White Paper
Holger Klemt, August 2018

Is the Pascal language finally dead?!

Or is it? Or perhaps even the opposite...

I'm known to most readers as a long-time advocate of the Pascal programming language. It all started in a computer science class at a school in Oldenburg, on a Siemens PC, a few of which our school had back then. It must have been around 1983 and the C64 computers, which were already abundant in various department stores at that time, did not really impress me. You could spend some free time on such PCs, get yourself a book, which was fortunately in the same department, then copy a few lines of basic source code from the book, start and hey presto, the whole thing did what you expected! But it wasn't really that exciting.

After all, the staff in the computer department at the Horten department store in the Oldenburg market place not only left you alone during the day, but back then I even had a positive influence on computer sales in the pre-Christmas period, because apparently I was the one-eyed man among the blind and quickly realized that the sales staff there had little knowledge, and were happy that their customers could discuss details with me, such as whether a datasette is really important, or whether you could manage without it. For younger readers: https://de.wikipedia.org/wiki/Datasette, maximum 1 MB storage space on a 30 minute audio cassette. The number of minutes would have also been the total loading time, but since not a single computer had 1 MB of RAM at that time, loading usually didn't take that long. What is an audio tape? If you don't know, ask your parents or grandparents, they will know that.

At that time the C64 world was quite interesting, but from my point of view still far too focussed on games. Even today most computer games bore me after a few minutes. The then wildly mixed goto commands in numerous basic source codes did the rest, and somehow the whole thing wasn't exactly brilliant.

In computer science lessons, a relatively young teacher tried out his skills, learnt at a recently attended computer science course, on us. His comparatively modest pedagogical skills, if I remember correctly, had to do with the fact that on his first educational path he had completed an apprenticeship as a butcher. Which of course, does not necessarily have to be a disadvantage when prevailing against cheeky students, if necessary using appropriate tools which he could certainly handle well.

Interestingly, there was a Basic version on the Siemens PCs, but fortunately his course had taught him the basics based on the Pascal programming language. Because these were the only documents available to him, the computer was now fed with Pascal programming language for one or even two hours a week. Some of the students were much more enthusiastic about this than others.

Thomas Mönkemeier happened to be at the same school and in the same year, who may still be known to some of the older readers with his shareware VGA Copy, which he was to later release. Perhaps also because his software was able to copy much more to a floppy disk than should actually fit on it. Or also by the fact that in the search for even more free bytes, parameters outside the specification could be set in the software in order to manipulate, or also irreparably destroy, the then very expensive floppy drives. Who takes the software manufacturer's warning seriously? ….

After passing my Abitur, I went to the German Bundeswehr in 1987 for the usual 15 months National Service, basically because I was too lazy to refuse, and following the pretty pointless period in basic training, I was very lucky to get a job in Oldenburg. As an air force soldier in a large army barracks I was the driver for a Lieutenant Colonel and former Starfighter pilot, a very influential person for me and an eternal role model! He too, could not really get along with the structures in the Bundeswehr. But as you know, there are no other employers who can provide such a nice toy as a Starfighter. The three of us, in a small office in these large divisional barracks, together with all the subordinate entities as liaison offices, represented the Air Force. Represented is actually the right word, because if there were no external appointments, we primarily sat in the office and read the local newspaper in detail, interrupted only by lunch or occasionally by visits to my boss' acquaintances.

My boss was aware of my interest in computers, but despite his good position, he wasn't able to set up a computer in the staff building just for private purposes. It was still the cold war era and the armed forces were of the opinion that the enemy could attack at any time. My boss viewed it less dramatically. Because of his rank, he could carry just about anything past the guard without anyone asking what it was, unless it was the size of a computer and monitor that couldn't fit in a briefcase.

So one day he came into the office and gave me and my colleague, in a slightly unusually rough tone, as the "order of the day" that from now on newspaper reading should be limited, because our superior expects more important things from us. His slightly sleazy grin made me aware of the questionable seriousness of the situation. When I looked at the details of the order of the day with my colleague, a long-serving sergeant major, who had little or no IT ambition, the sergeant major said in his inimitable manner that this device was

not part of his previous duties, but he would like to support us by performing the corresponding research in the daily newspaper, which my boss gladly agreed to.

