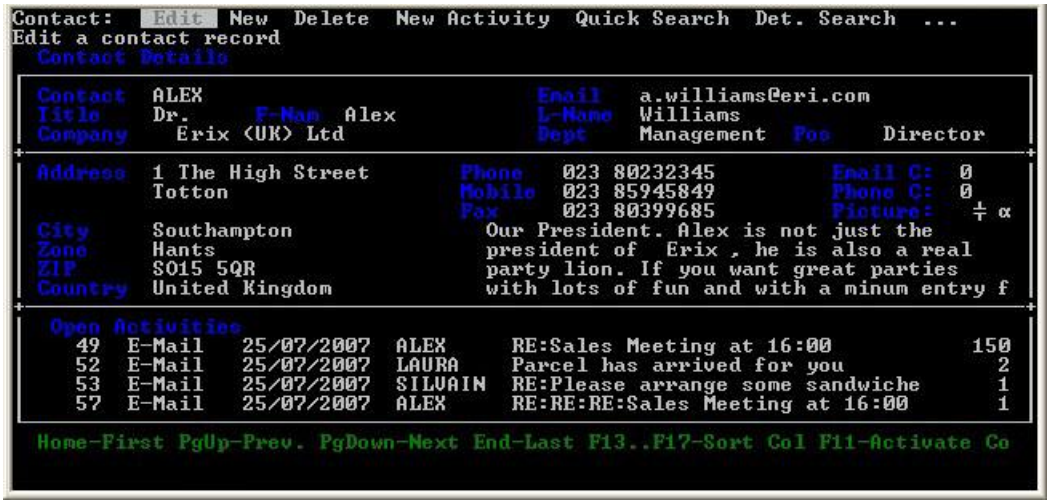
Vercelletto.com

Do the Informix-4GL applications

Have a future?



Study about the appropriateness of discontinuing Informix 4GL applications versus the adoption of a 4GL compatible development studio

Combrit, May 4th 2012

What can we do with those Informix 4GL applications ?

If you happened to find this article, chances are that this question has a real meaning for you.
You, or the former person in charge of your job position, have known pretty well this language that, in early 80's, strongly impacted the way to develop business applications on the very trendy platform at this time: Unix.

At this time, the Unix applications were developed in C language. This 3rd generation language had a strong impact on the developers community because it was the only platform that could almost easily be ported from one flavour of Unix to another. An extremely fast executing language due to its architecture close to the machine level, faster to code due to powerful commands, but not that fast to debug or to run without getting from time to time a sweet sigviolation or core dumped.

In 1984, Informix Software released Informix-4GL, a revolutionary language made on top of C language, containing statements and functions that were powerful and synthetic enough to quickly enable the developers to build interactive business application faster than ever. What was the brilliant idea? To enable the programmer to design screen forms on one side and write source code easily interacting with the screen on the other side. Another brilliant idea? Simply match table columns with form fields with the help of "BY NAME", or define variables "LIKE" these tables columns. A nice and powerful report writer was also part of the product. I will stop the list at this point, but it is far from being complete.

At the time I am writing this article (April 2012, AC), and against all odds, it is hard to recognize a real in depth progress in development technologies. Of course many development platforms will allow you to quickly build more or less functionally acceptable prototypes, with a beautiful user interface though.

But when it comes to detail programming, needed to achieve user's requirements, things quickly turn to difficult and complex. Very often, thousands of unreadable code lines are used to achieve a simple functionality. In addition, some of those languages are slow at runtime and heavy system resource consumers! All these factors will unfortunately bring bad surprises months or years later, when it comes to the user acceptance and to the real application maintenance cost. Is there a real progress in those conditions?

Informix 4GL is still not bad at all, and it is trouble-free!

A remarkable thing about Informix-4GL is the legibility of the code. This has a direct impact on the maintenance productivity, be you an experienced "salt-and-pepper" expert or a young developer full of energy. The less immediately visible parts are the high performance of the programs at runtime as well as their low cost in system resource.

I am curious about looking how would the monthly invoices calculation of a national telecom operator behave if it was written with one of those free interpreted languages (no name here, but property of the Evil Empire), as compared to the same program written in compiled Informix 4GL running on the database server itself!

Nevertheless, and despite those backward-looking considerations that I totally assume, one cannot deny that your application developed in 1993 on Informix-4GL, enhanced on a daily basis, is now looking like granny's face. But it's still there, bug free, running smoothly and responding to your company's requirements!
Herein lies the dilemma created between keeping a ridiculously primitive application that cannot be shown to anyone born after 1970, and remake it from scratch with modern tools, so that it looks prettier and up to date.

Problem definition and working hypotheses

In fact, the question is easy, but the answer is not. If we sincerely consider the subject, we are generally talking about around 20 years of office knowledge, technical know-how, interactions with users and managers packaged in this application. Unless your company's activity has radically changed since your last modifications, and the whole development team has totally disappeared, the investment for this application is very heavy, but the breakeven time for your investment was about 15 years ago!

