Michael Tschater
<tschater/at/web.de>

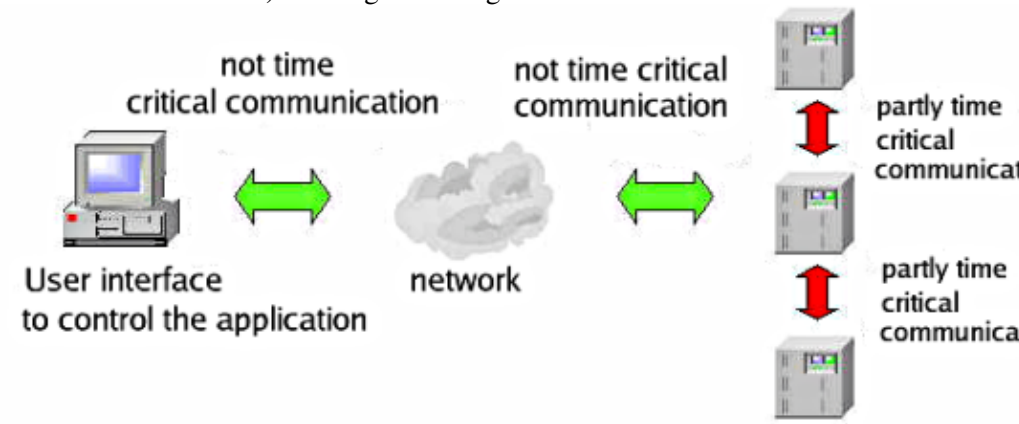
L'autore:

Michael e' principalmente impegnato con lo sviluppo di software applicato all'hardware (firmware). Per quanto riguarda il suo progetto corrente deve essere ancora presa una decisione circa la strategia relativa all'ambiente di sviluppo – da utilizzarsi per la programmazione dell'interfaccia del suo firmware– .

Sviluppo di software indipendente dalla piattaforma

Premessa:

Quasi tutto l'equipaggiamento usato nell'industria potrebbe essere controllato tramite u
rete. L'interfaccia utente gira sull'hardware e funziona come un semplice client, riceven
ed inviando dati per i quali il tempo non sia un fattore (es. parametri di inizializzazione
risultati di misurazioni). Nel seguente diagramma viene evidenziato in verde:



I progetti di software richiedono spesso una risposta alla domanda su quali sistemi operativi debbano essere supportati. Mentre i lettori di questa rivista propendono per Linux, vengono richiesti anche altri sistemi operativi (quasi sempre Windows). In via d
principio, il sistema operativo da adottare non costituisce una priorita' dominante per l'applicazione; l'utente deve essere capace di ottenere risultati intuitivamente.
Il seguente articolo dimostrera' che non e' richiesta una decisione su una specifica piattaforma da adottare visto che e' possibile scrivere software che puo' essere compilat
per vari sistemi operativi. Questo articolo si limitera' ai PC con Linux e Windows. Dovrebbe essere possibile usare le applicazioni anche sui Mac e MacOSX, ma questo n
si puo' dimostrare visto che manca l'hardware

Con librerie indipendenti dalla piattaforma differenziamo due approcci per produrre dei controlli per il dialogo con l'utente:

1. Librerie native: Per la visualizzazione degli elementi vengono utilizzate le corrispondenti routine dei sistemi operativi. Questo assicura che tutti i controlli appaiano come le applicazioni standard del sistema operativo. Una libreria nativa presenta controlli diversi sotto Linux piuttosto che sotto Windows 2000 o XP.
2. La seconda possibilita' e' programmare un aspetto appropriato (look & feel), volendo con questo dire che tutti i controlli devono essere visualizzati dalla libreria e devono apparire identici su tutti i sistemi operativi.

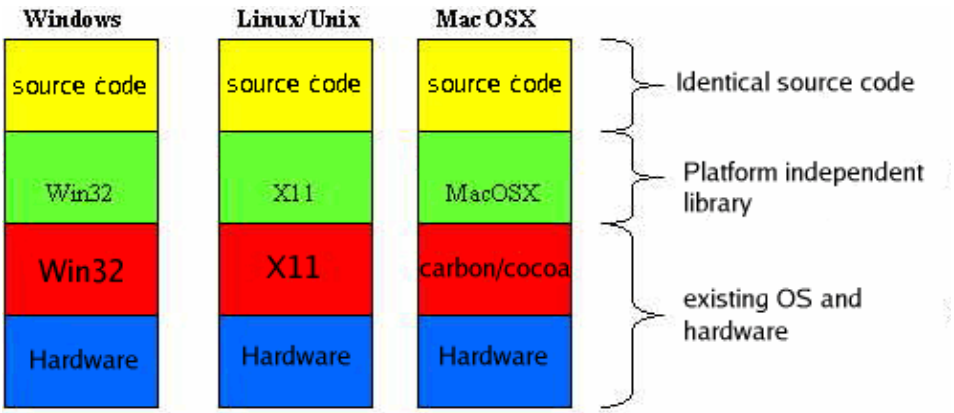
A parte le caratteristiche tecniche delle librerie, fattori operativi addizionali giocano un ruolo che deve essere anch'esso comparato:

- Ambiente di sviluppo: un ambiente di sviluppo integrato (es. costruttori di intefaccia grafiche – GUI Builder – e generatori di makefile) semplificano lo sviluppo del software.
- Documentazione e supporto: immediato aiuto in caso di problemi.
- Costi: Mentre molte librerie sono liberamente disponibili per uso privato, talvolta ci sono dei costi in caso di scrittura di applicazioni commerciali. Per prendere decisioni fondamentali sul progetto del software tali costi devono essere giustificati a chi deve decidere.
- Costi effettivi per il porting (l'adeguamento del codice per i vari sistemi operativi) tra i sistemi

In un caso reale un altro aspetto deve essere preso in considerazione; questo comunque non si applica a tutti i progetti:

- Il software prodotto dovra' utilizzare controlli nativi da integrare perfettamente nella esistente architettura del sistema. L'utente non dovrebbe essere in grado di notare differenze tra il sotware esistente nel sistema ed il nuovo.

Visualizzando le librerie in un modello a strati si sviluppa il seguente grafico:



Linguaggi di programmazione

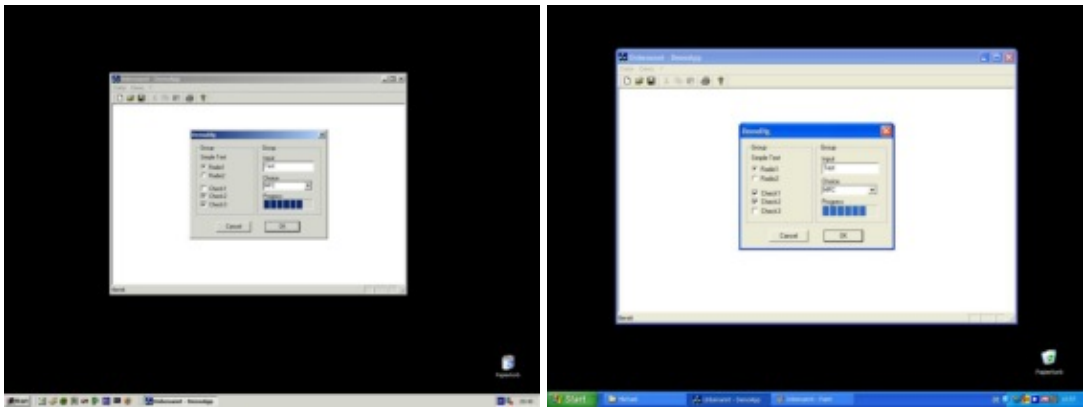
Il primo criterio da decidere e' il linguaggio di programmazione. Ci sono diverse scelte, che discuteremo di seguito:

1. Librerie C/C++
2. Java
3. Kylix
4. Smalltalk
5. Mozilla

Le alternative tra C e C++ saranno spiegate piu' dettagliatamente visto che sono le meno provate tra gli sviluppatori.