The device was an Amstrad PPC 512 Portable, one of the first portable computers to fit in a briefcase and, as early as 1988, had an amazingly powerful 720 KB 3.5 inch floppy drive. The 9 inch LCD screen with lousy contrast was not so great, but suddenly we had a new field of activity. My boss carried out his first attempts with Basic, but I managed to persuade him to purchase a Pascal development environment, and we spent many hours, days and weeks together trying to elicit what we could from this stubborn device. Gradually other ideas came from his circle of acquaintances and we quickly assembled a small club administration app.

Following my National Service I progressed to dBase in later projects with larger amounts of data. But when I saw the first beta version of Delphi in 1994, I became excited again. Finally this easy to read programming language with a database backend, which covered almost all sizes I needed, and the hype about the new Windows was also covered. Awesome! So that was the milestone for me, and since then I can say that I have rarely missed anything in the Pascal language. In my life I've gotten along without generics, interfaces, and the firm conviction that features do not generally make a language better or worse. Comprehensibility results in written form from legibility, and the wild stringing of special characters is just as enigmatic to me as the obsession with abbreviations in chats or texts of the younger generation today.

But in the last 20 years, time has not stood still and there are now many more computers where I have not been really happy using the Pascal language. If you look in your pockets you'll know what I mean. Smartphones, whether Android or iOS, dominate the markets in terms of numbers. For some years now, Delphi, the best-known commercial implementation of the Pascal programming language, has also been offering these target platforms, but the implementation could be regarded as a "story full of misunderstandings" based on a TV commercial from the past.

Native apps, no matter which platform was used to create them, cannot just simply be sent to the customer like an EXE file and started. Apple treats this even more restrictively than Google for Android, so that a released version can be online in the shop for a number of days or weeks, without you having a chance to forcibly transfer the bugfix version to all users, so that they have to continue working with the incorrect version. This gives the term, Application Lifecycle, a completely new meaning. If the user doesn't want to update because he doesn't start the Appstore at all, the software may have to be blocked by other means to ensure that the rollout is complete.

Everyone must decide for themselves, whether the FireMonkey platform can now be counted among the glorious deeds in the development of the Delphi platform strategy in recent years. I couldn't really get on with it at first contact back in 2011 and that has not changed to this day.

In my view it is incomprehensible that some larger developer teams were thrown out following several takeovers, but then it was announced with a huge amount of hype that Sencha and now also Froala have just been taken over, without announcing any comprehensible strategy why these products should offer such great potential. Many customers report that the new versions are quite good, but they are not used because larger projects can no longer be compiled error-free. For this reason, older XE releases continue to be used. However, if your IDE is still not a real 64-bit environment and still requires .NET, the multi-platform sector cannot really be taken seriously.

But currently I am much more worried about the situation, that there is not really any progress in the FireMonkey world. The mobile device manufacturers Apple and Google have already made a massive effort in recent years with forced conversions to 64-bit and increasingly inscrutable library dependencies, in order to make an application run reliably as a native app. What good is it if only 75% of existing customers get the new release up and running smoothly on their devices without any errors, and unfortunately their own support cannot do anything because the incompatibilities simply cannot be prevented by their own source codes?

Certainly in the past 6-7 years various Delphi customers have tried to get into the app development market. Following a very high learning curve, in which you had to learn things that you did not want to know, for example, to actually publish a software in the shop or just test it on your own device, you realize that the market for apps has once again changed significantly.

A developer who would like to charge 1 € for his app is too expensive. Ad-supported models are used to try to get onto client devices. However customers are now no longer eager for new apps, many areas are covered by existing apps that serve their purpose. The big sales are made in the gaming sector anyway. For a long time, the Appstores were also willing helpers for fraudsters, who then triggered insidious charges by clicking on the wrong page. The direct identification of the mobile device by means of a telephone number is a more rewarding goal than a more or less anonymous PC, where nowadays encryption Trojans have become more common for fraud.

