Gorm und ProjectCenter, die GNUstep RAD Werkzeuge

by Georges Tarbouriech
<georges.t(at)linuxfocus.org>

About the author:

Georges ist ein langjähriger Unixbenutzer. Er liebt GNUstep und die Werkzeuge, die dieses großartige Rahmenwerk zur Verfügung stellt.

Abstract:

RAD steht für Rapid Application Development (schnelle Applikationsentwicklung). Am Ende der 80er, als NeXTstep herausgebracht wurde, kam es mit einem unglaublichen Wekzeug namens InterfaceBuilder. Zusammen benutzt mit einem anderen Werkzeug namens ProjectBuilder, erlaubte es, grafische Applikationen blitzartig zu bauen. GNUstep bietet eine freie Version von diesen Werkzeugen, genannt Gorm.app und ProjectCenter.app.

Zu Beginn...

Von den Anfängen der Computer an, war Softwareentwicklung eine große Herausforderung. Computer waren recht gewaltig an Größe trotz ihrer nur sehr geringen Power. Sie waren recht teuer, nicht wirklich zahlreich und die Entwickler waren nicht in der Lage, sie so oft zu benutzen, wie sie es wünschten, da sie sie mit anderen Leuten teilen mußten. Dann versuchten Forscher einen Weg zu finden, um mehr als eine Aufgabe auf einmal ausführen zu können, um die Effizienz zu erhöhen. Offensichtlich mußten sie Programmiesprachen von der Pieke auf designen und entwickeln, und die armseligen Ressourcen der verfügbaren Maschinen mit in Betracht ziehen.
Auf diese Weise erschienen während der 60er verschiedene neue Programmiersprachen: LISP, FORTRAN, BASIC, Algol68, BCPL, etc. Als nächstes kam die B Sprache, die bald die Sprache C wurde. Diese letzte veränderte die Programmierwelt.
Die objektorientierten Sprachen (SmallTalk, Objective C, C++, etc) erschienen später mit der "grafischen Ära".
In den 80ern hatten einige Maschinen grafische Betriebssysteme (Apple Macintosh, Commodore Amiga, Atari ST, etc.) und das X Window System war auf dem Weg. Zur selben Zeit arbeitete ein Unternehmen an einem GUI für IBM OS2, namens Presentation Manager. Vor Beendigung der Arbeit veröffentlichte diese Firma ihr "eigenes" GUI für ihr DOS, namens... Windos. Die ersten zwei Versionen waren kaum brauchbar, aber... die dritte startete alles. Das MvAI (Microsoft very Artificial Intelligence) war geboren !
Dadurch wurde jeder Benutzer ein Computerwissenschaftler. Seither haben wir "großartige" Anwendungen gesehen, geschrieben unter Benutzung von Excel oder Word und Visual Basic:−(

Macht nichts !
Zum Glück, lange bevor wir die obige Situation erreicht haben, wurde NeXTstep geboren und mit ihm erschien Interface Buider. Dieses Werkzeug erlaubte dir, ein GUI für deine Applikation in sehr kurzer Zeit und mit sehr großer Leichtigkeit zu erzeugen. Von dort hat sich diese Art von Werkzeugen ausgebreitet. Unter anderen laßt uns Omnis, 4D, Delphi, Kylix, etc. erwähnen. Ein paar wenige von ihnen sind Multiplattformen, während die große Mehrheit Windos gewidmet ist. Laßt uns auch erwähnen, daß es freie Toolkits gibt, die solch eine Philosophie benutzen, Gtk (Gimp Tool Kit) zum Beispiel. Proprietäre Unixe bieten ebenfalls diese Art von Werkzeugen.
Das wichtigste Feature dieser Werkzeuge ist, daß du nicht den Code für die 200 Fenster deiner Applikation schreiben mußt, sondern nur den einen, um die Daten zu verwalten.
Ob du diese Art von Werkzeugen magst oder nicht, ist nicht der Punkt. Die Entwicklungszeit ist kurz: dies ist eine Tatsache (daher der Name, "Rapid Application Development").
GNUstep versorgt uns mit freien RAD Werkzeugen. Sie werden Gorm und ProjectCenter genannt. Natürlich sind diese Werkzeuge sehr "jung", aber sie laufen. Laßt sie uns anschauen.