Un'applicazione di esempio

Per essere in grado di confrontare i vari pacchetti di software verra' generata una applicazione di esempio, usando tutte le librerie. L'implementazione dell'applicazione non possiede alcuna funzionalita' ma mostra tutti i controlli piu' importanti. Per il lato windows verra' creato puro software Windows (Visual C++ 6.0, MFC Class–Library), gli altri pacchetti verranno comparati in base all'aspetto (look & feel). Come distro linux usero' una RedHat Fedora Core 2 e Debian 3.0.



Videate da Windows 2000 e Windows XP (codice sorgente per Visual C++ qui (win32_src.zip)).

Librerie C/C++

Trolltech Qt

Qt è una libreria di classi della ditta norvegese Trolltech per una programmazione indipendente dalla piattaforma in C++. Il gestore di finestre di Linux KDE si basa sul pacchetto QT. In origine Qt era sotto un tipo di licenza inaccettabile da molti utenti Linux. Per questa ragione fu sviluppata la libreria GTK+, la quale costituisce la base per il gestore di finestre Gnome. Nel frattempo, la versione QT Linux, cosi' come quella

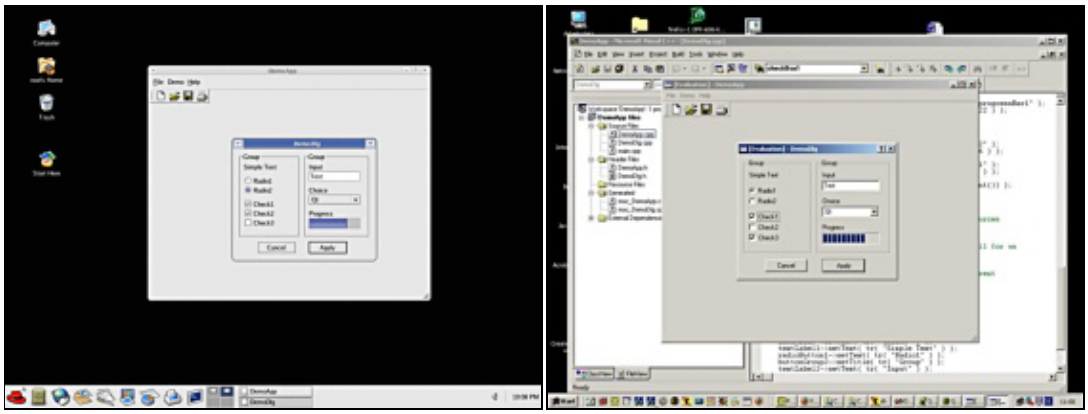
MacOS e' stata resa disponibile sotto la licenza GPL, incluso tutto il codice sorgente. Qt per Windows, invece, e' ancora sotto licenza ad uso commerciale. Un versione di prova limitata nel tempo puo' essere scaricata dal loro sito – sara' differenziata dall'uso, che potra' essere commerciale o didattico. Qui viene spiegata la versione commerciale di prova. Questa versione richiede una registrazione.

Oltre alla versione per Windows, Linux(Unix) e Mac e' disponibile una versione "embedded", che gira su varianti di sistemi "embedded" Linux ed fornisce un'amministrazione piu' leggera delle finestre.

Come previsto l'installazione sotto Linux si svolge semza problemi. Incluso vi e' il generatore di interfaccia grafica utente (GUI) Qt Designer. C'e' una dettagliata documentazione sui progetti di esempio, una guida di inizio rapido ed una visione d'insieme delle classi. Qt Designer genera come output una descrizione in formato XML della GUI. Usando lo strumento Qt qmake si puo' generare un Makefile funzionante ricavato dalla descrizione in XML. Questo Makefile poi genera del codice sorgente C++ dalla descrizione della GUI (Qt–Tool: uic) e chiama il compilatore Meta Object Compiler (Qt–Tool: moc). Quest'ultimo traduce le estensioni specifiche del linguaggio Qt in codice sorgente C++. Dopodiche' puo' essere usata una procedura standard di make per compilare l'eseguibile.

La seguente sequenza e' necessaria per generare i file sorgente manualmente (Il file di input e' MyDialog.ui):

- uic MyDialog.ui > MyDialog.h
- uic –impl MyDialog.h MyDialog.ui > MyDialog.cpp
- moc –o moc_MyDialog.cpp MyDialog.h



Videate di Linux e Windows 2000 qui (qt_src.tar.gz) il codice sorgente per QtDesigner.

Qt: Riepilogo

Nome:	Trolltech Qt
Versione:	3.3.2
Sistemi operativi:	Linux, Win32, MacOS, Solaris, IRIX, AIX, HP–UX
Linguaggio di programmazione:	C++
Licenza:	GPL o Licenza proprietaria (commerciale)
Vantaggi:	

	• Libreria base per KDE Windows Manager in Linux • pacchetti di installazione su tutte le distribuzioni standard (installazione molto semplice) • Controlli generici sotto Windows • potente ambiente(i) di sviluppo • affidabile • il supporto per la migrazione delle applicazioni MFC per Win32 consente una conversione incrementale del codice sorgente MFC.
Svantaggi:	• possibili costi di licenza (esosi) • La versione del software di prova genera errori durante l'installazione sotto Windows
Ambiente di sviluppo:	es. QtDesigner, KDevelop
WWW:	http://www.trolltech.com
Documentazione:	manuali, tutorials, mailing lists es. http://doc.trolltech.com/3.3/index.html
Progetti di riferimento:	• KDE Desktop (Default es. con SuSE) • Opera Browser • Photoshop Album
Distribuzioni:	molto diffuso

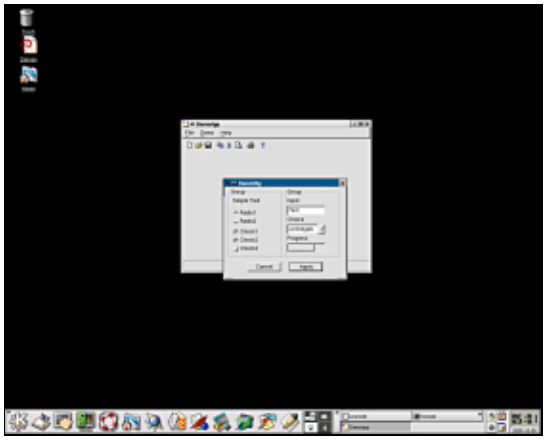
wxWidgets

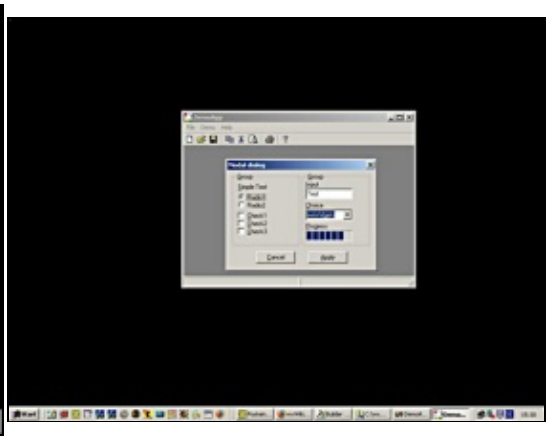
Il toolkit wxWidgets e' disponibile da 12 anni, ma solo pochi mesi fa il pacchetto ha assunto il suo nome attuale. Il nome wxWindows, usato fino ad allora, e' stato abbandonato dopo "contatti" con Microsoft. wxWidgets include un'enorme raccolta di classi per tutte le aree di applicazione. L'elenco delle applicazioni di riferimento dimostra la maturita' del pacchetto software.

