Introducing python into industrial environment applications

paper prepared by Fabio Pliger
SIA s.r.l.
fabio.pliger@siavr.it

ABSTRACT

Too many people inside the developers community complain that Python is just another dynamic scripting language, very useful for scripting task but not suitable for serious real world application.

About 3 years ago python became our main development language. This paper covers our experience with python in all those years, how it fits in our applications for industrial production supervision, how it helped to solve many of our problems and why, in most cases, python was the perfect language for us.

INTRODUCTION

Why should someone (really, really mad?) decide to use python for industrial production supervision applications development? Analyzing the industrial reality this might seem like a really bad choice. It's an ugly world…

- people are really mistrustful about new solutions or system changes. Because new technologies = new bugs…

- applications must work with high performance requirements (thousands of information data must be sampled with high frequency (1, 5, 30 seconds….)).

- Many critical applications cannot fail! Some productions batches can cost a lot of money and we don't want troubles!

- Applications must interface SCADA systems and work (well) together
- …

So… how can an interpreted language fit this world? Well… Things are not that bad! Let's give Python a try…

WHY (NOT) PYTHON?

Before introducing python, 99% of our applications were written in languages like Visual Basic, C and C++. The very first impression of using python was a huge time reduction compared to old projects written in other languages. This reduction was about 50% (maybe more…) and includes our efforts to "learn" python and enter inside it's "philosophy".

Python revealed itself a great language for team development because it's simple, clean and its program size is usually smaller. It's less hard to read a piece of code written by a teammate if it's written in python… and helps to debug code. Dynamic typing is a wonderful thing and multi-paradigm helps a team working together.

At the beginning we were really afraid if python would fit our performance needs. In most cases, our applications must communicate with hardware devices (PLC) with harsh timing requirements and it's really important to have no lacks! About that, since our first experiments, python performance was very close to programs written in c (and VB, of course! ☺). The only occasion we saw a real difference between python and c was writing a (particular kind of) driver. We'll talk about that later…

It's cross platform! Well… to be honest it was positive to us only in a commercial point of view, because all our applications must run at windows operating systems (yes.. I've never seen a industrial supervision system running under other operating systems!) and it's not a feature we can use. But our customers were, somehow, impressed that their new applications would be written in a language that is cross platform and has no performance problems! This was a nice positive surprise!

Last but not least, I can say that python also has a really active community that always tries to help you if you have any problems, and it has a huge set of cool libraries that you can use in almost all programming areas and it's beautiful!

A SCADA FRAMEWORK

In industrial production supervision scada systems are like the control panel of a car. It's used in almost all pharmaceutical sites and its usage is growing very fast. But, what is it? I'll try to give a brief description.

A scada framework is a middleware system that runs on a pc and gives users the possibility to monitorize, control, store data of hardware devices (PLC…). There are many systems available on the market but in the pharmaceutical world two of these applications are the main actors. iFix and Wincc.

The main features of these systems are:

- Data Collection
- Data Supervision
- Human/Machine interface
- Alarms Supervision
- Devices Control
- Events Control and scheduling
- Reports
- ….

Simplifying it's architecture, the scada engine contains Process database tags that maps variables from the devices, setting their types, length, characteristics and collect their values with a very high frequency. Scada exposes many ways to communicate with itself:

- OPC DA A&E
- ODBC
- DATA SYSTEM OCX
- Integration Toolkits (VB and C libraries)
- DDE
- OPC DRIVERS

PYTHON FIRST DAY AT WORK

The first project we chose to use python was “the right man in the right place”. We had to develop an application that had to interface a scada framework, ask for some specific data, (in some cases) normalize it and then plot it. The plot had to have some features like, logarithmic scale, visualization switch, zooming, report extraction….

In this case the application was already sold before we decided to use python, so we had a very close deadline. We were looking for a language that could give a set of libraries to achieve our client needs. The only candidate to have more or less all we wanted was python! Python was some really cool frameworks for plotting data!

Finding a cool tool to plot our data was only one of the problems we had! We needed:

- to collect data from a SCADA framework (with no support for python.. of course! Only c and visual basic libraries were supplied by scada)
- to generate cool report documents with some analysis of all this data
- to freeze the application into an executable! Our customers would never permit to install a python build into their pc, because of their standard security procedure
- User interfaces must be very simple! In most of the cases, the users of our applications are factory workers with not much computer experience

This first project was really useful for us to have a first approach with python and to see its power on the field.

Accessing SCADA libraries with python (ctypes) was really fun and after a few times we wrote our own libraries and extended the c/vb libraries supplied by it. We decided to use wxPython for our GUI, py2exe to freeze the application and reportlab for pdf reports. The fact is that every time we needed to do some task, python always had a good solution for it!

At the end of this first experience, the customer was so satisfied with that application that suddenly asked us to implement new features that extend information saved in the SCADA framework. The main result was that python seemed a really good language for us to work with SCADA systems. It’s dynamic typing so we are really free to manage data returned from the hardware machines, saving time for controlling it. Lists and dictionaries are really useful for managing and joining data from many sources. Decorators, generators,

metaclasses are powerful! But actually we didn’t have any particular performance requirement in this project to push python to its limits.

