UNISYS | Securing Your Tomorrow®

Python for ClearPath Forward® Applications

Enabling Business Innovation on COBOL Codebases

White Paper

By: Arun Ravindran

Introduction

Proven business models are facing unique challenges with digital transformation and changing customer expectations. Organizations want new operating models that react faster to changing environments, integrate seamlessly into their value chain and deliver compelling user experiences.

ClearPath Forward® has been enabling clients to adapt to changing demands successfully while maintaining highest levels of security and reliability. However, their business application codebases are becoming challenging to maintain and enhance with declining skillsets.

This paper looks at how such codebases can be extended using a modern language like Python to meet these new business challenges and fuel revenue growth.

Need for Modern Programming Languages

Many organizations fear the big rewrite. As high as 75% of all rewrite projects have resulted in failure.[1] Large-scale migrations have resulted in spectacular disasters that have proven costly in terms of time and money.

Remarkably, most of the hardware running business applications have evolved considerably over time. In fact, new ClearPath Forward systems run on some of the fastest processors and integrate seamlessly with cutting-edge data center technologies. However, they largely run the same application software written decades ago.

These applications, written 20 or 30 years ago, were written in languages like COBOL, FORTRAN, ALGOL, Natural, RPG and PL/I. Among them, COBOL is the predominant language used in banks, insurance companies and other organizations.

COBOL Concerns

The biggest concern facing such organizations is finding the next generation of COBOL programmers who can maintain their codebases. Young programmers do not find COBOL an interesting skill to learn. It is not taught in any educational institution either.

This shifts the burden of maintenance to elderly COBOL programmers. The average age of a COBOL programmer is between 45 to 55 years.[2] COBOL skills have been steadily declining over the years (main reasons being "retirement and death").[3]

COBOL language is itself quite low-level with no type systems or dynamic memory allocation. It does not support certain modern language features like package management. However, the Agile and DevOps movement has found good adoption within modern COBOL environments.

According to a 2017 survey,[4] over 85% of respondents believe their existing COBOL applications are strategic to the business. However, 77% of them are looking at integrating with open source technologies and are facing a shortage of development talent. A contemporary open source language like Python can address both needs quite effectively.

With a growing backlog, IT departments are looking at modern languages, which can natively integrate and co-exists with the COBOL environments.

Characteristics of an Enterprise Language

A single programming language will not be ideal for all kinds of problem domains. In this paper, we shall focus on languages for developing enterprise applications. Hence, it is important to list down the criteria for picking such a language:

Business-Centric

Business applications are designed to support business workflows and improve productivity. Essentially, they involve consuming large volumes of data quickly and applying business rules with high reliability. This might sound simple or boring from a technology point of view but business rules can get very complex.

This was why COBOL was a good fit for enterprises. To quote Ben Shneiderman (1995):

"COBOL has an orientation toward business data processing ... in which the problems are ... relatively simple algorithms coupled with high-volume input-output (e.g. computing the payroll for a large organization)."

Maintainability

Successful business applications will be in use for quite a long time. The team responsible for maintaining the code may be quite different from the original team that developed it.

Therefore, it must be easy for new developers to understand and extend the code without risking breaking a working application. Languages having high readability and known best practices are appropriate here.

[1] https://freedomafterthesharks.com/2016/06/27/exactly-what-is-cobol-and-why-is-cobol-still-a-widely-used-language-in-it/

[2] http://fingfx.thomsonreuters.com/gfx/rngs/USA-BANKS-COBOL/010040KH18J/index.html

[3] https://blog.codinghorror.com/cobol-everywhere-and-nowhere/

[4] https://dzone.com/articles/do-cobol-applications-have-a-future

Maturity

Corporate IT teams are often designed as cost centers. Hence, to keep costs down, it is important to leverage mainstream technologies that are easily available in the market. A skill that is rare or in the early stage of the hype cycle could be unavailable or expensive to staff.

Hence, enterprise applications are often developed with proven technologies whose skills can be easily found. A fine balance of a mature and yet an active ecosystem would be ideal for ensuring stability and skill availability.

Why Python?

Python is a multi-paradigm open source programming language conceived in the late 80s. Its distinctive features are high readability, comprehensive standard library (fondly referred to as "batteries included") and programming at scale.

It is the fastest growing programming language and has been consistently ranked among the top ten programming languages. Google, Facebook, NASA, as well as several startups are among some of the many who use Python as their primary language of choice.

Python is a general-purpose language, but finds extensive use in machine learning, web development, scientific computing and information security.

Among modern languages, we believe Python meets all the criteria for enterprise development. Let us examine them in detail.

Enterprise Friendly

Python is widespread in the enterprise as it can easily interface with other systems. It has been ported to most major operating systems and developing portable code is quite easy. Python has been implemented on both Java and .NET platforms.

Like COBOL, Python supports fixed-point arithmetic (via decimal module) in its standard library. For intensive number crunching, there are mature libraries like NumPy for numerical computing and Pandas for data analysis. Python's elegance allows the depiction of complex business logic in a highly readable form.