GNUstep

Um beide, Gorm und ProjectCenter, benutzen zu können, mußt du GNUstep installieren. Wie man dies macht, würde in diesem Artikel zu weit führen. Du findest alles, was du brauchst, auf der GNUstep Webseite. Dies beinhaltet den Quellcode, HOWTOs, Tutorien, etc.
Du kannst auch die folgenden Artikel anschauen: GNUstep, das Open Source OpenStep und GNUMail.app, der Portabilitätsbeweis.
Die Tests für den gegenwärtigen Artikel wurden unter FreeBSD 4.7 mit Window Maker 0.80.1 gemacht, unter Benutzung von gnustep−make−1.5.0, gnustep−base−1.5.0, gnustep−gui−0.8.2 und gnustep−back−0.8.2. Diese letzten sind die neuesten GNUstep instabilen Versionen. Du kannst auch die stabilen Versionen benutzen, wenn du möchtest. Außderdem benutzen wir noch den gcc 3.0.4 Compiler.

Gorm.app

Gorm steht für Graphic Object Relationship Modeler (oder vielleicht GNUstep Object Relationship Modeler, wie in der README Datei gesagt wird). Es ist ein Clon des oben erwähnten NeXTstep Interface Builder (oder des heutigen MacOS X).
Gorm ist das Werk von Richard Frith−Macdonald, der diese Projekt anfing. Gregory Casamento kümmert sich um die aktuelle Version (0.1.9) zusammen mit Pierre−Yves Rivaille. Neuere CVS Snapshots sind verfügbar unter http://savannah.gnu.org/projects/gnustep.
Du kannst die neueste stabile Version von der Gnustep Webseite herunterladen.
Die Philosophie hinter Gorm (und Interface Builder) ist, den Benutzer mit Objekten zu versorgen, die in Paletten gefunden werden und durch Schieben dieser Objekte zu leeren Fenstern, die grafischen Komponenten deiner Applikation zu entwerfen.
Die Objekte können Knöpfe (buttons), Felder (fields), checkboxes, Panel (panels), etc. sein. D.h., alles, was du zu einem Fenster hinzufügen kannst, um es benutzerfreundlich zu machen. Als nächstes kannst du sie durch Benutzen von Inspectoren verändern. Von den inspectors kannst du die Attribute verändern, Verbindungen, Größe, Hilfe definieren und Klassen für die gewählten Objekte manipulieren. Nach dem Erzeugen einer Klasse kannst du Outlets und Aktionen zu den Objekten hinzufügen.
Als nächstes instantiierst du die Klasse, wodurch ein neues Objekt (die instance) im Gorm Hauptfenster erzeugt wird und du kannst die Outlets und die Aktionen mit den dazugehörigen Komponenten verbinden. Du machtst diese genaue Steuerung durch Ziehen von der Instanz zu dem gewählten Objekt, um die Outlets zu

verbinden und von den Objekten zu der Instanz, um die Aktionen zu verbinden. Als letztes erzeugst du ein Skelett der Klassenquelldateien und du bist fertig. Mehr dazu später.

ProjectCenter.app

ProjectCenter ist, wie der Name sagt, das "Herz" des Projekts. Es ist ein Clone von Project Builder, gefunden unter NeXTstep und Mac OS X.
ProjectCenter ist ein Werk von Philippe C.D.Robert und die aktuelle Version ist 0.3.0. Wie Gorm kann man es von der GNUstep Webseite herunterladen, wenn man zu der Developper apps Section geht. Natürlich kannst du den neuesten CVS Snapshot bekommen: wir benutzen ihn für diesen Artikel und es ist Version 0.3.1.
Von ProjectCenter aus kannst du ein Projekt erzeugen, sein Interface (durch Benutzen von Gorm), seinen Quellcode schreiben; du kannst dieses Projekt bauen und laufen lassen (debugging ist noch nicht verfügbar). Kurz, du kannst alle erforderlichen Ressourcen für das Projekt verwalten: source code, Dokumentation, libraries, Unterprojekte, Schnittstellen, etc.
Wenn du ein neues Projekt erzeugst, kannst du einen Typ wählen. Du kannst wählen zwischen application, bundle, tool, library und Gorm application.
Unter anderem versorgt ProjectCenter dich mit einem Editor, mit dem du in der Lage bist, den Gorm Skelettcode zu vervollständigen.
Wie arbeiten Gorm und ProjectCenter zusammen ? Sehr gut, danke !
Ernsthaft, wir benutzen zwei Beispiele, um es zu illustrieren.