La programmazione viene svolta in C++, simile a Visual C++ sotto Windows.

Uno svantaggio e' che riceverete degli errori con wxWindows2.4.2 sotto RedHat Fedora Core 2 quando compilerete i programmi di esempio. La causa e' che le chiamate a GTK+ sono dichiarate private nella versione modificata da RedHat di Gtk+. La chiamata a queste funzioni e' quindi non consentita. Comunque si tratta di problemi minori. Tutto funziona senza problemi quando viene usata la libreria standard GTK+. Sotto Debian tutto funziona bene da subito.

L'installazione sotto Windows e' stata fatta senza problemi.





Videate Linux e Windows 2000 qui (wx_src.zip) il codice sorgente .

wxWidgets: riepilogo

Nome:	wxWidgets
Versione:	2.4.2
Sistema operativo:	Linux, Win32, dispositivi embedded
Linguaggio di programmazione:	C++
Licenza:	LGPL
Vantaggi:	• semplice gestione (molti esempi). • documentazione molto buona.
Svantaggi:	• Problemi con la combinazione: Fedora Core 2 – wxWindows2.4.2
Ambiente di sviluppo:	
WWW:	http://www.wxwidgets.org
Documentazione:	manuali, tutorials, mailing lists, wiki es. http://wiki.wxwidgets.org
Progetti di riferimento:	AOL Communicator
Distribuzione:	non molto diffusa

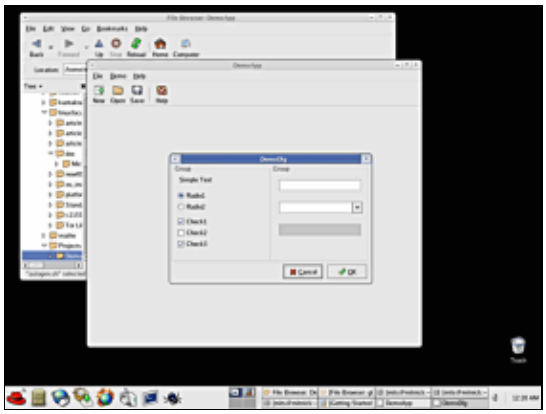
GTK+ (con gtkmm)

L'acronimo sta per "The GIMP Toolkit". I due ben noti progetti sono Gnome Windows Manger, parte di qualsiasi distribuzione standard di Linux e l'applicazione grafica GIMP. Gnome e' il secondo maggior ambiente desktop, KDE a parte (vedi QT) sotto Linux. E' l'ambiente desktop di default di molte distribuzione.

Con l'introduzione della versione 2 di GTK+ l'aspetto (look & feel) e' stato sostanziosamente arricchito.

Una particolarita' di GTK+ e' la sua completa implementazione in C. Di conseguenza il costruttore di interfaccia utente (GUI) *glade2* produce codice C. Usando *gtkmm* (precedentemente noto come GTK−−) si puo' programmare anche in C++.

Al contrario dell'aspetto professionale di GTK+ per Linux, "GTK+ per Win32" non e' altrettanto impressionante. Cliccando sul link della pagina principale di GTK+ otteniamo immediatamente un avvertimento "**The program(s) might crash unexpectedly or behave otherwise strangely ".** (I(l) programmi potrebbero terminare improvvisamente o comportarsi in modo strano) − ma naturalmente, cosi' fanno molti programmi commerciali in Windows−. La stabilita' sembra basarsi molto sulla macchina, sui driver dei dispositivi di visualizzazione, su altro software installato o meno (status al 6−9−2004). Il coraggioso sviluppatore clicca sulla pagina di download comunque ad affronta una lunga lista di singoli componenti da scaricare. La ricerca di un pacchetto onnicomprensivo sarebbe inutile. Invece si puo' leggere come installare i vari componenti del software, per poi tornare alla pagina di download qualora mancasse qualche componente. Nella pagina Web di "GTK+ per Windows" si legge : "You are expected to be quite experienced to be able to use GTK+ in your own programs. This isn't Visual Basic." (Dovete essere piuttosto esperti per potere usare GTK+ nei vostri programmi. Questo non e' Visual Basic). Dopo l'installazione dei componenti iniziali ed un mancato tentativo di far partire una delle applicazioni di esempio, la maggior parte degli sviluppatori potrebbe perdere la voglia di approfondire. La presentazione non professionale dei componenti "GTK+ per Windows" scoraggia l'uso del pacchetto software in una qualsiasi applicazione professionale.



Videata di GTK+ per Linux qui (gtk_src.tar.gz) il codice sorgente per glade2)

Sguardo d'insieme di GTK+

Nome:	GTK+ − The GIMP Toolkit
Sistemi operativi:	Linux, Win32
Linguaggi di programmazione:	C (C++ con gtkmm)
Licenza	LGPL
Vantaggi:	• libreria base per Gnome Windows Manager sotto Linux

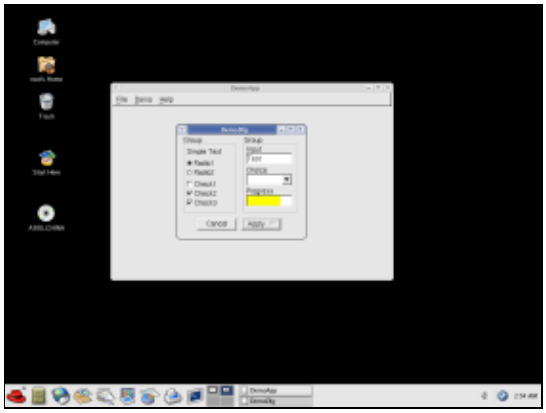
	• pacchetto d'installazione incluso in tutte le distribuzioni standard (installazione molto semplice) • controlli generici sotto Windows • affidabile (sotto Linux)
Svantaggi:	• Implementazione in Win32 pesante, non e' stabile (status 09−2004)
Ambiente di sviluppo:	es. *2glade* (GUI Builder), Anjuta
WWW:	http://www.gtk.org
Documentazione:	manuali, tutorials, mailing lists e.g. http://developer.gnome.org/doc/API/2.0/gtk/index.html
Progetti di riferimento:	• Gnome Desktop • GIMP • Gnumeric
Distribuzione:	Linux: molto diffusa, Windows: marginale