Let's now consider the following case : «our old application goes to trash, and we rebuild all from scratch"
What are the possible solutions?

- Solution #0 : *we purchase a ready to run ERP solution, and we have a "specialized" integrator customize it for our activity*.
  I have worked for 8 years as an employee, then as a partner of a famous vendor of this kind of solutions. What are my conclusions? Those applications are very expensive license-wise, and deadlines for application customization are never met, far from it! I also noticed an unbelievable proportion of cancelled projects, due to lack of relevant functional analysis and inexistence of gap analysis. The third reason is the tremendous increase (multiplication) of budget either because the product purchased is not the right one, or because, apart from the project manager, the majority of the developers are learning the development tool at your own cost. Also consider that some software integrators will push you for this kind of solution just because they will make more money with it, totally ignoring your budget, your requirements and finally your satisfaction.

- Solution #1 : *we migrate the database to another vendor and we remake the application with modern tools that work on this new database*.
  This is what I will always call a stupid decision. Its consequences will make the competition happy, and make you say in a few months: « finally, with Informix databases, we didn't have performance and stability issues, and we had 3 times less DBA's administering this application". In my Informix life, I have seen a number of projects switch to another DBMS vendor, then come back to Informix because it simply did not work and users were highly unsatisfied. If this solution is your decision, you can stop reading this article and we'll talk again in a year or two.

- Solution #2 : *we keep the Informix databases (wise decision), but we get rid of 4GL because the interface is out-dated and the young users do not accept it. In addition, we need a web based interface.*
  These arguments are solid and acceptable, but the trade-off is costly as explained for solution #1. In addition, the application runs great, few or no bugs, the end users handle it quite fast and efficiently. A total change can only introduce an important number of new risks, such as:
  - Risk of losing functional analysis quality, generated by the fact that it will probably be conducted by people who do not know your company, all internal non-document business processes and how it works.

- or, if your 4GL team leads the project, it will take a considerable time for the team to get familiarized with the new development tool, and beyond simple knowledge of the product, it is not easy to re-capitalize 20 years of 4GL experience in 6 months or even one year.
- Simple risk of budget and/or deadline rip-off, just because it happens too often in IT projects,
- Add to this the fact that the application will be different, so end users will need time to be efficient with the new application.
- And finally the risk of losing end user adoption during some time, because the 4GL application was working well and it was simpler to use.

You can effectively achieve the target with this solution, but the risk is high and you need to carefully staff your project team.

- Solution #3 : *How about conserving my whole 4GL application and giving a great GUI as it would deserve?*
  If you are not a 4GL fan, or you prefer other environments just because they are more robust, faster in development, faster in production and easier to maintain, you can stop reading here: we have no solution for you.

  Nevertheless, if you really have estimated pros and cons of rewriting a brand new application, you probably got to the conclusion that if you could give a graphic user interface to your 4GL application without having to provide a big effort nor to learn a new language from scratch, you would probably have in hand the less risky and quickest solution.

The need for evolution has been identified for a long time

I won't talk about the history of Informix-4GL and its successors that have been forgotten due to lack of clear and persistent marketing strategy. Since about 1996, non-Informix companies understood that Informix Software was building a huge vacuum in the users demand, i.e. the lack of a graphic interface that could easily work on a PC instead of a vt100-like terminal.

Two actors started to explore that field: a French originated company (Fourjs), as well as a British one (Querix) understood the opportunity created by this vacuum. Both companies placed a bet on the concept to keep the Informix 4GL language, because it was a popular and productive language, and to build a compiler/interpreter allowing to have the application run in a three-tier architecture with a X-Windows, and later Windows client. A few years later, the family was extended by an open-source project: AUBIT-4GL.

During this long desert crossing of the Informix brand, just after the IBM acquisition, the « x4gl » family of products kept growing out of the public sight, probably due to the fact that many competitors (and not only competitors), said that Informix was definitely a dead product.

This was, we now know it, without reckoning with the determination of the Informix user companies to keep using a product that gives entire satisfaction. A fact is that those x4gl products not only are still active, but they are in clear expansion in terms of sales and functionality.

At the same time as the common sense is surfacing again, the fact that those applications are running on expensive maintenance hardware, and above all threatening to collapse at any time, is becoming a critical question to be urgently addressed.

After the IBM acquisition, Informix-4GL changed very little as expected, apart from a few maintenance release and some new functionality such as Dynamic Arrays (at last!) and limited support for Webservices. Fortunately, the x4gl products have been on a persistent development and extension trend.

The « x4gl » family is the best answer to your requirements!

Starting from the initial objective which was a mere revamping of Informix 4GL applications, those products progressively got promoted to real development studios, enriching the Informix 4GL language with crucially missing functionality such as ascii file IO operations, dynamic arrays and many more.