Ein paar Anmerkungen

Dieser Artikel ist KEIN Tutorium. Die Idee ist, die Leichtigkeit der Benutzung dieser Werkzeuge zu zeigen, während wir auf der Tatsache bestehen, daß du in der Lage sein wirst, denselben Code für beide, GNUstep (d.h. viele Unixplattformen... und, wenn du "kämpfen" magst, auch Windos) und MacOS X. Das einzige, was du tun mußt, ist das Interface auf jeder Plattform zu entwickeln, da die nib (InterfaceBuilder oder Gorm) Dateien nicht portierbar sind (zumindest jetzt).
Der oben erwähnte GNUMail.app Artikel zeigte die Portabilität aus der Sicht des Benutzers. Dieser hier konzentriert sich auf die Sicht des Entwicklers, noch mit der Portabilität im Gedächtnis. D.h. in GNUMail.app benutzten wir das Werk von Ludovic und Freunden und hier erzeugen wir eine GUI Applikation für beide, GNUstep und MacOS X.
Viele Tutorien sind verfügbar, entweder für MacOS X oder GNUstep. Du kannst die meisten GNUstep Tutorien von der GNUstep Webseite aus finden oder von http://www.gnustep.net, aber laßt uns ein paar von ihnen erwähnen.
− An application using Gorm and ProjectCenter von Pierre−Yves Rivaille.
− The Nicola Pero's tutorial page
− Ein älteres Tutorim, wie man einen HTMLEditor entwickelt:
http://stepwise.com/Articles/Technical/HTMLEditor/
Um mehr darüber zu lernen, kannst du auch den Quellcode, die nib Dateien, etc. von den existierenden GNUstep Applikationen (Gorm, ProjectCenter, GNUMail, GWorkspace, etc.) durchschauen und natürlich die Gnustep−Beispiele anschauen.

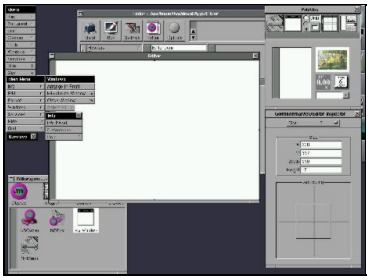
Nenn es einfach "VerySimpleEditor" (Sehr einfachen Editor)

Unter den zahlreichen MacOS X Tutorien für InterfaceBuilder, die im Internet verfügbar sind, benutzen wir das folgende als ein erstes Modell: http://www.macdevcenter.com/pub/a/mac/2001/05/18/cocoa.html. Der Autor, Mike Beam schrieb eine Menge weiterer hochentwickelter Tutorien verfügbar unter http://www.macdevcenter.com/pub/ct/37.
Warum dieses ? Weil es dich mit einem arbeitenden Texteditor versorgt, ohne eine einzige Zeile Code zu schreiben. Dies zeigt die Macht dieser Entwicklungswerkzeuge, ob sie nun unter MacOS X oder unter GNUstep laufen.
Unter Benutzung von ProjectCenter.app und Gorm.app unter GNUstep erzeugen wir einen sehr einfachen Texteditor, der in der Lage ist, auszuschneiden, zu kopieren, zu pasten. Natürlich bist du nicht in der Lage, deine Arbeit zu speichern: erinnere dich, wir werden keine einzige Zeile Code schreiben. Unter Benutzung von ProjectBuilder und InterfaceBuilder unter MacOS X machen wir dasselbe. Natürlich gibt es eine Menge an diesem Editor zu verbessern und wir lassen dies als eine Übung für den Leser. Nochmal, dieser Artikel ist kein Tutorium !
Es geht los.