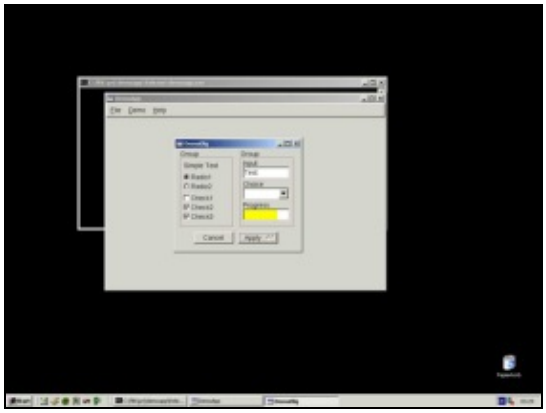
FLTK

FLTK Toolkit (Fast, Light Tool Kit) e' un pacchetto piuttosto sconosciuto, e' stato implementato come successore di XForms. I sorgenti completi sono offerti per il download dal sito web del programma. La dimensione di 2.3MB (Linux) e 3MB (Windows) tiene fede al suo nome (fast = leggero). L'installazione sotto Linux fila liscia, scompattate e lanciate '*make*' ed e' fatta . Quindi l'utente ha a disposizione librerie, applicazioni di esempio, il costruttore di interfaccia grafica (GUI) "*fluid*" ed un manuale di programmazione. Ovviamente il numero di classi disponibili e' piu' piccolo rispetto ai pesi massimi come Qt e wxWindows. Le classi incluse comprendono il mondo GUI, il che significa: finestre, menu, controlli, OpenGL e visualizzazione di immagini. Le classi per la comunicazione in rete e simili non sono incluse .

L'installazione sotto Windows e' stata molto piu' complicata. Quando si usa l'ambiente di sviluppo Visual C++ occorre trascodificare solo il progetto principale. Queso comunque causa problemi con le librerie grafiche. Una semplice soluzione e' di decommentarle nel file di configurazione config.h. Una specifica caratteristica sotto Windows e' che la versione DEBUG della libreria FLTK apre sempre una finestra DOS in piu'. Questo fa si' che i programmi che sono lanciati dalla riga comandi possono scrivere verso stderr e stdout

Nel suo insieme il FLTK Toolkit lascia l'inpressione di essere stato ben congegnato. La documentazione enfatizza la limitatezza dell'eseguibile (80kb per un "ciao mondo") ed offre una veloce grafica 2D e 3D (OpenGL). In piu' occorre menzionare anche una buona portabilita' .





Videate di Linux e Windows 2000 (codice sorgente) qui (fltk_src.tar.gz))

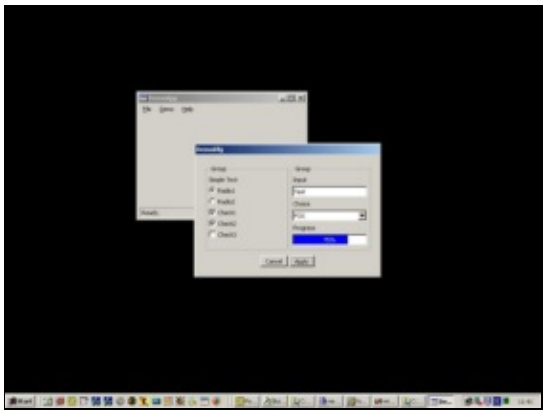
FLTK; Riepilogo

Nome:	Fast Light Tool Kit
Versione	1.1.5rc2
Sistemi operativi:	Linux, Win32, MacOS
Linguaggio di programmazione:	C++
Licenza:	LGPL
Vantaggi	• una libreria molto snella • Il codice sorgente inclusa documentazione ed ambiente di sviluppo "fluid". • buon supporto per OpenGL (che non e' stato testato) • controlli generici sotto Windows
Svantaggi	• installazione sotto Win32 (Visual C++) non senza problemi • L' ambiente di sviluppo "fluid" non gira in modo stabile sotto Windows.
Ambiente di sviluppo:	es. fluid (GUI Builder)
WWW:	http://www.fltk.org, Download: http://freshmeat.net/projects/fltk/
Documentazione:	Manuali, Tutorials, Mailing Lists es. http://
Progetti di riferimento:	• http://vtkfltk.sourceforge.net/
Distribuzione:	limitata, addirittura sconosciuta anche per la maggior parte degli sviluppatori

FOX Toolkit

Il Fox Toolkit si vanta di essere il toolkit piu' veloce disponibilie. Offre un vasto numero di elementi GUI ed una interfaccia OpenGL.

L'installazione viene completata senza problemi sia sotto Windows che sotto Linux. Sono disponibili una dettagliata documentazione e progetti di esempio. Uno sguardo d'insieme delle classi non e' incluso nella versione qui presentata ma e' disponibile online.



Videata di Windows 2000 (codice sorgente qui (fox_src.zip))

FOX: rieplogo

Nome:	FOX Toolkit
Versione	1.2.9
Sistemi operativi:	Linux, Win32
Linguaggio di programmaziones:	C++
Licenza:	LGPL
Vantaggi	• buona documentazione
Svantaggi	
Ambiente di sviluppo:	
WWW:	http://www.fox−toolkit.org
Documentazione:	Manuali, Tutorials, Mailinglist
Progetti di riferimento:	• X File Explorer (Xfe)
Distribuzione:	scarsa distribuzione

Altre possibilita'

In aggiunta alle librerie sopramenzionate vorrei anche citare i seguenti progetti che comunque non esaminero' ulteriormente:

- GNUstep [http://www.gnustep.org/]: Limitata usabilita' sotto windows
- Visual Component Framework [http://vcf.sourceforge.net/]: Non c'e una versione completa per Linux disponibile

JAVA

Nel 1995 la Sun introdusse una nuovo linguaggio di programmazione. Oltre che per gli abituali desktop−PC Java fu pianificato per prodotti industriali (macchine da caffe', tostapane etc.). La penetrazione nel mercatto avvenne inizialmente attraverso le applicazioni internet (applet) in connessione con i browser web. Nel frattempo Java viene anche usato per applicazioni a se' stanti, per le quali e' ben indicato per la varieta' delle sue caratteristiche.

Sotto elencheremo e spiegheremo in breve le caratteristiche piu' importanti di Java.

Indipendente dalla piattaforma (Platform−independent)

Java e' indipendente dalla piattaforma su cui gira. Le applicazioni Java consistono in byte−code, che puo' essere interpretato da una macchina virtuale. Quindi le applicazioni sono in grado di girare su qualsiasi hardware per il quale sia presente una macchina virtuale. L'interpretazione tramite la macchina virtuale implica pero' una velocita' di elaborazione piu' bassa, se confrontata con quella dei software compilati. Per controbilanciare questo svantaggio, sono stati sviluppati dei miglioramenti tipo la compilazione just−in−time (JIT) che traduce le istruzioni del programma della macchina virtuale in istruzioni per la macchina fisica. Il risultato in questo caso e' un programma caricato in memoria che puo' essere eseguito rapidamente senza essere interpretato. Analisi addizionali del comportamento in fase di esecuzione con la tecnologia Hotsport forniscono ulteriori miglioramenti.