So is app development a feasible market for developers and in-house software departments?

In my view, native apps as a business model are already dead. However, mobile devices are becoming increasingly important for interfacing with the back office for news, DMS traffic, order entry and inspection, CRM reports, and visit reports. Hardly any software manufacturer can get away with not offering his customers a mobile client, albeit a very limited one, making the most important data from the back office available.

How do you achieve this goal, without constructing both a complete Android development environment and a complete iOS development environment, if you are starting from scratch in both worlds and can only earn money with it indirectly, because the customer is not prepared to pay a surcharge for it?

And here we come back to where I started: What areas does the Pascal language cover in the long term?

The trend towards free development tools is undisputed. In the Linux world, no one can imagine paying serious money for development tools. On the other hand, 23 years of Windows source code based on Delphi is a popular argument in the current situation for not being able to easily change.

We at IBExpert have to deal with the same problems and have already ported large parts of our source code from Delphi to Lazarus. Currently, however, this primarily concerns the non-visual part. A 64-bit scripting engine with a subset of IBEScript.exe functions compiled with Lazarus already exists, but is not yet complete. After all, the IBExpert software consists of approximately 1.8 million lines of Pascal source code. When switching from Delphi to Lazarus, we can take over large parts of the source code 1:1 with a few compiler switches, and continue working in the existing development environment, and at the same time create missing components for the Lazarus platform, often even only as a dummy, so that the compiler does not complain.

Completely demonizing components that are not standard on the respective platform is certainly not the way to go. Sometimes you also come across things, the significance of which is only recognized much later. This is what happened to me with the tmssoftware.com FNC components.

While in the FireMonkey world you have to decide to either or, with the TMS FNC components Bruno offers a platform for which he also provides an implementation on all relevant platforms. In addition to the Windows VCL platform common in Delphi, this also includes Lazarus with the LCL platform and the WebCoRE platform.

What does that mean? A cleverly designed source code can be completely identical on the platforms Windows, Linux, Mac and even in a web browser on desktops and mobile devices, so that operating concepts can be implemented in a similar way on all platforms and the user does not have to learn everything from scratch. Nevertheless, it should not lead to the mouse-optimized 27-inch Full HD screen software being transferred to a 4-inch smartphone screen. This attempt is doomed to failure. But a grid can make sense on the tablet. It’s not possible to create a binary app using TMS WebCoRE and Lazarus pas2js, that can only be distributed through the store. The entire Pascal code is transferred to a universally executable JavaScript and can therefore be used on any current JavaScript-enabled browser. The application can be distributed at any time as an e-mail attachment or be downloaded as an HTML container with manifest files, without having to involve a store provider. In other words, just as you've been used to for years in the Delphi and Windows world.

However, web applications require highly complex middleware layers and rest servers and various other character combinations.....

Calculating customer turnover by transferring all orders to the client for calculation makes about as much sense as calling the information desk and having the telephone book read out loud, until you have arrived at the data you wanted. As such information costs 2 € per minute, this is not very useful.

It's nothing new but, for such tasks, intelligent software should answer questions in a mutually understandable language and provide the result in a suitable format. As long as you can determine your own processes, a database server is a good choice. Since this can also play an important role in the implementation of business logic, an open source database, such as Firebird for example, is very well suited. A database server may be supplied by a provider, or operated by a specialized service provider, such as IBExpert GmbH, or operated on its own virtual server instance. This service can be provided by Hosteurope for 10 € per month

(even with Windows at no extra charge if desired) for 100 GB at an acceptable speed for Firebird. Virtually every serious software requires a database as the final instance for data storage. Then why not just bet on Firebird? A multiplatform database server is only important if the customer insists on it. While this still exists in enterprise customer segments, most SMB customers do not care how software stores data as long as it's secure and fast. Due to a mixture of administrative ignorance and an inability to understand physical fundamentals, many religiously-oriented virtualization advocates often fail because of the performance of such database servers. However, this can be easily solved with the appropriate know-how. Just because a computer was expensive and can do something good, it is by no means suitable for Firebird. And even if you replace Firebird with text files in which you store the data, an unsuitable system with the corresponding I/O load will never be satisfactory in terms of performance.