Unter GNUstep

Öffne ProjectCenter.app und erzeuge ein neues Projekt namens Editor. Wähle ein Gorm Application project unten im Fenster, bevor du seinen Namen speicherst. Dies liefert dir ein Interface Item in der linken Spalte von ProjectCenter.
Klicken auf *Interfaces* zeigt *Editor.gorm* an. Doppelklick auf *Editor.gorm* und Gorm.app öffnet sich. Selektiere das default Fenster (MyWindow) und benutze das Werkzeug inspector, um den Namen in *Editor* in den Attributes zu ändern.
Aus der Palette ziehe eine TextView zu dem Editorfenster. Die TextView ist das größte Objekt, das in der gewählten Palette durch Benutzen des am weitesten rechts liegenden Icons oben im Palettenfenster gefunden wird. Verändere die Größe des Objekts, so daß es das gesamte Fenster füllt und du bist fertig.
Wähle nochmal durch Benutzen des GormInternalViewEditor inspector (während die TextView selektiert ist), wähle *Size* und ändere die Werte, damit sie zu den Editorfenstergrößenwerten passen. Diese letzten werden auf dieselbe Weise erhalten, d.h. durch Selektieren des Fensters und Überprüfen der Größe im GormNSWindow inspector. Wenn du die X und Y Werte z.B. nicht änderst, wirst du nicht in der Lage sein, die volle Breite des Editors zu nutzen, ob du die Größe des Fensters veränderst oder nicht.
Speichere alles im Gorm Document menu und beende es, um zurück zum ProjectCenter zu gehen. Selektiere das Build icon und klicke in das neue build Icon in der zweiten horizontalen Hälfte des Fensters. Alles sollte gut gehen, wenn du die richtigen Präferenzen für deinen Kompilierer, Debugger etc. gewählt hast. Z.B. mußt du, wenn du FreeBSD benutzt, make in gmake umändern (einschließlich des Pfades) durch Klicken in das Settings Icon von ProjectCenter. Überprüfe auch die Pfade des Preferences menu in ProjectCenter.
Wenn das Bauen (built) erfolgreich war (es sollte es !), mache dasselbe mit *Run* und du wirst die Editorapplikation sehen. Spiele einfach damit herum, schreibe, schneide aus, paste etc. Natürlich kannst du es später erneut starten durch Benutzen des openapp Befehls.
Wie lange hat es gedauert? Nun, ich würde sagen, ein paar Minuten.

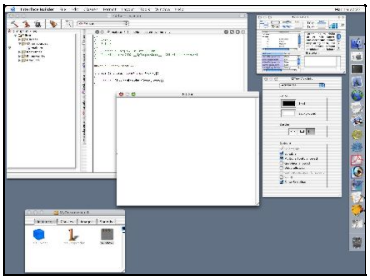
Hier ist, wie es während der Entwicklungsphase aussehen sollte:



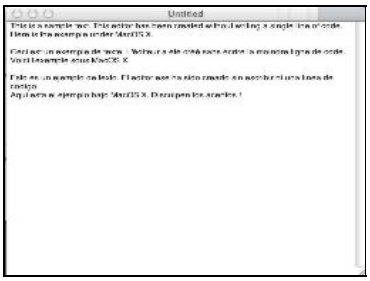
und die sich ergebende Applikation:

Unter MacOS X

Es gibt nicht viel zu sagen, da du dasselbe tun mußt wie oben. Hier ist, wie es während des GUIentwickelns aussehen sollte:



Und hier der Editor "bei der Arbeit":



Und jetzt... eine wirklich laufende Applikation

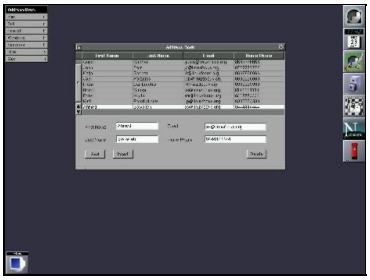
Jetzt wählen wir ein anderes Beispiel von Mike Beam. Dieses Mal eine voll arbeitende Applikation, in der Lage, Daten zu verwalten: ein Adreßbuch. Mikes Tutorium über das Adreßbuch (wie jedes andere) wird zum Lesen empfohlen, um zu verstehen, wie das "Ding" arbeitet. Schau dir die Tutoriumsliste an, da Mike verschiedene Schritte des Entwicklungsprozesses für ein und dieselbe Applikation liefert, was es erlaubt, sie zu verbessern. Wieder erzeugen wir die Applikation und lassen sie auf beiden, GNUstep und MacOS X laufen.