Orientato agli oggetti

Java e' orientato agli oggetti. Gli sviluppatori del linguaggio orientato agli oggetti furono ispirati da Smalltalk. Ci sono ancora tipi di dati primitivi che, presumibilmente per ragioni di efficienza, non sono amministrati come oggetti.

Sintassi del linguaggio

La sintassi del linguaggio e' simile quella di C e C++, anche se le inconsistenze che in quei linguaggi portano a dei bug non sono state adottate. Un principio degli sviluppatori del linguaggio era quello di combinare i migliori concetti dei due linguaggi di programmazione .

Alcuni esempi:

- nessun pre−processore. Il pre−processor ed i file header non servono piu' visto che tutte le informazioni sono lette direttamente dai file delle classi. .
- puntatori: Java non li usa, al loro posto sono usate delle references. Una reference rappresenta un oggetto .
- garbage−collector: per prevenire problemi derivanti dalla creazione ed eliminazione degli oggetti, l'amministrazione dell'oggetto viene gestita dall'ambiente Java in fase di esecuzione. Uscendo dall'array attivo, gli oggetti sono automaticamente cancellati. Con questa tecnica gli oggetti o gli array di memoria che non sono stati abilitati, cosi' come i falsi distruttori sono evitati.
- eccezioni: al contrario della gestione delle eccezioni in C++, in Java le eccezioni sono usate molto piu' frequentemente, spesso sono obbligatorie

Liberie di classi

Java include una vasta libreria di classi: JFC (Java Foundation Class) per la generazione di superfici. *Swing*

Sicurezza

Il codice Java viene inizialmente controllato da un verificatore per quanto riguarda la correttezza strutturale e la sicurezza dei tipi. Un gestore di sicurezza verifica gli accessi alle periferiche. Qualsiasi problema di sicurezza viene riportato come eccezione in fase di runtime.

Idoneita' per i progetti

I vantaggi summenzionati hanno l'effetto collaterale di rendere Java non adatto a tutti i progetti. Queste proprieta' non sono state implementate a ragion veduta e non a causa di errori o manchevolezze, esse appartengono alla filosofia del linguaggio.
Tra le altre ci sono ad esempio:

- accessi alle periferiche specifici della piattaforma
- accessi diretti all'hardware
- ingerenze nel sistema operativo

Java Development Kit (JDK)

Il Java Development Kit puo' essere scaricato dal sito internet di Sun. Include un carnet base di applicazioni, classi java e documentazione in linea. Le applicazioni sono un compilatore, un debugger, un visualizzatore di applet ed una varieta' di programmi ausiliari necessari per la generazione e la verifica della applicazioni Java e delle applet. Questo kit offre solo l'essenziale, il compilatore deve essere lanciato da riga comandi. In piu' il pacchetto contiene il Java Runtime Environment (JRE, che include la macchina virtuale), richiesto per eseguire il byte code. La documentazione descrive interamente le API

JHelloWorld

Con l'aiuto del JDK la classica applicazione "hello world" verra' implementata.
Passo 1: Generazione del codice sorgente.

```
sh>vi Helloworld.java
```

```
public class HelloWorld {
 public static void main (String[] args) {

    System.out.println("Hello World!");

  }

}
```

Il nome del file ed il nome della classe devono corrispondere.
Passo 2: Trascodifica

```
sh>javac Helloworld.java
```

Passo 3: Lancio dell'applicazione usando la macchina virtuale.

```
sh>java Helloworld
```

JavaScript e Java

JavaScript e Java sono stati spesso accostati in quanto si pensa siano simili. Sbagliato, JavaScript fu sviluppato originariamente da Netscape come linguaggio di script da incorporare nell'HTML. Non e' un linguaggio di programmazione autonomo, esso dipende dal browser. Il nome JavaScript deve essere visto piu' come un trucco di marketing.

Tentativi di standardizzazione

Fino a ora tutti i tentativi di standardizzare Java sono falliti. La ragione potrebbe essere la riluttanza di Sun a rinunciare al controllo esclusivo sul successivo sviluppo degli standard Java.

Disassemblaggio

Un problema potrebbe essere che le applicazioni possono essere disassemblato. A dispetto della sicurezza attualmente e' possibile riconvertire il Bytecode in codice sorgente. Questo e' possibile perche' il Bytecode e' scritto per un processore virtuale e contiene, al contrario di un tradizionale assemblatore, importanti informazioni addizionali. Esse rendono molto piu' agevole il disassemblaggio del codice. Quindi non si puo' nascondere una API proprietarie o speciali conoscenze nel codice.

Linguaggio–miracolo o aspettativa presto delusa

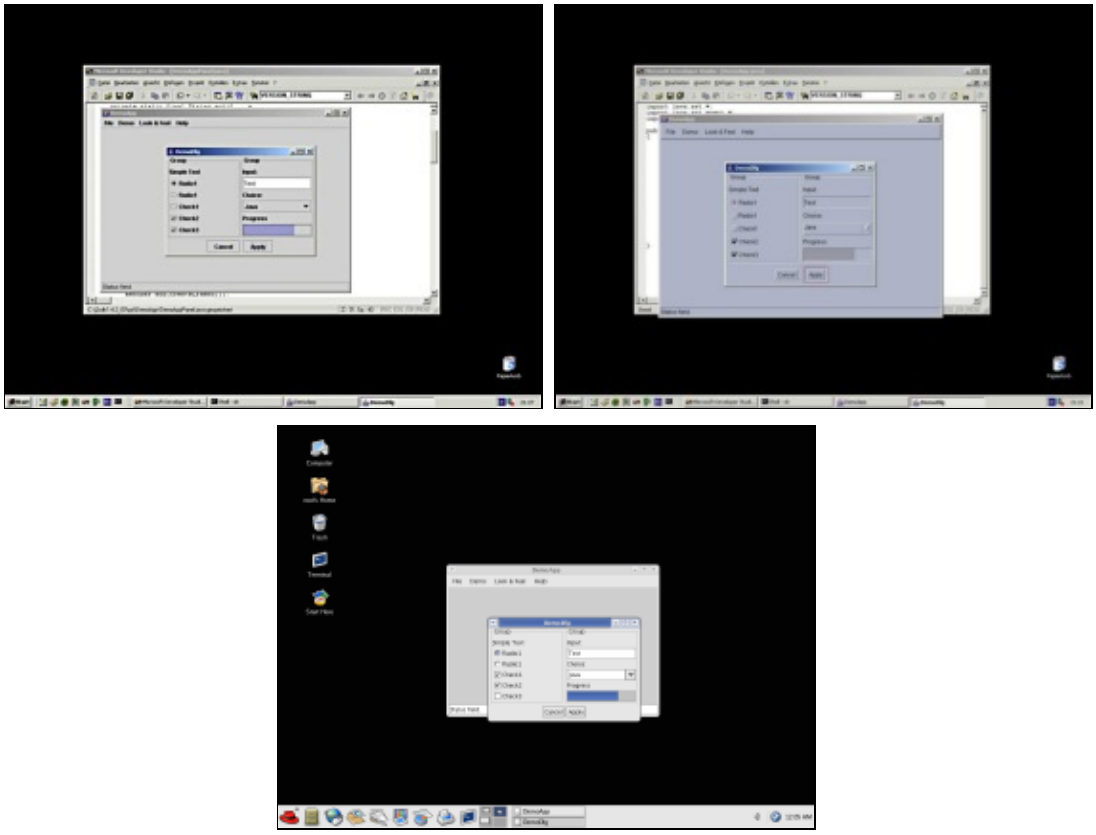
Il concetto dietro Java fu visto all'inizio come la risposta definitiva allo sviluppo di software indipendente dalla piattaforma. Tuttavia le aspettative iniziali sono state disattese. Ci sono versioni che sono in conflitto con le diverse macchine Java e la velocita' di esecuzione rappresenta un problema. Molte compagnie, dopo i primi tentativi tornarono alla programmazione standard C++. L'accresciuto numero di download di wxWidgets e' una prova di questo.