Python also has extensive libraries to support interfacing with most enterprise systems like Active Directory or LDAP. There are adapters for nearly all leading enterprise databases.

High Productivity

Python code is shorter and more expressive than most programming languages, often by orders of magnitude. The apparent simplicity of the language for achieving complex tasks makes it ideal, not only for prototyping, but also for building large maintainable systems.

Expressiveness and maintainability is hard to balance. Consider PHP, which is popular in many enterprises thanks to its expressiveness and ease of deployment. However the language provides very little safety against security vulnerabilities and large codebases can become difficult to maintain. Python based web frameworks like Django can be more secure and an ideal replacement in such situations.

Python's gentle learning curve helps teams ramp up and be productive in significantly shorter time. A large ecosystem of existing libraries (167,860 in the PyPI – Python Package Index at the time of writing) catering every need help jump-start projects rather than start from scratch. There is also a rich ecosystem of developer tools and plugins, which further accelerate developer productivity.

Skill Availability

In many leading colleges and schools, Python has become the introductory programming language of choice.[5] Python is seen as a good balance of being easy to learn and placement oriented due to its wide acceptance in the industry.

There is a plethora of free and online resources to learn the language. Many of these are self-trainings, by which programmers can pick up the language quickly. In fact, many non-programmers are also attracted to Python, as seen in the scientific community.

Easy Maintainability

Python imbibes structured and object oriented paradigms suitable for programming "in the large". It is strongly typed but dynamically inferred. This means types need not be explicitly specified (though newer version of Python support optional type hinting) saving development time while allowing detection of type mismatches at runtime.

It is safe and secure. Python checks for common errors like array out of bounds and string encoding issues ignored by traditional languages like C.

[5] https://cacm.acm.org/blogs/blog-cacm/176450-python-is-now-the-most-popular-introductory-teaching-language-at-top-u-s-universities/fulltext

This gives you the best of both worlds. Combined with comprehensive unit tests, agile teams can easily refactor and add new functionality with minimal risk. Python coding guidelines and best practices are well-documented and followed across organizations.

Strong Community

From the very beginning, Python has had a strong and active open source community. Being not only cross platform but also pre-installed in most operating systems, the user base is vast and growing.

Across the world, there are thousands of Python in-person meetup groups, which cater to even specialized interests like Python Data Science. Even more numerous are online Python communities on Reddit, Quora, StackOverflow and perhaps the oldest on IRC, which have hundreds of thousands of members. You can post a query or issue in such communities and answers would start flowing typically within minutes. In fact, most common issues are already answered and are just a search away.

The community is diverse and welcoming to novices and experienced users alike. This has led to higher code quality through peer review and growing innovation from millions of contributors.

Python on ClearPath Forward

You may wonder if Python can run natively on ClearPath Forward platforms. Python applications have been recently tested successfully with efficient access to native computing resources on both MCP and OS 2200 operating systems.

Most importantly, these applications run in an environment as secure as a native MCP or OS 2200 application. This environment is carefully designed to protect data security, maintain resiliency and control resource usage. All this runs with minimal overhead by using lightweight virtualization.

Similarities Between Python and COBOL

Python and COBOL are two very different languages. However, there are some striking similarities when you compare them:

- Both are readable by non-developers. Often business people can understand the code without prior training.
- Both are imperative and procedural.
- Support for fixed point decimal arithmetic (especially for financial calculations).
- Suitable for large scale data processing.

However, there are numerous differences too. Notably, COBOL is extremely verbose and Python is quite terse for expressing algorithms. While COBOL has found a niche in writing business applications, Python is prominent in several domains like data science and web development.

In short, Python shares many of the readability and scalability advantages that COBOL has enjoyed over the years. This makes it a suitable contender for being the language of choice to extend COBOL applications.

Myths About Python

To provide a balanced view, let us look at the common criticism directed at Python and understand if they have any basis and if so, how companies are overcoming them.

Python Is Only for Scripting

Programming languages evolve over time. In the early nineties, Python was popular among system administrators as a scripting language.

Scripting languages are typically defined as interpreted languages, i.e. which do not have a compilation step. In fact, Python is always compiled into bytecode (whose output is commonly seen as .pyc files). Then the bytecode is executed by the Python virtual machine. Hence, the term “Python interpreter” is a misnomer.

Java, follows the exact same execution model as Python. Java is never referred to as a scripting language. Python also has a just-in-time compiler implementation called PyPy as well as ports to .NET (IronPython) and Java (Jython) platforms.

Today, Python is used as a general-purpose language by varied organizations like Merrill Lynch, Cisco, NASA, VMware and Philips for building various business applications.

Python Does Not Scale

Organizations like Dropbox, YouTube and Instagram rely on Python to support millions of users. They have well documented case studies of successfully scaling horizontally and vertically over a short period.