Unter GNUstep

Wie du es für das Editorbeispiel gemacht hast, starte ProjectCenter.app. Selektiere eine Gormapplikation und nenne sie AddressBook.Von ProjectCenter aus starte Gorm durch Doppelklicken auf Interfaces –> AddressBook.gorm. Ziehe eine TableView aus der Palette zu dem Defaultfenster. In anderen Worten, folge Mikes Tutorium wie du es für MacOS X machen würdest. Du mußt ein paar wenige Dinge anpassen, da sie in Gorm anders arbeiten als in InterfaceBuilder.
Zum Beispiel die Anzahl der Spalten in der TableView kann in Gorm nicht vom attributes inspector definiert werden. Um die Dinge einfach zu halten, kopiere einfach eine Spalte und paste sie nebendran, um die erforderliche Anzahl (4 in unserem Fall) zu erhalten. Du solltest soetwas wie das hier bekommen:

Wenn du damit fertig bist, speichere alles und gehe zurück zum ProjectCenter, um den Code zu tippen oder zu verändern. Für den Fall, daß du einige Fehler machst, liefert Mike dir die vollständige Anwendungsquelle. Wenn du sie herunterlädst, reicht es, den Code in deine eigenen Controller.m und Controller.h Dateien, die von Gorm generiert werden, zu kopieren und zu pasten. Schließe (import in der Objective C Terminologie) *Cocoa.h* nicht mit ein, da es für MacOS X reserviert ist. Übrigens solltest du das Skelett, "geschrieben" von Gorm, behalten und ein paar Dinge ändern. Zum Beispiel ersetze das *void* durch IBAction in Controller.h und Controller.m. Füge IBOutlet vor den outlets id in Controller.h hinzu.
Irgendwie, wenn dir danach ist, kannst du den ganzen gelieferten Code behalten: ersetze nur die cocoa Einschließung durch *#import <AppKit/AppKit.h>*. Jetzt kannst du die Applikation bauen und laufen lassen. Da bist du: du kannst damit anfangen, mit deinem neuen Adreßbuch zu spielen.

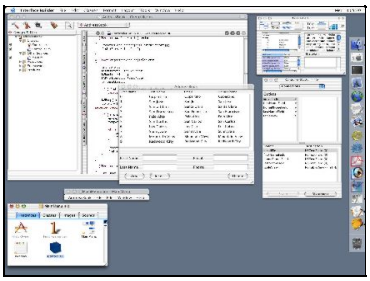
Hier ist, wie es aussieht:



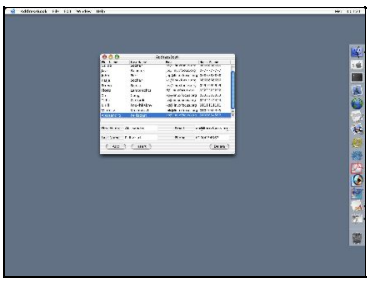
Unter MacOS X

Mike Beam hat die ganze Arbeit gemacht, was könnte ich hinzufügen ?

Hier ist ein Screenshot des Entwicklungsprozesses unter MacOS X:



Hier ist die resultierende Applikation:



So was? Nun, es dauerte ein paar Minuten, um das GUI zu entwerfen und der gesamte Code stellt ein bißchen mehr als 2000 Bytes dar. Nicht schlecht, oder ?
Nochmal, es gibt eine Menge Verbesserungen, die gemacht werden können, aber das ganze Ding läuft! Jetzt kannst du mit dem Code herumspielen, um ein besseres Adreßbuch zu bekommen...

GNUstep und MacOS X

Natürlich können die GNUstep Entwicklungswerkzeuge nicht so voran sein, wie die von Apple. Apple und NeXT stellen eine 15 Jahre alte Erfahrung mit hunderten von Entwicklern dar. GNUstep ist die Arbeit (ohne Bezahlung) von ein paar wenigen Individuen, die noch etwas anderes tun müssen, um sich den Lebensunterhalt zu verdienen. Demgemäß sei nicht überrascht, im InterfaceBuilder z.B. sehr viel mehr

verfügbare Klassen zu finden als in Gorm. Erinnere dich, Gorm ist bei Version 0.1.9 (oder 0.2.0).
Außerdem machten wir die Tests auf dem "schwierigen" Weg. D.h. wir "portierten" von OS X nach GNUstep. Andersherum wäre es leichter wegen der oben genannten Unterschiede zwischen den Werkzeugen.
Zum Beipiel, Portieren von Applikationen, die unter MacOS X 10.2 entwickelt wurden, wäre sehr viel schwieriger, da die neuen Apple Entwicklungswerkzeuge sich stark verbessert haben. Wie schon gesagt, es gibt viele neue verfügbare Klassen oder ausgearbeitertere.
Jedoch, die Werkzeuge basieren auf derselben Philosohphie, ob sie unter GNUstep oder MacOS X laufen... und GNUstep verbessert sich jeden Tag. Eine Sache sieht für mich sehr nett aus: Die GNUstep Leute arbeiten wirklich zusammen. Sie helfen einander, was einzelne Projekte angeht und sie tragen dazu bei, den Kern von GNUstep zu verbessern. Dies ist die Art freier Software zu arbeiten, wie ich sie mag. Glückwunsch zu einem solchen Verhalten Mr.Fedor und Freunde.