A questo proposito un sito interessante e':
http://www.internalmemos.com/memos/memodetails.php?memo_id=1321 dove impiegati della Sun forniscono argomenti contro Java.

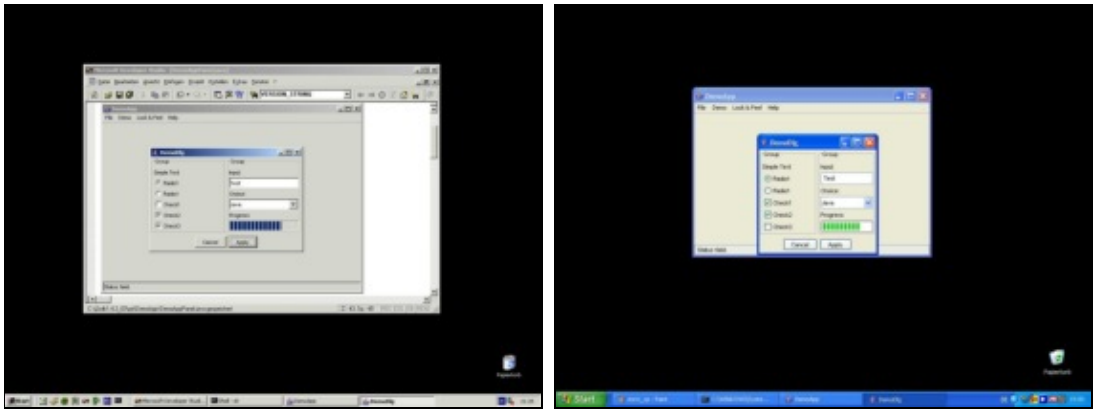
GUIi (Interfacce grafiche utente) con Java

Java offre per default 2 possibilita' per programmare interfacce grafiche :

1. Java ha una ricca libreria di classi (JFC, Swing). Qui non sono usate funzioni del sistema operativo. Tutti i widget sono disegnati con istruzioni Java. Questo fa si che si possa cambiare l'aspetto in fase di esecuzione. Come potete vedere dalle videate qui sotto .
2. Le funzioni base di AWT. AWT non ha elementi complessi come le strutture ad albero, quindi non e' adatto per la maggior parte delle applicazioni. .

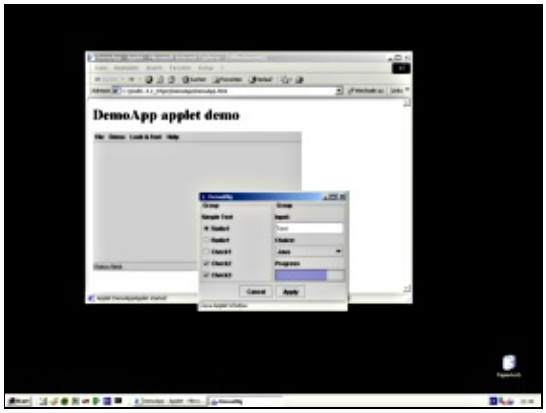


Videata di Java con l'aspetto Metal–, Motif– e GTK+ (Quellcode qui il codice (java_src.zip))



Videata di Java con l'aspetto di Windows sotto Windows 200 ed XP (identico codice sorgente)

Visto che tutti i browser piu' comuni supportano Java, tutte le applicazioni possono essere scritte in modo che possano girare come applet all'interno di un browser web. Questa caratteristica puo' essere sfruttata ad esempio con le tecnologie embedded dove il Bytecode java viene scaricato da un server web che e' integrato nell'applicazione.
Le seguenti videate mostrano la stessa applicazione sotto forma di applet Java integrata in una pagina web



Videata di Java con l'applicazione di esempio come Applet (code here (java_applet.zip))

SWT ed Eclipse

Sebbene Java offra elementi GUI simili ad altri toolkit, gli sviluppatori si lamentano di essi. I problemi maggiori sono l'insufficiente velocita' di esecuzione e la mancanza di funzionalita'. IBM ha sviluppato un'alternativa allo Standard Widget Toolkit (SWT) che consente l'uso di elementi GUI nativi sotto Java. Un progetto di riferimento e' l'ambiente integrato di sviluppo Eclipse che offre strumenti di sviluppo indipendenti dalla piattaforma. Sia il toolkit che l'ambiente di sviluppo sono software libero.

Abbreviazioni usate nel contesto con JAVA

JDK (Java Development Kit)	

	Il pacchetto Java completo per generare applicazioni Java, comprende l'applicazione, le classi Java e la documentazione
JRE (Java Runtime Environment)	comprende la macchina virtuale, obbligatorio per l'uso di applicazioni Java.
J2ME (Java 2 Micro Edition)	Java per dispositivi con risorse limitate.
J2SE (Java 2 Standard Edition)	Java per il desktop (Linux, Windows, ...)
J2EE (Java 2 Enterprise Edition)	Java per la generazione di applicazioni client/server multistrato cosi' come di Java servlets e pagine Java−server .
JFC (Java Foundation Class)	Classi per lo svilppo di interfacce grafiche utente (−>Swing)

Java: riepilogo

Nome:	JAVA 2 PLATFORM STANDARD EDITION DEVELOPMENT KIT 5.0
Versione	5.0
Sistemi operativi:	• Linux, Windows, Solaris (SUN) • Linux, Windows, AIX, Solaris (probabile MacOS, OS/2, FreeBSD, Amiga, BeOS) (Jikes −> IBM)
Linguaggio di programmazione:	JAVA
Licenza:	licenza proprietaria (SUN)
Vantaggi	• linguaggio robusto (molte fonti di errore sono eliminate dalla concezione stessa del linguaggio). •
Svantaggi	• inguaggio proprietario, controllato esclusivamente da Sun • Macchina virtuale, deve corrispondere esattamente alla piattforma su cui gira • Lento in esecuzione • La programmazione di SWT e' piu' complessa di Swing
Ambiente di sviluppo:	es. Eclipse
WWW:	http://java.sun.com
Documentazione:	manuali, tutorials in generale http://java.sun.com/j2se/1.5.0/docs/, http://www−e.uni−magdeburg.de/mayer/java.html SWT: http://eclipse−wiki.info/SWT, http://www.java−tutor.com/java/swtlinks.html
Progetti di riferimento:	
Distribuziione:	molto vasta

Kylix

Kxlix e' una piattaforma di sviluppo per Linux e Windows. Con l'aiuto della libreria CLX di Borland (Component Library for Cross−platform) le applicazioni possono essere sviluppate sotto Delphi e C++, e possono girare sotto entrambe le piattaforme. Secondo quando riportato nella homepage di wikipedia (Link de.wikipedia.org/wiki/Kylix) questa libreria e' solo un contenitore per la summenzionata libreria Qt. Inoltre l'ambiente di sviluppo integrato (IDE) Kylix e' chiaramente un'applicazione non nativa Linux basata su *wine*. (Link de.wikipedia.org/wiki/WINE_Is_Not_an_Emulator) i cui eseguibili devono essere linkati a *libwine*. Considerando tutto cio', Kylix potrebbe non avere molto senso per sviluppatori C++ visto che l'uso di Qt in un IDE libero e' molto piu' diretto.