Beyond the language aspect, the big advantage of those solutions is the use of a three-tier architecture, whose components are the database server, the application server and the clients. Mike Saranga(oh! DB2's creator and also main architect of the Dynamic Server Architecture) was an adept of this architecture and applied it to the Siebel CRM product, product nowadays in the enemy's solutions catalog). The principle was to go beyond the client-server concept in order to reuse the transaction monitor technology and extend the concept.

By splitting the user data presentation layer from the data handling layer, the client is freed from all the application logics, from now on handled by the application server in charge. The client just handles the user interface, significantly decreasing the PC workload as compared to what the client-server architecture would result. We are now in a situation where the server is less solicited than it would with a terminals-based configuration, but also the client is less solicited.

From here to totally get rid of the client, the distance was short, and the actors of the x4gl family. The user interface layer being separated from the SQL layer and application layer, it is easy to adapt it so that it can work with an internet browser, thus making the "zero deployment" or quitting the necessity of installing anything on the client. This is an enormous advantage when we talk about a nationwide or worldwide deployment.

The most recent versions of those products also feature a nice and easy integration of other languages. It is a reality that for instance Java can do things that 4GL will never do. It is now-on possible to integrate portions of java code, invoke methods and declare objects from the 4GL application, exactly as you would call a function or define a variable in genuine 4GL.

The report generators have also done significant progresses. Beyond the very mighty, but too much text-oriented 4GL REPORT functionality, the vendors have extended the concept way further by including a graphic and WYSIWYG interface, such as Genero Report Writer or LyciaBI by Querix. After being built with a totally graphic design tool, the reports can go beyond the simple report aspect and be classified as real Business Intelligence documents generators, and compared to BI documents produced by the major BI tools. They are executed thru a dedicated report engine and can be saved under a large diversity of formats (Excel, Word, pdf and many more). Of course, a set of dedicated functions will allow you to handle your reports from the 4GL application.

Did I hear that you can have your 4GL application work with other DBMS brands ?

The x4gl companies had also understood another message at this time: if the 4gl language is that much appreciated in the developers community, why shouldn't they have it work with other DBMS brands? Thanks to the principle of abstraction layers, it is possible since many years to have those applications work with DB2, postgreSQL, MS Sql Server, mySql and even Oracle! Why in this case do not use your x4gl product to streamline your too dispersed development environments?

Let's face the facts : how many years will this project take ?

A migration to an x4gl environment generally takes two steps :
Step 1 : initial compile

For the programs that do not include system-specific commands (cp, lp, cat, echo etc…), the initial compile is generally a matter of minutes. For more complex programs, some edition of the code may be necessary to resolve minor issues such as system specific calls for instance. According to the size of your application, your application will be ready to run on your target environment (Windows client, Web client, mobile client) in one or few days. Although it will have a basic interface, it will be usable "as is" as long as the users community temporarily accept a transition phase. The interface already includes the use of the mouse, buttons and some basic graphic objects.

Step 2: Advanced re-definition of the GUI.
The task will mainly consist in re-designing the screen-forms layout in order to achieve an elaborated GUI. You can define your graphical chart (colors, fonts etc…) thanks to the THEME DESIGNER. This step will take longer than the initial compile, but remember that this is where you will bring added value to the migration project. It is to be noted that, according to which product you use, you will work more time on the 4GL code (Genero), or on the screen forms (Querix Lycia has chosen this option so that the 4GL code remains untouched). You main also want to skip a deployment after step one and concentrate on the beautifying action of step 2.

It will be up to you to enrich the functionality of your application by using blobs (images, Office douments, pdf, videos, etc.), or even implement Java functionality, so that you can give your application the new impetus it needed. But again, you will be developing with a well known language, therefore you will better control the deadlines.

Conclusion :

I heard so many times the following words: « I had never seen a single line of Informix 4GL code before, but I must admit that, after having used it for some time, it is very easy to read and even a non-experienced programmer can quickly understand in detail what the program deals about. "

Despite all the stories about the "new" languages, they don't have the flexibility, the ease of read, the stability and the performance that one can require in a production environment. This topic, hardly discoverable at the early stages of an implementation project, can be experienced as long as the time goes on mainly by deadline missings and budget uncontrolled increases. Also despite the fact that Informix 4GL looks obsolete and "has been", it remains a programming language rich in

functionality and above all terribly efficient in terms of productivity and maintenance ease.

And last but not least, not only the x4Gl development systems vendors did enrich the 4GL language to bring it up to the functional level of the “new” languages, but they also have managed to hide the programming complexity so that they kept the so much appreciated simplicity of the 4GL language.

So, are your still about to throw away an application that works fine, to replace it with an application that you don’t know when it will be ready to be rolled-out and if it will be accepted by the end users?

Appendices

Click on the logos to visit the vendors site. Free downloads are available