buttons for toolbars, sliders, progress bars, etc. Java now rivals the capabilities of existing GUI interfaces (Visual Basic on Windows, X/Motif on Unix) but with the distinct advantage of being portable. In many cases Java is also faster than competing products as compiler technology takes advantage of the built-in safety features of Java.

A new feature of Java is that the "look-and-feel" of the GUI can be chosen by the programmer, independently of the underlying platform. Earlier versions of Java GUIs always reflected the platform; the same Java application would use the Microsoft Windows "look-and-feel" on Windows platforms while it appeared like a X/Motif GUI on Unix machines. With the most recent Java version one can choose either a Windows, X/Motif, a special Java or even a user-defined "look-and-feel" on all platforms. A Macintosh "look and feel" is in preparation. Some examples of the different "look-and-feel" appearances under Windows and X/Motif can be found below.

Java has become the first choice when a new application has to be written. Its various characteristics allow for quick, safe, portable and cost-effective code

development. Where necessary, it can be interfaced with other languages, for example, where speed is critical in certain code sections. This does, however, compromise the portability.

For existing applications, other approaches may be more effective but this must be decided on a case-by-case basis. The cost of other products, the degree of portability, the available programming resources, etc. must be weighed against each other before one can reach a definitive answer. The following sections give an overview of various other portable GUI toolkits.

The wide availability of the JVM has opened the door to portable code from other languages as well. Once a program in any language is compiled into bytecode that conforms to the JVM it can be run anywhere like any Java programme. A few languages have already taken advantage of this feature. Insofar as they have GUI capabilities, similar comments apply to them as to Java.

Running X/Motif Applications on Windows

This approach does not involve writing inherently portable code, but instead provides a method by which code written for an X-Windows/Motif platform can be run under Windows. The code is written and tested on the UNIX system using the local X-Windows libraries. The solution that allows this code to be run under Windows comes in two parts. The first is a library with which the code is linked when compiled on the PC. The second is a run time X server that allows the executable to draw to the screen using X/Motif calls. Some products, such as OpenNT, provide a complete set of UNIX functionality like `fork()` and `exec()` as well as X/Windows and Motif, under Windows NT.

This approach has several advantages if a base of Motif code already exists. Only a small number of changes, if any, need be made for the Windows version, resulting in source code which is consistent between versions. Existing Motif expertise can be used, removing the overhead required in learning Windows programming.

There are some disadvantages. The final product will have a Motif look and feel even under Windows. All Windows users of the product must have the X server application, which may increase the product cost. Not all Motif calls are included in the libraries used to produce Windows versions.

As one would expect, there are some disadvantages with this approach. The development libraries must be purchased for all supported UNIX platforms, which are expensive. Also, some UNIX users may not like having an application with a Microsoft look and feel if all their other applications are Motif.

Products:

- WIND/U
- MainWin

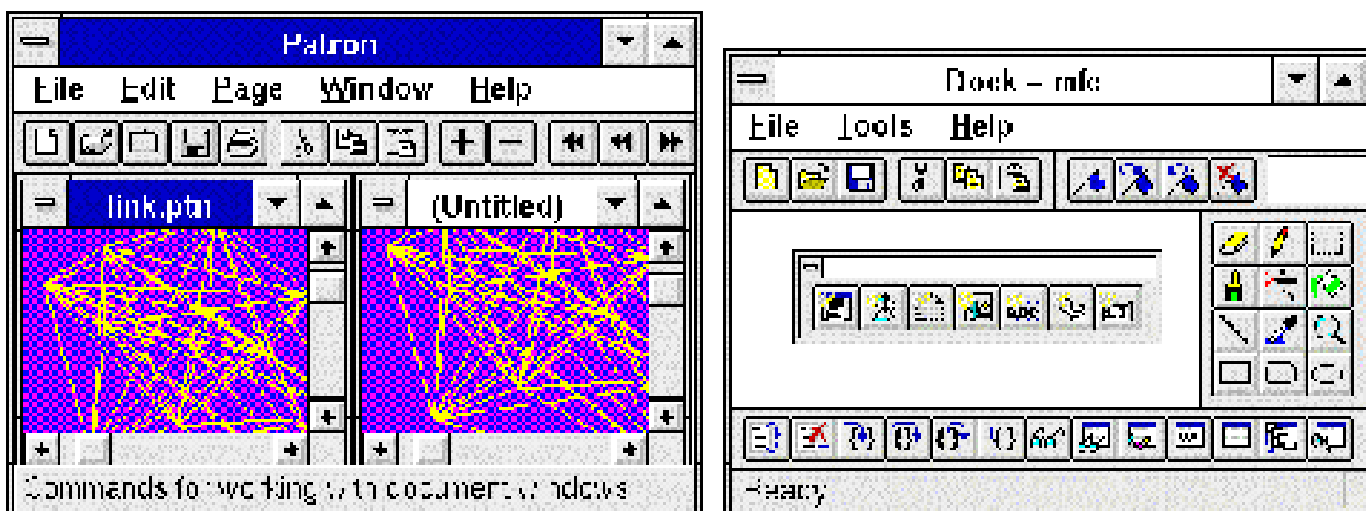


Figure 2: Windows created using MainWin

Platform Independent GUI Toolkits

If a new application is being written, rather than an existing one being ported, the most attractive method of creating a portable GUI is to use a portable library. This can then be used to create user interface components on all target platforms.

Portability toolkits tend to take either a ‘lowest common denominator’ approach, providing only those interface components that appear on all platforms, or attempt to provide all widgets on all platforms, often by supplying their own widget set.

One product following the former approach is the XVT toolkit. This is supplied as a set of libraries that must be linked with the application code, which can be C or C++. The application makes calls to the same XVT functions, whatever the destination platform. The code is then linked with the library for the appropriate platform.

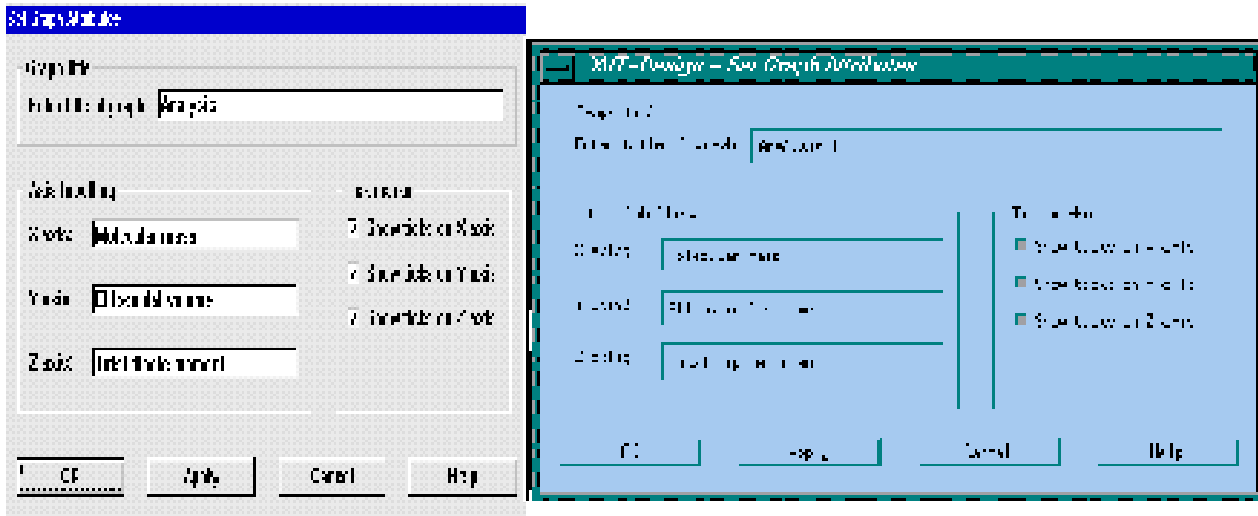


Figure 3: The same XVT dialog under Windows and Motif

Development is straight forward provided only the standard controls are used. However, if more complex functionality is required, it is often necessary to resort to platform dependent code. Many of the extensions supplied with XVT are actually written by third party suppliers and so quality varies.

