Piotr Kałuski's (too) detailed resume

Before you start reading this, I want to be sure that you understand what this document is about. It is NOT a standard resume. You can find it under this address http://www.piotrkaluski.com/files/Piotr_Kaluski_20180619.pdf.

Read this document in case you would like to know more about my experience to better asses if my skills are good fit to your needs. I tell here a little bit more about, what I've done in my life, in a reverse chronological order. This kind of document is not well suited for presenting a virtue of modesty. Therefore, whatever I've done nicely, I am going to write about it and I am going to express my confidence that what I have done was really good.

Testing and test automation

Major Swiss telecom provider

I currently work on several projects for a major Swiss telecom provider. Here is what I've done there so far and what my achievements are.

I am working intensively on testing the prepaid billing system. But not only testing. I was actually not only testing stuff, but I took the responsibility for everything, which should be done with the package, once it is released from development:

1. Testing the installation
2. Testing the uninstallation
3. Testing the software
4. Preparing necessary installation notes, so the system can be quickly installed by somebody else
5. Supporting installation on production

I've put quite some effort in scripting as much as possible, including very detailed verification of database changes performed during installation. I have actually set some standards on how software installation is tested and documented in the organization.

During the tests I used scripting intensively. As of today I have developed around 40 several scripts. Some of them are sophisticated, some are very simple, but they all make a big difference. They make smart selects that retrieve valuable information instantly. They scan logs and provide important hints in a readable form, otherwise scattered in tons of log files and thus virtually unreadable.

Another area of my activity is revenue assurance system, which verifies that usage is correctly billed. Testing this application has its own specific challenges. In particular, you have to deal with enormous amounts of data. In most cases, when you test you focus on some individual entities or events. Since revenue assurance processed files from GSM network elements, you have to deal with gigabytes of data. You have to find a way of dealing with this. You have to be smart, otherwise the amount of data will bog you down.

I have created couple of really smart Perl scripts, so I can spot the right CDR out of several hundred thousand CDR in couple of seconds.

Also, I have created a framework for automating regression test of this system. The framework was built in Perl (which my favorite language for such things).

Another area – testing web services, which expose billing system functionality to external parties. I was given a task of testing it with SoapUI. Being a person who is curious about tools he puts his hands on,

I have quickly realized, that SoapUI gives some fancy tools for building quite reliable regression tests. So while doing the tests, I have also implemented the regression tests framework. It was pretty sophisticated. Highly parameterized, runnable on different environments. It has served us couple of years.

Major polish GSM provider

Now it's time to talk about CGI-AMS. I started to work in AMS, then AMS was bought by CGI, so when leaving company, I was leaving CGI.

I've been there for more than 10 years, so I have learned a lot there. This was my first "serious", big company after I have graduated. It shaped me. It has some culture, especially managerial one, which influenced on how I do things, even now. CGI was not perfect (who is?) but on the end of the day – it is them who have given me a solid professional foundation, from which I benefit today.

All those 10 years were spent working for 2 telecoms – Polish GSM and German fixnet.

In Poland I have started as a developer but then I have quickly moved to testing (which I always wanted to try). The most enjoyable part of this job was that it was untouched by any kind of test automation. I have started my "automator" career by automating the execution of billing. Simple thing, very helpful, but nobody before me had an idea how to do it in a clear and consistent way.

Then I have started my long, relentless effort to automate as much as possible, in the same time doing normal manual testing. Automation was not easy, since it was difficult to get a budget for it. Therefore most of my automation activities were done during platforms downtime (when normal tests were not possible) or during my personal time. Probably the main factor, which kept me on it and prevented me from dropping the whole thing, was my passion. And the satisfaction coming from observing that my concepts materialized, provided the value which I expected or even surpassed the expectations.

My main goal for automation activities was to streamline the flow between elements of the chain. The process of generating a bill for a subscriber, which used his phone in a particular way, is a chain of activities. Some of those activities were not particularly sophisticated, whereas other were complex. During those efforts I have realized one important think – in the first place we should not try to automate. We should "toolify" instead. What does it mean? By "toolifying" I mean providing tools for making tedious things quickly and reliably. This may sound like the same thing as automation, but they are not the same things. The difference is subtle, but important.

When you focus on automation you risk losing the main point, which is test more quickly and reliably. Automation can be difficult and automation can be done wrongly.

When you focus on "toolifying" your actions, supporting their execution by proper tools, you focus on making stuff more effectively. In the same time you do not resign from automation. You can still achieve it. But it comes more naturally. When you wisely toolify, you may one day realize that if you create some simple interfaces between tools, you might get your tests automated. And if it does not work, you still have your tools, which have proven their value. If you start with the high priority goal to automate, if you take the wrong approach, you may end up with nothing.