Technically speaking, Python has a Global Interpreter Lock (GIL), which prevents multiple threads from executing Python bytecodes at once in the same process (thereby ensuring thread safety). In practice, this is not a big concern, as scaling is possible using approaches like multiprocessing or even using alternative implementations such as Jython.

Python Is Not Performant

Python is not the fastest language in terms of raw speed like C or C++ because it was created with higher developer productivity in mind. This is an acceptable tradeoff for most companies as developer time is more expensive than computation time. In fact, Python is faster than many similar languages like Ruby or PHP.

Most enterprise applications are IO-bound rather than CPU-bound. In most cases, Python is fast enough and performance is usually limited by other bottlenecks like the database or network.

For performance-hungry use cases, Python has several techniques to speed up code from using native code libraries to switching to faster runtimes engines like PyPy.

Alternatives

Python might not be well known for building enterprise applications. Surveys indicate the top five languages used for enterprise development are Java, JavaScript, C++, C# (.NET) and Python.[6]

We will look at each of these languages from the perspective of their suitability to build or augment enterprise applications on ClearPath® infrastructure.

Java

Java was popular due to abundant availability of developers as it was a commonly taught language. However, Python has dethroned that position in most educational institutions.

Java's verbosity due to large amount of boilerplate makes it unattractive for Agile development. Newer languages are better and easier than Java. Even though other JVM based languages like Kotlin, Scala, Clojure and Groovy have a huge mindshare, Java language itself is no longer a popular choice among current developers.

From a security standpoint, hundreds of Java vulnerabilities were discovered in the last couple of years – including a critical vulnerability that affected more than 1 billion Java SE 5, 6 and 7 users. On the other hand, only a small number have been identified in Python.

C#

C# is the most popular language in Microsoft's .NET ecosystem. In the last few years, .Net has evolved from being predominantly a Windows based platform to being a more open and cross platform framework.

The library ecosystem is rapidly growing with Microsoft placing heavy emphasis on open source development. However, .NET is still primarily run by Microsoft and based on Microsoft technologies. Although .NET Core and Xamarin are open source, the entire ecosystem is a long way from being community driven.

C# is statically typed object oriented programming language, which is used by small companies and large enterprises. Originally, it grew out of Java and Microsoft kept adding new features. It has grown quite big over time to maintain backward compatibility. This can be quite challenging for new developers.

[6] https://jaxenter.com/cloud-foundry-enterprise-languages-148591.html

JavaScript

JavaScript is becoming popular in the enterprise as a common language to develop browser based apps, mobile apps, and server apps. Nevertheless, enterprise development involving large teams need a better type system for building robust software.

This has several "wrapper languages" that support optional typing such as TypeScript by Microsoft, Flow by Facebook and Dart by Google. Without a clear winner among them, this has somewhat fragmented the community.

JavaScript ecosystem, which includes Node.js and libraries, is constantly changing and often breaks backward compatibility. There is also a lack of integration to products from major enterprise vendors like SAP, Microsoft or Oracle. All these factors make JavaScript less attractive for enterprise development.

C++

C and C++ are systems programming languages that are ideal for coding close to the metal and extracting every bit of performance. However, the lack of memory safety leading to catastrophic security vulnerabilities makes it a poor choice for many business purposes.

It is also a poor fit for most financial and business applications involving fixed-point arithmetic and record oriented I/O. C++ is a difficult language to master with many corner cases. The compiler places a lot of responsibility on the programmer to manage low-level resources allocation and deallocation without leaks. This makes truly competent C++ programmers very hard to find and high in demand.

Growing Language Diversity

Enterprises have broadened their choice of languages quite significantly in the last few years. While we talked about the most well-known languages, there are many smaller language ecosystems like Go and Scala gaining rapid adoption.

This change is due to several factors such as flexible cloud-based platforms, Microservices based architectures and agile development practices. Today we have a lot more flexibility to build new services in a different language or combine multiple languages in the same project.

No single programming language would be appropriate for every kind of organization. Multiple teams with different implementation choices can freely interoperate with good API management practices.

ClearPath team's Modern Language Initiative recognizes this need to support diverse languages. Future releases will focus on expanding native capabilities to other languages.

Conclusion

Enterprises have been using languages like COBOL for developing business critical application for decades. These skills are becoming rare and maintenance is getting expensive. We examine the possibility of a modern language that meets all the needs of an enterprise language.

Python is one of the most popular and fastest growing languages today. We find Python to be not just enterprise friendly but has several significant advantages for building business applications.

We attempt to define an objective framework to evaluate the characteristics of an enterprise programming language. Then we evaluate Python against these parameters and find that it fares quite favorably. Further, we also dispel various myths about Python, which might crop up while considering it.

Finally, we look at popular enterprise programming languages to understand how they fare as alternatives. Every language has its unique strengths and weaknesses. Python might be an ideal choice for building and enhancing maintainable ClearPath applications leveraging readily available libraries and a strong community.

Ask your Unisys sales representative about Python and the Modern Languages support.

For more information visit www.unisys.com

Printed in the United States of America 09/19 19-0493