Galaxy, in contrast, takes the ‘superset’ approach to providing interface widgets. Development is done in C++ using the library of classes provided with the product. A complete development environment is supplied, but this constrains developers to a prescribed development methodology. Galaxy is a very complete library that can be used to produce highly portable code. It is also very well documented.

The highly object-oriented approach required by Galaxy, along with the specialised development environment, does result in a very steep learning curve, building skills that are not necessarily transferable to other products or environments.

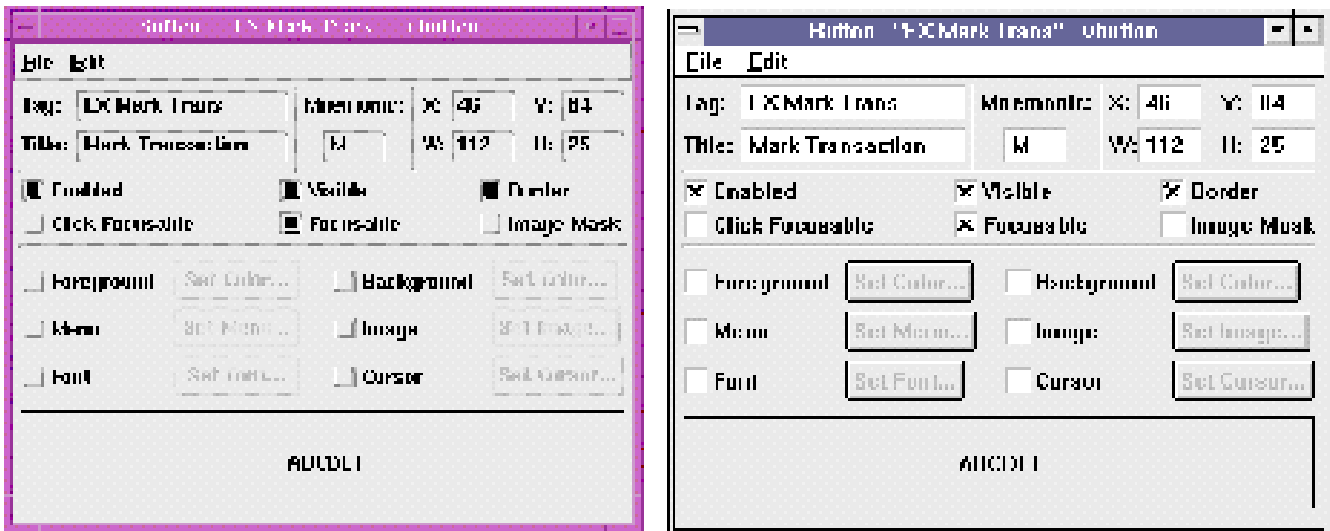


Figure 4: A dialog created in Galaxy viewed in Motif and Windows environments

Another option is the Qt library, produced by Troll Technology. The latest version (3.x) is portable between the Windows, Unix, Linux and Mac platforms. The library supplies a full set of widgets for all platforms and development is carried out in C++. The look and feel can be the standard one for the current platform or a user defined one (and can be changed at runtime). A version of this library is also available for more limited environments such as embedded platforms. The fact that the Linux KDE environment (including the KOffice application) is entirely based on Qt demonstrates that it is a good quality, robust library. Once a licence fee per Qt developer has been paid, there are no further royalty or deployment fees for its use.