Kylix: Riepilogo

Nome:	Kylix
Versione	3
Sistemi operativi:	Windows, Linux
Linguaggio di programmazione:	Delphi, C++
Licenza:	Software proprietario
Vantaggi	• sviluppo sotto Delphi e C++
Svantaggi	• Costi di licenza
Ambiente di sviluppo:	Kylix
WWW:	http://www.borland.de/kylix
Documentazione:	
Progetti di riferimento:	
Distribuzione:	non molto diffuso

Smalltalk

Smalltalk e' un classico tra i linguaggi di programmazione. Fu sviluppato nel 1969/70 da Xerox e fino ad oggi rappresenta un buon esempio di linguaggio orientato agli oggetti. Tutto e' un oggetto in Smalltalk. Non ci sono tipi di dati semplici, Smalltalk lavora come Java e .Net (vedi sotto) in una macchina virtuale. La sintassi tenta di avvicinarsi all'inglese parlato, ma e' totalmente diversa da qualsiasi altro linguaggio di programmazione. Smalltalk naque gia' in un ambiente grafico. Smalltalk era circa 10−15 anni avanti al suo tempo. Smalltalk aveva un buon successo fino all'arrivo di Java..

Eccco il programma 'Hello world !' sotto smalltalk:

```
Transcript show: 'Hello world !'; cr.
```

Smalltalk e' ancora usato oggi. La variante piu' diffusa e' Smalltalk−80 (standardizzato nel 1980). Un ambiente di sviluppo potente e' ad esempio Squeak.

Smalltalk: riepilogo

Nome:	Smalltalk (e.g. Squeak)
Versione	3.6
Sistemi operativi:	Windows, Linux, Solaris, MacOSX, Darwin
Linguaggio di programmazione:	Smalltalk
Licenza:	Open Source
Vantaggi:	Totalmente orientato agli oggetti
Svantaggi	Smalltalk e' messo da parte da Java ed ha un base utenti significativamente minore.
Ambiente di sviluppo:	es. Squeak
WWW:	http://www.smalltalk.org
Documentazione:	
Progetti di riferimento:	
Distribuzione:	non molto diffuso

Mozilla

Mozilla? Un web browser? Come si puo' programmare con un web browser? Mozilla non e' solamente un browser, ma anche una piattaforma indipendente che include diversi standard tipo XUL (XML based interface language − linguaggio XML basato sull'interfaccia−). XUL viene usato per definire la struttura ed il contenuto di un'applicazione. Tutti i file usati sono in testo semplice. Mozilla non fa differenza tra programmi e pagine web..

Se digitate la seguente stringa nel campo dell'indirizzo in mozilla, verra' mostrato il broswer stesso

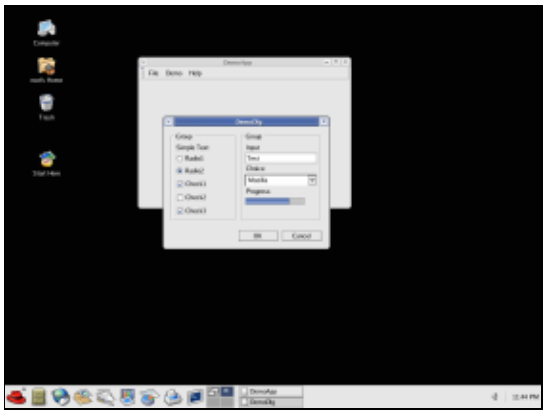
```
chrome://navigator/content
```

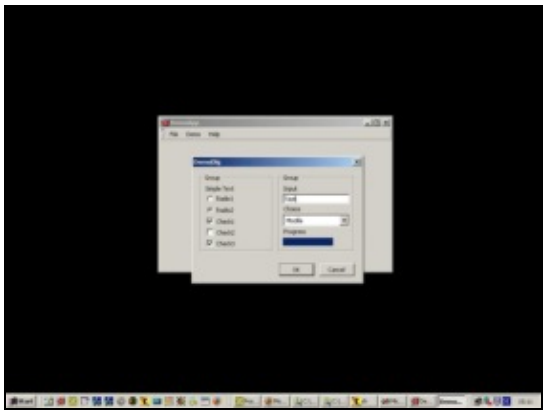
Il codice seguente visualizza un bottone in Mozilla che aprira' una finestra con il testo "Hello World" quando ci si clicca sopra :

```
<?xml version="1.0"?>
<!-- Beispiel XUL Datei -->
<window
xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">
<box align="center">
  <button label="Push" onclick="alert('Hello World');" />
</box>
```

```
</window>
```

Sviluppare software con mozilla e' molto diverso dallo sviluppo di software classico. Mozilla ha diverse innovazioni tipo la separazione dell'applicazione dalla sua presentazione. Questo rende possibile cambiare l'aspetto dell'applicazione ("Temi"). Progetti di successo come il browser web firefox dimostrano che siamo di fronte ad un framework robusto .





Videate in Linux e Windows 2000 qui (moz_src.tar.gz))il codice sorgente.

Mozilla: riepilogo

Nome:	Mozilla
Versione	1.6
Sistemi operativi:	Windows, Linux,
Linguaggio di programmazione:	XUL
Licenza:	Mozilla Public License, Netscape Public License
Vantaggi	• concetti innovativi • supporto per la maggior parte degli standard web (JavaScript, Stylesheets,...) • le applicazioni girano nel browser o a se stanti
Svantaggi	
Ambiente di sviluppo:	
WWW:	http://www.mozilla.org
Documentazione:	Manuals, tutorials, mailing lists. Es. www.xulplanet.com
Progetti di riferimento:	Mozilla firefox
Distrubuzione:	largamente diffuso ma di rado usato per progetti software .

La risposta di Microsoft

Nel frattempo Microsoft ha naturalmente riconosciuto i segni del tempo ed ha proposto il proprio approccio. E' stata sviluppota una piattaforma chiamata .NET, che, ultimo ma non meno importante, dovrebbe ridurre la migrazione degli sviluppatori di software verso la concorrente piattaforma Java. Uno sguardo piu' approfondito rivela in effetti molti parallelismi tra i contendenti, anche se sotto forma di nomi diversi. L'equivalente del 'bytecode' di Java e' chiamato C# 'Intermediate Language' MSIL).

Cosa e' .NET ?

.NET e' una tecnoologia proprietaria di Microsoft, che sara' la base per tutti i futuri prodeotti Microsoft. Il supporto per quella che fino ad ora era la libreria preferita "MFC per Visual C++" e' stato abbandonato con l'indroduzione di .NET. .NET dovrebbe semplificare lo sviluppo di applicazioni di rete ed internet; molti concetti di Java sono stati adottati. .NET supporta la programmazione orientata agli oggetti e viene fornito con una singola libreria di classi la quale potrebbe essere utilizzata da vari linguaggi di programmaziones (C#, VB.NET). Questo vuole dire che il " linguaggio intermedio" – che accede all'hardware – viene generato dal codice del programma (cosi' come il Java Sourcecode – –> Java Bytecode –> macchina virtuale –> hardware fisico)..