To summarize, the development stack can be limited to Lazarus as the development environment and Firebird as the backend and replacement or addition for a middleware.

What else do you need? Initially absolutely nothing. The comfort and ease of developing web-based applications in comparison to TMS WebCoRE is certainly not directly available in the Lazarus IDE at the moment, but this is only a matter of time and Lazarus is likely to have a visual solution to further develop web applications by the end of 2018, or there will be a TMS WebCoRE IDE integration for Lazarus.

But why not stay with Delphi? Let's just take a quick look back: In the year 2000, the InterBase database software source code was released as open source, because at that time Borland wanted to contribute something to the "dot com" era. When the success failed to materialize, they turned it off again and continued to charge for new InterBase versions. However, based on the InterBase source code, the Firebird project was formed and after a few years, not only outdated InterBase in terms of functionality, but also completely outran it in terms of stability and general performance. No more than 5% of IBExpert customers still use InterBase, and most of them only on systems that have not been changed for years anyway and simply continue to run. Software vendors who bring new customers to InterBase are no longer known to me, and even in the relevant newsgroups there are hardly any reactions when new versions are released.

Or in other words: Open Source killed Closed Source

In the past however, development environments such as Delphi were not yet so easy to replace in their complexity with comparable open source applications. But the lethargy regarding real, new, long-lasting features within the IDE at Borland/Inprise/Borland/Codegear/Embarcadero/ Idera, shows that loyal existing customers are increasingly being forced to pay their contracts on an annual basis, because otherwise they are out of all update cycles, and have to pay 3-4 times to come back to bug fixes, even without any genuine benefits of real new features.

Artificial incompatibilities in a new release version also ensure that nothing can be compiled for days or weeks, because either their own or external components from third-party manufacturers function differently or no longer work at all. It's a long way to go before everything starts to run again as it’s supposed to.

An example of a customer migration of a Lazarus 1.6-based software with Firebird 2.5 as the backend:

- Backup of the FB 2.5 database with approx. 25 GB: 8 minutes
- Remove Firebird 2.5 from server, install Firebird 3.0 with Firebird conf in 2.5 compatibility mode: 2 minutes
- Restore the database on Firebird 3.0: 15 minutes
- Exe files remained in the network and on all machines as they were

So the lunch break on Friday was sufficient to change the whole system. Of course, the Firebird-optimized servers were very important, but with the activated compatibility mode, the clients noticed no difference and a FB 2.5-proven EXE also worked with Firebird 3.0 without any adjustments necessary.

Following the success during the lunch break, Lazarus was upgraded from version 1.6.4 to the new and then current version 1.8.0 - not due to any technical necessity, but to take advantage of the benefits of the new environment.

As I have made some direct extensions in the LazReport components in my application, in order to be able to send the PDF content directly from the print preview to e-mail recipients, I had to add this functionality in the new Lazarus version, which was simply done by replacing the LazReport components provided by Lazarus 1.8.0 in the "components" path 1:1 with the version I already used in Lazarus 1.6.4 with my changes. This package, like three other *.lpk packages, could be incorporated in the IDE within a few minutes, and after about 25 minutes I was able to recreate the EXE and make it available as a test EXE to the test employee in the company, who didn't notice any difference. The whole conversion took less than an hour, and when migrating to different hardware I can get the complete Lazarus IDE up and running again at any time by simply copying the Lazarus path.

I don't want to think back to the days, when people tried to restore a Delphi environment to another computer without virtual machines, so that they could work again. I've certainly wasted a few months of my life in the last 23 years, and I'm now thrilled how easy it can be. So far, it has not been of commercial importance that my complete source code runs with minimal changes on Mac OSX and also on Linux, however the real 64-bit version with a 64-bit version of the IDE under Windows has been. The Delphi IDE is still only available on 32-bit Windows. Multiplatform is something else and preaching IOT as a strategy does not really fit in with, what I see in reality being implemented by Idera.