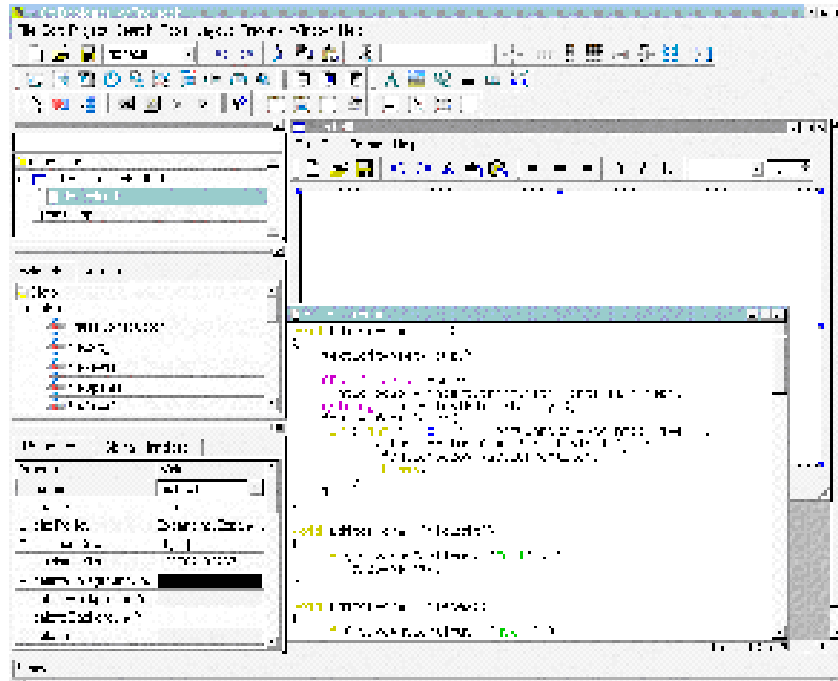


Figure 5: Qt in action – the development and design environment

Wine

Wine is an Open Source implementation of the Windows API built on top of the Unix operating system. Wine can be thought of as a Windows compatibility layer, which provides a program loader that allows many Windows applications to run unmodified on x86-based Unix systems, such as Linux and Solaris, and also includes a development toolkit, Winelib. This allows existing Windows based source code to be ported to Unix.

Wine is still under development, and it is not yet suitable for general use. Nevertheless, many people find it useful in running a growing number of Windows programs.

Code Generators and Screen Designers

These applications generally provide a graphical environment in which the user can build an interface by dragging and dropping interface components onto a design canvas. The application will then generate the appropriate source code for the target platform. This is a very efficient method for producing simple interfaces where the interface module can be maintained using the screen designer application.

If the required interface is more complex, it is often impossible to create the complete interface using the screen designer tool. If this is the case, the developer has no choice but to make changes to the generated code. These changes will generally be platform dependent. As soon as this has been done, the screen designer cannot be used to maintain the interface and separate versions of the interface code must be maintained for each target platform. These versions can quickly diverge, making maintenance more difficult.

Products:

- X-Designer
- UIM/X.