Versioni future di Windows saranno distribuite con il framework .NET .

Cosa e' Visual Studio .NET ?

Visual Studio .NET e' un ambiente di programmazione che semplifica lo sviluppo di software .NET, ma non e' obbligatorio averlo.

Differenze tra Visual Basic (VB) e VB .NET

Anche se VB.NET – per ragioni di compatibilita' – supporta molte funzioni originali di VB e la sintassi del linguaggio e' stata mantenuta, e' un linguaggio di programmazione completamente nuovo.

Quale linguaggio di programmazione e' piu' indicato?

Visto che il codice sorgente di VB.NET e quello di C# vengono trascodificati in MSIL, il linguaggio di programmazione non fa alcuna differenza. Non ci sono, ad esempio, differenze di velocita' tra codice VB.NET e C#. Il compilatore C# dovrebbe essere piu' adatto visto che e' stato sviluppato specificatamente per il framework .NET

.NET e Linux

Alla faccia dell'approccio indipendente dalla piattaforma, Microsoft probabilmente non sviluppera' una variante per Linux di .NET, e questa e' la ragione per la quale un gruppo di sviluppatori – vicino a Miguel de Icaza (Ximian. Evolution) – e' stato ingaggiato con questo compito. Il pacchetto open–source *Mono*, versione 1.0, e' nel frattempo disponibile .

.NET: riepilogo

Nome:	Microsoft .NET–Framework
Versione	
Sistemi operativi:	Windows, Linux
Linguaggio di programmazione:	C#, Windows: VB.NET
Licenza:	proprietaria
Vantaggi	• parte dei futuri Windows
Svantaggi	• software proprietario • nessuna versione di .NET disponibile per Linuxe • API completamente nuove
Ambiente di sviluppo:	Visual Studio .NET
WWW:	
Documentazione:	
Progetti di riferimento:	
Distribuzione:	Limitata allo stato attuale

Conclusioni

Prima della valutazione finale, ricordiamo il compito che dobbiamo svolgere: quello dello sviluppo di un front–end, che dovra' comunicare tramite rete con l'hardware connesso. Per questo il codice sorgente dovra' essere in grado di essere trascodificato nelle piattaforme Linux e Win32. L'applicazione non dovra' essere diversa dal software esistente nel sistema. Con questo compito da svolgere la vista dei pacchetti testati apparira' distorta e non puo' essere considerata come un valido giudizio a livello generale..

Il miglior esempio di questo concetto e' il toolkit FLTK. Con esso abbiamo un sistema molto capace e di dimensioni limitate. I punti di forza sono un codice sorgente ridotto, una buona interfaccia grafica ed una buona portabilita'. Queste caratteristiche rendono il toolkit adatto per progetti di applicazioni grafiche ed embedded. Per lo sviluppo dell'interfaccia il numero di classi disponibili, la gestione e l'aspetto delle

applicazioni generate costituiscono un problema. Quindi FLTK e' meno indicato per questo tipo di applicazioni.

Una grossa delusione per lo sviluppatore di software puo' essere rappresentata dal progetto GTK+ sotto Windows. La comunita' Linux potrebbe dimostrare molto piu' coinvolgimento. Gli avvertimenti piazzati nel sito web non sono certi messi per aumentare la confidenza. La cosa e' ancor piu' disdicevole visto che il pacchetto GTK+ in se' sembra piuttosto valido. Il potenziale e' molto ampio; l'implementazione verso la piattaforma Windows e' ancora in divenire.

L'utilizzo degli outsider Smalltalk e Mozilla costituisce la mia preferenza personale. Una ditta, che genera i propri profitti con lo sviluppo interno dell'hardware, dovrebbe tenere in scarsa considerazione gli aspetti filosofici. Sebbene Smalltalk sia un linguaggio meglio orientato alla programmazione ad oggetti e la programmazione XUL di Mozilla accresca di importanza il browser incluso, questi pacchetti non sono prodotti "tipo" per lo sviluppo di software.

In questa recensione Kylix, cosi' come GTK+ PER Win32 hanno lasciato un'impressione negativa. Rimane ben poco della gloria del prodotto originale: Turbo Pascal. Negli anni 80 Borland forniva un potente IDE per questo progetto, che girava sia sugli home computers che sui primi PC. Era conosciuto per il suo prezzo ragionevole ed il codice veloce. Nel frattempo molto e' cambiato. Borland e' diventata Inprise, quindi e' ritornata Borland. Turbo Pascal e' cambiato in Object Pascal, poi in Delphi ed in ultimo in Kylix (naturalmente con modifiche ed espansioni). Il suo uso non ha senso attualmente – almeno per nuovi progetti.

In questo ambiente Microsoft dimostra che ha individuato quello che il nostro tempo richiede. Inizialmente, la compagnia ha tentato di accrescere gli standard Java con Visual++. A parte i comandi standard di Java veniva consentito anche l'accesso alle API Win32 ed anche al registro di Windows (il che era l'esatto contrario della filosofia del linguaggio). Inoltre gli eseguibili Win32 erano generati automaticamente. Dopo alcune battaglie legali con Sun, si dovette esporre un'avvertenza relativa al fatto che l'applicazione poteva non girare su altri sistemi operativi. Alla fine della fiera Microsoft ha interrotto i contatti con Java. E' stata sviluppata una strategia completamente nuova. Con .NET e C# e' stato generato un nuovo standard. La combinazione di Windows, .NET e C# costituisce sicuramente un buon pacchetto, ma si era detto lo stesso anche con la combinazione di Windows, Visual++ e la libreria di classi MFC. Lo svantaggio e' che si e' completamente alla merce' di un fornitore che vuole imporre i "propri" standard (Windows). E' molto probabile che per l'immediato futuro Microsoft non voglia pianificare alcuna implementazione di .NET verso altri sistemi operativi. La conversione free Mono deve ancora dimostrare di avere una reale capacita', A scapito dei primi successi, ad oggi nessuna conclusione puo' essere tratta

Senza remore sono consiglabili i pacchetti Qt, wxWindows e Java. La scelta finale e' difficile visto che tutti e tre i prodotti sono in grado di generare complicate interfacce software. Si possono qui sviluppare diverse opinioni, a seconda dell'adeguatezza del supposto, dei costi, della prontezza, della filosofia di programmazione, etc. Le distinzioni si possono cercare nei dettagli; la filosofia Java, in effetti, non consente l'accesso diretto all'hardware, ma puo' risultare vantaggiosa in altri aspetti. Da un punto di vista tecnico i 3 contendenti possono gestire il compito richiesto senza problemi

Questa e' una conclusione soggettiva dell'autore: il tifoso dell'Open Source potrebbe propendere verso wxWindows per il lavoro da svolgere. A parte una concezione gradevole ed un buon supporto e' disponibile una documentazione sufficiente.

Translation information:
de −−> −− : Michael Tschater <tschater/at/web.de>
de −−> en: Jürgen Pohl <sept.sapins/at/verizon.net>