I like to recall the time, especially between 1995 and 2000, when I did Delphi live programming at the Borland booth at CeBIT and hundreds of people would have taken the product with them there and then, because they knew that it offered so many new and good usable features, that they didn't have to think for a minute, whether the purchase was worthwhile. Even at the Borcon conferences in the USA, the standing ovations were absolutely appropriate, and not only because of highlights such as the Star Wars movie.

However, a lot has changed and the leading edge that commercial providers have enjoyed has either shrunk or even lagged behind when compared to open source products. But here, too, there are more and more providers who view Lazarus as a viable platform, for example, the FastReport and IBDAC components from the Delphi world are fully deployable under Lazarus.

So if you currently decide to renovate an existing project based on the Pascal programming language or to start a completely new project, in my opinion you can do nothing wrong with the Pascal language. With the appropriate support, strategies can be developed so that the transition from Delphi to Lazarus is much easier than you think.

You can learn important aspects at the LazProCon.net conference in Bonn in September 2018.

… and finally:

PS: If you have recognized some aspects of your own career here, perhaps the following image serves as a small reminder. On the lower right you see the PPC 512 mentioned above in the text, which was offered at that time for 1,698 DM (Deutsche Mark). Since I was already acquainted with the portable Amstrad during my time in the German Armed Forces and I was particularly unimpressed by the lousy screen, I went on the search for a suitable computer.

A comparatively well-paid job in the Oldenburg nightlife following my time in the German Forces enabled me to save a little money together in order to purchase a good technical basis for my upcoming computer science studies.

Since the mail order company Quelle was, so to speak, the paper version of Amazon at that time, they always had very good offers and I opted for the device at the top right with a floppy disk drive, an incredibly "large" and "fast" 20 MB hard drive and a monochrome monitor, which could even be operated with a mouse using the graphical MS DOS extension GEM.

Even if this may sound strange from today's point of view, the price of 2,598 DM offered a very good price/performance ratio for a PC. As you can see on the left, the Amiga is in a different price category, and we are not talking about the C64. Incidentally, monochrome at that time meant: Pixel is either dark or light. There were no gray scales. The resolution of 720*348 pixels was however already significantly higher than that which IBM delivered at that time for over twice the price with the 320*200 pixel monitor in a nasty green hue, and which could only be used meaningfully with 40*25 characters in text mode anyway. The additional charge for the color version of the Amstrad PC was at that time too expensive for me and had too few advantages in my view.

The venerable piece served me well for about 2 years, and fortunately, due to a continued good employment situation and also my first programming activities, I was soon able to afford new and faster computers. The burgeoning computer chains such as Vobis and Escom, which were literally shooting up out of the ground at the time, ensured a much wider distribution of computers at significantly lower prices. That effected a very positive order situation for myself as a programmer, because the use of PCs increased hugely, also in smaller companies, and here and there a special program was needed, or my assistance was in demand to get the stubborn things to do that, for which they were purchased.

I am also quite sure that that even today you can still start a Turbo Pascal version on an operational version of this computer, and work with the same programming language as was used 30 years ago. Certainly you would have to familiarize yourself with a few basic conditions again, for example that the function keys are somewhere else and were sometimes occupied differently, but compiling a file test.pas and reconstructing screen interactions with readln/writeln etc. would certainly take less than an hour. Somebody should try that with Java or C#.

PPS: The image of the Quelle catalogue comes from an eBay auction, where you can buy the catalogue for 40 €. I didn't bother to keep the catalogue for 30 years, although if you could earn 40 € today for something you got for free back then, that would be a very interesting yield, but because of "division by 0" both then and now it would also be a very common error cause.

Conclusion:

Not everything that is old is automatically bad, but also not everything that is new is automatically good.

Or, in relation to employees:

Age per se is not a qualification, neither is youth.

It always depends ...