Stell dir vor...

Das Ziel dieses Artikels war es, die Mächtigkeit der GNUstep "RAD" Werkzeuge, Gorm.app und ProjectCenter.app zu zeigen. Trotz ihrer "Jugend" können sie helfen, nette Applikationen auf sehr einfache Weise zu entwickeln.
Außerdem bieten diese Werkzeuge einen sehr angenehmen Weg, zu arbeiten, während sie sehr effizient sind. Objective C ist eine sehr kompakte Sprache und nach meiner Meinung sehr viel einfacher zu lernen als C++ für jemanden mit C Wissen (ich weiß, ich habe das schon gesagt !). Dies erlaubt es, nett aussehende Applikationen zu entwickeln (nun, es ist eine Frage des Geschmacks, aber ich liebe diesen Look and Feel), während sie eher klein in der Größe gehalten werden.
Ich muß zugeben, daß ich mich nie von dem Schock erholt habe, den ich bekam, als ich das erste Mal auf eine NeXT Maschine traf. Die Tatsache, daß Apple eine moderne Version von NeXTstep herausgegeben hat, erfreut mich. Dies ist auch der Grund, warum ich Projekte wie GNUstep oder Window Maker so gern habe. Jedoch, auch wenn ich freie Software liebe, bin ich doch kein "Fundamentalist" und von daher bin ich nicht gegen proprietäre Software (nun, vielleicht ein bißchen gegen einen bestimmten Editor... aber nur ein bißchen !).
GNUstep kann von Apple profitieren... aber Apple auch von GNUstep. GNUstep ist kein Applekonkurrent, es ist freie Software. Soweit ich weiß, wird freie Software weitlich in OS X benutzt. Das heißt, sogar noch mehr freie Software auf den Apple zu bringen, kann keine schlechte Sache sein. Was Ludovic und Freunde mit GNUMail.app machten, ist ein sehr gutes Beispiel für das, was geschehen könnte.
"Ich hatte einen Traum"... Apple lieferte den größten Teil seines Entwicklungswerkzeugsquellcodes zu GNUstep. GNUstep und Apple Entwickler arbeiteten zusammen, um eine großartige Applikation zu den Unixbenutzern zu bringen. Und langsam realisierten die Leute, daß sie ohne Windos leben konnten...
Leider war es nur ein Traum ;−)
Nun gut, wenn du GNUstep und seine Applikationen nicht kennst, fühl dich frei, sie auszuprobieren. Erinnere dich, GNUstep ist ein Rahmenwerk und Werkzeuge wie Gorm und ProjectCenter liefern dir alles zum Entwicklen, zum Erfinden. In anderen Worten, mit ein bißchen Vorstellungskraft kannst du "Produkte" entwickeln, ganz anders von denen, die wir heute sehen: Windosapplikations Clone !
Wir leben in einer großartigen Zeit !

Dank...

An die GNUstep Leute: A.Fedor, N.Pero, G.Casamento, P.Y.Rivaille, N.Roard, L.Marcotte, R.Frith−Macdonald, P. C.D.Robert, E.Sersale, A.Froloff, F.Kiefer, M.Viviani, M.Guesdon und alle, die ich vergessen habe, für die großartige Arbeit für das Rahmenwerk oder für seine Applikationen.

An die Window Maker Leute: A.Kojima, D.Pascu und Freunde, weil sie uns ein freies NeXTstep Interface für X gebracht haben.
An J.M.Hullot und B.Serlet für das Erfinden von InterfaceBuilder.
An "Steve Jobs INC.", für das Bringen von NeXT, NeXTstep und MacOS X.
An alle Leute, die hier nicht erwähnt sind, die dazu beigetragen haben, unser Berufsleben weniger traurig zu machen.

2005−01−11, generated by lfparser_pdf version 2.51