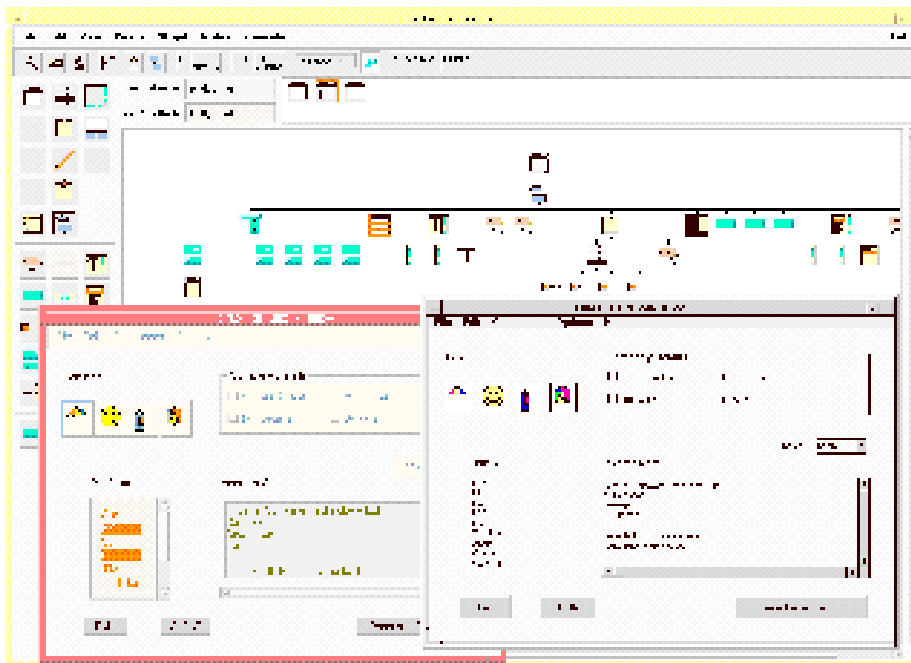


Figure 6: X-Designer screen showing Windows and Motif versions of the same dialog

Third Party Applications

For some applications it may be more appropriate to use a third party product that is available on all the target platforms, although Java is gaining considerable support in this area.

If the application is an interface to a database, then the most appropriate choice might be the interface tools supplied by the database vendor. Oracle, for example,

provides tools to create either a forms interface on multiple platforms or to create a Java applet interface.

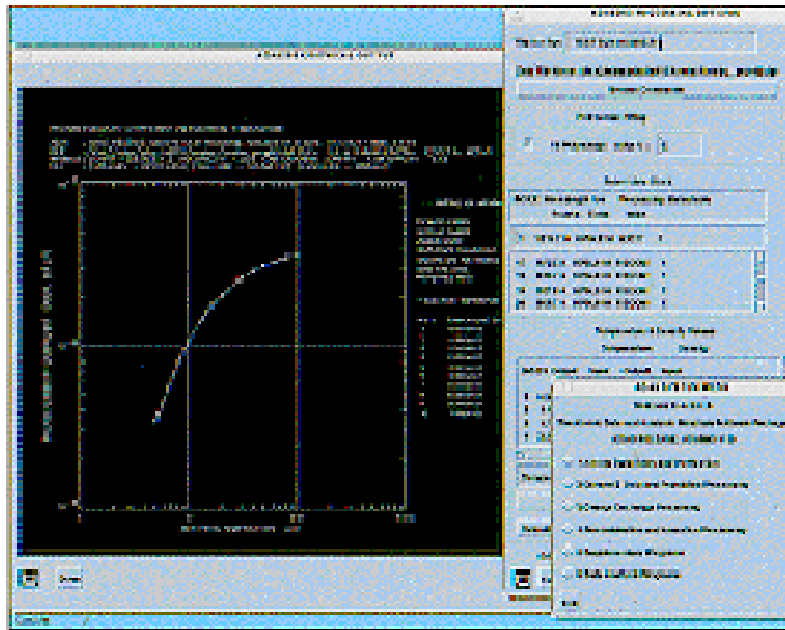


Figure 7: The GUI for this application was written using IDL

For data visualisation applications it might be more appropriate to use a specialist package such as PV-Wave, IDL or AVS to create the user interface. Interfaces built using third party tools will often have a proprietary look and feel, which, although similar on all platforms, will not be the standard look and feel on any platform. In addition they will often require a licence for the third party software for every user, or alternatively an expensive developer's licence, which allows the creation of stand-alone executables.

The Mono Project

The Mono project is an effort to create an open source implementation of the .NET Development Framework, which is platform independent.

The .NET platform contains a language specification that compilers can follow if they want to generate classes and code that can interoperate with other programming languages (The Common Language Specification: CLS). This Common Language Infrastructure platform provides language independence to programmers.

Any API that is written using a CLS provider language can be used by any language that is a CLS consumer. Compilers generate code in a format called Common Intermediate Language (CIL), which is an intermediate representation of a compiled program and is easy to compile to native code or using Just-in-Time (JIT) engines. The restrictions placed by the runtime on the CIL byte codes ensure that it is possible to do a good job at optimising the code in a JIT compiler. Mono aims to bring the Common Language Infrastructure platform to free systems, such as Unix and Linux.

