Interview with Miguel De Icaza

Tony Mobily

Miguel, first of all I'd like to ask you a personal question: are you enjoying yourself at the moment? How are the United States treating you?

I am having a lot of fun. Working at Novell has been great, and I'm really enjoying watching Gnome and Mono grow from tiny little projects to mature and happy communities.

You are considered one of the leaders of the GNOME project. Apart from having to answer silly interview questions (err...), what's great about it? And what's not-so-great about it?

The best part of the Gnome project is that early on, some of its users recognized that we needed to have not only a technical steering committee, but a foundation that would ensure its development, promotion and adoption.

The single best idea was the creation of the Gnome Foundation, because it formalized a number of processes, helped give a direction to the project and helped various companies join and provide their input to the developers directly: telling us what was needed and what important areas there were to address.

There were obviously growing pains as we went from a group of hackers just working on the project for fun to putting the structure in place, but I think this has helped Gnome tremendously in many areas. For example, a lot of work went into the appearance and usability of Gnome, well beyond the themes and artwork: a concerted effort to create a set of Human Interface Guidelines and the application of these to the whole desktop gave Gnome an edge and set it apart from just being a set of tools for geeks.

Then there were some very positive corporate influences, like the addition of accessible technologies (largely funded by Sun) and the adoption of a compromise for binary compatibility: a promise not to break the interfaces that developers are using for their applications.

The latter is key for traditional software developers who might not be able to afford to recompile their software every time a library changes.

I had a vision about a free desktop. Here it is: there is GNU/Linux (or BSD, or Solaris, or...) that acts as a kernel and interfaces with the hardware. There is the graphical interface (the GNOME libraries) which gives uniform, well documented and "low level" functionalities. And there is Mono, that sits on top of everything, which lets any application run smoothly. In this vision, all "end-user" applications (Word processor, email, etc.) would be Mono applications. This would mean that you can just copy them on a CD, and give them to anybody running any system (Solaris, BSD, GNU/Linux, GNU/Hurd, etc.).

Now... do you share this vision?

That is possible today.

Well, I have a question then. Wouldn't it make sense if an application was, in this respect, self-contained? I love the way MacOs does it: a directory called "SomeApplication.app" contains *everything* needed by that particular application. You can put it onto a CD, give it to a friend, and know that it will work once it's copied back onto a hard drive. The special directory includes everything—the application's icon, its data, etc. If this were true for MONO applications, it would mean (finally, in 2005) the end of the dependency hell for the end user. Am I dreaming?

Mono makes this possible in various forms, for example applications can be completely self-contained and they can run completely off a directory.

Now, to be fair this is not a Mono-only feature, other systems can be built like this. It [spreading an application across the file system] is more of a tradition that has been used in Unix and the GNU project for a while, that we are only now realizing might not be the best approach for deploying software.

To fix this we need a cultural change in the way that we package software and expect it to be deployed. Mono-based applications that have a GNU heritage suffer from the same problems as other applications: they are configured to be

installed on a given prefix, and the tree underneath the prefix is populated with all kinds files everywhere. The systems are consistent, but the deployment is sometimes painful.

What do you think could be a practical "solution"?

I believe that there is a case to be made for both approaches and I think they are both compatible. The GNU-heritage should continue to be used for key components of the system, but the packaging of non-core components of the system could be packaged in a modular fashion that would allow for directory-based deployments.

We must start to discuss these issues openly, and get the various players to the table: autoconf, automake as the foundation that most of us use, and talk to people who have been trying to solve this problem like the installer crowd like AutoPackage and BitRock.

I should not miss this opportunity to plug a feature into Mono that we like very much: Bundles. Bundles are just a mechanism in which you can deploy your Mono-based application and all of its dependencies as a statically linked binary with no external dependencies.

I see what you mean. However... well, this is just an idea. Don't you think that MONO should set some kind of standards then, to which application developers will need to adhere? This should go beyond the ability of running "off a directory". For example, the standard could set what the icon file is, what to run in order to run the application, and so on. What I mean is, if there were the files "icon.png" and "run_me" in each application directory, a GUI would be able to show the directory as an icon and run it.

This is part of the discussion that must be had. The work is not only about the packaging, but also making file managers—aware of these kinds of setups—to do the right thing.

I see this as a critical piece of the puzzle. Is anybody working on this right now? If not, are there plans and resources to do so?

Nobody at this time, but we should start.

Going back to a MONO world... OpenOffice, Evolution, Abiword, are critical pieces of free software, but they are not MONO. Does this mean that we will have to deal with a hybrid world for a long time, possibly forever?

Well, use the right tool for each problem. Mono is well suited for some applications, but not all of them. There is a large legacy that must be supported, and for too many reasons those applications should not be rewritten.

I'd rather see people use Mono as a platform for innovation and for solving problems in new ways or creating new models that we have not thought of yet.

Is there a Kylix-like development tool for MONO? Is it possible for me, for example, to develop a neat database front-end fully visually, and compile it as a MONO application?

There is the commercial X-Develop and the open source MonoDevelop. They are good IDEs, but they are not as advanced as their Windows counterparts.

Do you think MonoDevelop will eventually be as advanced as Kylix or Visual Basic? Or do you think another free software project will need to develop in that direction? What do you think a time frame could be?

Absolutely, I am convinced that MonoDevelop is on the right track. The issues are not complicated, it's just a matter of surface: there are many things pending that people expect to do from an IDE, and each one of those features takes a lot of time.

There is good news though: MonoDevelop is based on the SharpDevelop IDE, so it has already some very advanced features like a tree of your code that is continuously updated as you write code. This is used today for an Intellisense-like completion but it can be used for implementing things like code refactoring or providing more semantic information as the user types.

In any case, this is one area where hackers can make large contributions and influence the direction of the project.

Copyright information

About the author

Tony Mobily is the Editor In Chief of Free Software Magazine.