THE DAY AFTER

After this first project we had something like 10 new applications to be developed in no time (having enough time to develop an application is just a dream…)! So we decided to use python in many of them, since it was outstanding during our first application.

The data plot application was updated. It's was much more evoluted and had many new powerful features. Finally we decided that the best plotting framework for us was chaco, from enthought. It's interactive, scalable, scriptable and is very simple extend with the features we need. I must say also that in the last year that the enthought python build is growing really fast and well! Those guys are doing a great job out there! Their svn repository is always updated and the community is very helpful.
Note: Enthought build has many other amazing packages that do some amazing scientific tasks. Kiva, Traits, PIL, MayaVi, Numeric, VTK…. are just some of them!

We had to develop another program that was much more low level than plotting data received from scada. It had to communicate with a PLC device and download data collected in it's buffers ram. This data is collected when the scada system is down. This application seems not much complex but by the other hand it has to communicate with the device, check if it can send the data, receive it, ack that data was received, after the first part of the ram is download it must ask and wait for data to be moved from an eprom ram, syncronize itself with the plc and then restart downloading.

I'm not obsessed by performance over all but in this case python had to show it's muscles an demonstrate minimum lacks. We had other applications written in Visual Basic and C that did more or less the same tasks. Comparing both application I can say that python did the work very well and performed much better than VB and a little worse than C. But comparing with the C program it was much easier to handle data from the device with python using dictionaries. I can say that, yes, python was fast enough for this kind of task.

The cool thing was that at the end of this project we were able to give a "special gift" to the customer adapting the plotting tool to mix data from the scada and the data collected from the plc buffer and covering data missing in the plot when the scada system was down.

THE BATCH REPORT GENERATOR

Sometimes the industrial production batch cycle can be really complex and many machines/sectors are involved in the production chain. Nowadays it is really important to track these production processes (talking about pharmaceutical companies it's a "must"). From the beginning to the moment when the product is finished, all the information about how it's being produced must be tracked, readable and organized.

One scada framework can control many machines but is not able to organize a meaningful report. This huge amount of data (temperatures, pressures, particles quantities, production steps…) is just collected as it is and is very complex to track each important thing that is needed to confirm that the production process of a product is good.

What we did is a high classification of the structure of all the production system and its actors, variables, subsystems and so on. All this data is stored in a SQL database outside the scada database and defines all the connections between the machines, tanks and sensors.

The second step is to map all those objects in the production process with their respective variables in the scada database. So, the program can run outside the scada system, monitor what is happening and at the end of the main production batch can collect data and generate a production batch report with only the information that is important to credit. So data is assigned to each machine that generated it, each phase of each machine is tracked, every alarm called during the batch is tracked and assigned and the user can see, in the report, if all the process is ok or it had any problem. This kind of documentation is very useful and can be attached to any production batch.

Python is really performing for this kind of tasks! All the data structures are handled with list and dictionaries, data search is done connecting directly to the scada system or managing it's ascii data files. Python dynamics typing was really useful. Many of the important variables and it's substructures in the program that refers to database/scada records are assigned dynamically (__dict__).

We had too much information to manage, so string manipulation and the creation of the pdf document slowed down performance. To bypass this problem we decided to handle string manipulation with lists and the .join() string method. It's more or less 10 times faster!

We had to use some timing strategies to handle data arriving from the machines, because digitals and alarms are event based and analog variables are sampled at a predefined frequency.

Different machines can mean different data structures… For instance one machine can handle phases progression with one analog variable and another machine can use digital events! One case is completely different from the other! Python is such dynamic that with some effort and use of generators, descriptors and decorators is possible to manage those differences very elegantly.

The result is that just after few minutes the scada system has finished registering the production batch, the program is able to generate reports of the machines batch connect them and print it all. It can show alarms, analog variables linked to any machine, customize them and remake the report with any configuration the user wants.

CONCLUSION

Well, the conclusions to be drawn from the effort of introducing python in our applications developments is that python payed it back!

It's surprising how python is easy to introduce into new projects. The learning curve is very low and after few days a programmer is ready to start. It's full of standard libraries that, in most cases, do what you are looking for. And if it's not present in the standard libraries, in 99% of cases, you'll find that someone has done it for you (and probably has done a great and performing work!). It's very flexible! We used it for charting and statistics programs, reporting, realtime alarms monitoring, web development…

Python reduced about 50% of the time required for development and code is much more readable between coders working on the same project. Extending or embedding other libraries is easy. Debugging is much easier and nice.

Customers were happy about our results.

Python is not "just another scripting language"! It has it's warts but I think it really amazing. You must use it to understand how it can be so lovable! And finally I can say that it can really fit for industrial environments applications!

THE FUTURE

In the future we look very interested in ironpython evolution. It can be a nice friend. ☺

We are starting to extend some application to web frameworks and maybe network distributed applications to give our customers the possibility to extend visibility on their systems. We are also studying an intranet web server to manage project from different areas (software, scada, plc…) or developed together. For this project we are planning to use a python web development framework like cherrypy or zope, and pysvn for code projects/documents revisions.