Mono includes: a compiler for the C# language, a runtime for the Common Language Infrastructure (also referred as the CLR) and a set of class libraries. Mono has implementations of both ADO.NET and ASP.NET as part of its distribution.

Currently, Mono is still under development and as yet does not fully provide User Interface support (the Windows.Forms packages are still under development), but in time this initiative may provide a viable option for portable GUI development.

Summary

There are a wide variety of options available for the development of portable GUI applications. Each different situation will require a different solution. The exact choice will depend on the user's requirements, the platforms involved and the type of application being developed.

Tessella Support Services plc
Creating Software for Science and Engineering

Tessella's services range from feasibility studies, through system design, development, implementation and ongoing support. Our expertise includes:

Data Analysis Software
Data Capture
Simulation Software
Advanced Graphics
Systems Support
Database Applications

Other Technical Supplements available include:

- | | |
|---|--|
| <input type="checkbox"/> Archiving of Electronic Info | <input type="checkbox"/> Object Oriented Programming |
| <input type="checkbox"/> Active Server Pages | <input type="checkbox"/> Pocket PC |
| <input type="checkbox"/> Automated GUI Testing | <input type="checkbox"/> Portable GUI Development |
| <input type="checkbox"/> Bayesian Statistics | <input type="checkbox"/> Printer Technology Guide |
| <input type="checkbox"/> Beowulf Clusters | <input type="checkbox"/> Real Time Systems |
| <input type="checkbox"/> C++ | <input type="checkbox"/> Regression Testing |
| <input type="checkbox"/> Client-Server Technology | <input type="checkbox"/> Security and the Internet |
| <input type="checkbox"/> COM | <input type="checkbox"/> Simulation |
| <input type="checkbox"/> Computational Fluid Dynamics | <input type="checkbox"/> Soft Computing |
| <input type="checkbox"/> Computer Image Processing | <input type="checkbox"/> Software Design Methodologies |
| <input type="checkbox"/> Decision Support Systems | <input type="checkbox"/> Software Development Cycle |
| <input type="checkbox"/> Electronic Data Capture | <input type="checkbox"/> Software Documentation |
| <input type="checkbox"/> Electronic Lab Notebooks | <input type="checkbox"/> Software Portability |
| <input type="checkbox"/> Excel | <input type="checkbox"/> Software Re-engineering |
| <input type="checkbox"/> Extending the Life of Software | <input type="checkbox"/> Software Specification |
| <input type="checkbox"/> Federal Drug Administration | <input type="checkbox"/> SQL |
| <input type="checkbox"/> FORTRAN 90 | <input type="checkbox"/> UNIX Inter-Process Comms |
| <input type="checkbox"/> Grid Computing | <input type="checkbox"/> UNIX Systems Performance |
| <input type="checkbox"/> High Throughput Screening | <input type="checkbox"/> UNIX Workstations |
| <input type="checkbox"/> Instrumentation | <input type="checkbox"/> Visual Basic 6 |
| <input type="checkbox"/> Integrated Lab Systems | <input type="checkbox"/> WAP |
| <input type="checkbox"/> J2EE | <input type="checkbox"/> Web Services |
| <input type="checkbox"/> Java | <input type="checkbox"/> Windows 2000 Services |
| <input type="checkbox"/> Lims | <input type="checkbox"/> XML |
| <input type="checkbox"/> Linux | <input type="checkbox"/> X Windows |
| <input type="checkbox"/> Microsoft Net | |



INVESTOR IN PEOPLE

Tessella Support Services plc

3 Vineyard Chambers, Abingdon, Oxon, OX14 3PX, England

Tel: (+44) (0) 1235 555511 Fax: (+44) (0) 1235 553301

E-mail: info@tessella.com Web Address: <http://www.tessella.com>