So, in my efforts in billing automation, I have started by creating a set of tools, which have radically reduced the time and effort needed to execute steps, which before were done manually. And then, one day I have noticed, that I have all the elements of my chain, and with a very little effort I have connected those element in one string of a very reliable test automation. Those toolified steps were:

- Retrieving test data

- Generating customer
- Generating network usage
- Processing the usage by rating and billing
- Analyzing bills

The final chain was pretty impressive – I had STAF/STAX executing steps on the GUI, automated with Win32::GuiTest Perl library, Perl scripts generating the network usage for tested numbers, billing being running with minimal supervision and that bills being parsed and presented in a way, which made verification much easier.

Development

As of now, my main language, which I chose when I am on a position to choose is Perl. It does have its weaknesses, featuring writing hardly readable code being the most serious one, but it does provide some many charming functionalities. You can implement many things in quite an elegant and clear way.

Apart from that I have experience with JavaScript, Python, Java, and C++.

I have left the university with a strong belief that I want to be a developer. Before I finally decided to give it up and try with testing, I was lucky to take part in some interesting projects.

German fixnet provider

I was a C++ developer in project, which was to deliver a cutting edge billing system for big telecom German operator. The system was written from scratch, which made it a really exciting opportunity. In IT you pretty often find yourself in projects, in which dated software is being maintained. Such a maintenance can of course provide lots of interesting challenges, but it can hardly be compared to writing a new stuff using current, popular and proven technologies. And those days C++ was such a thing. Java was growing up and gaining ground, but was not yet considered to be a viable option for high performance backend systems. I have quickly become an expert in an Order Processing area and were a lead developer in implementation of some sophisticated functionalities.

Before joining aforementioned project, I was working on voting system for the polish parliament. Again, I was lucky to be in the right place at the right time. I was one of the people who designed the concept of the system, its architecture, chosen technologies. We had to design and implement many areas – real time interfacing with voting terminal, interface to database server, using embedded SQL, parallel processing of missing critical logical paths. Tremendous opportunities for learning so much, especially for a young IT graduate

My first serious professional project was a system for storing and presenting sampling results of some measuring equipment. I used C++. Those days PCs were dominated by DOS, Windows 3.0 and 3.1 were on the market but I could not assume it's presence on customers' machines. So I had to create my GUI library myself. I can't think of a better opportunity to understand what windowing system all about is and what kind of challenges you face during the GUI design.

During all those professional experiences I was actively participating in open source projects. One of them was Win32:GuiTest (Perl gui test automation library). I have contributed few functions to it, but my biggest contribution was a very good documentation for it, which included tutorial and pretty good coverage of all available methods.

Another project is my child, lreport.sf.net – it is a tool for comparing csv files. It is written in Perl, has quite useful documentation. The tool is stable and I use it a lot in my work.

Analysis

I worked for Orange, in a special department, which evaluated all IT projects inside this huge company. Quite an exciting experience. It helped me to understand how complex system is a telecommunication company as a whole and what kind of system are being used in all departments. How many players are inside, what are their problems, what are their goals. Very valuable time. Also the pleasant one. The head of a department was a guy who managed to achieve 2 things in the same time – the department was working effectively and smoothly. In the same time internal atmosphere was really nice. But due to personal reason and the desire to take new challenges I have left this lovely place.

Another analytical project which was a tremendous learning experience was a lead analyst role in number portability project in Poland. I had to analyze, which systems are affected by the change. This forced me to understand the big picture of a GSM infrastructure. In many projects you are limited to some specialized area. You understand your part, you understand the interfaces but you barely understand how it all fits together. Number portability project was the thing which helped me to understand the big picture. And my analysis was correct and thorough enough, because once the number portability started in Poland there were no problems resulting from some omissions. Sure, there were areas for improvements, but my analysis has given a strong base, which benefited next releases.

Summary

There were times in my life, when I was determined to focus on some areas and ignore others. So I planned to focus only on testing and test automation for telecoms. But world is changing, profitable areas become not that profitable as they used to be and first of all – IT is too fascinating to limit myself to some narrow areas. So I am opened for opportunities related to development, testing and test automation. Apart from the skills acquired at work, I learn new things, just for fun. Recently I am spending quite some of my personal time on JavaScript and Python. And I am opened for opportunities, which would be both – exciting (so I have fun while working) and would utilize my tremendous experience so I can share it and make my client benefit from it. The last thing is really important. Money matters, that's for sure. But to the same extent it's really important to feel that the person who pays you gets something really valuable in return. Without this feeling of being useful, there is no fun.