OPEN SOURCE LANGUAGE AUTOMATION PRIMER:

VALUE STREAM CREATION

ActiveState®

TABLE OF CONTENTS

MESSAGE FROM BART COPELAND, CEO & PRESIDENT

"**Polyglot is killing the enterprise.** There is a void in the open source ecosystem when it comes to languages. And keeping open source language builds up to date at scale is virtually impossible."

Bart Copeland, CEO & President, ActiveState

After more than 20 years building open source languages for 97% of the Fortune 1000 and millions of developers, we've learned enterprises can't gauge the risk of their polyglot environments and are taxed with developers wasting time retrofitting languages.

The first building block of most software applications is an open source language. Yet the industry continues to be plagued by disparate tools, manual build engineering processes and lack of visibility of open source languages in production. In fact, our annual developer survey for 2018 reported that 67% of developers wouldn't add a language because of the associated hassles and risks.

As the leader in Open Source Language Automation,
ActiveState pledges the following:

As a member of the open source languages industry, ActiveState sees the increased usage and complexity of open source tech stacks in agile development, DevOps and run-time application quality, in environments in which people and applications are increasingly dispersed yet connected, as major megatrends impacting our industry. We believe the open source language industry has the opportunity to automate and manage the certification, build, deployment and operational management of open source to help organizations accelerate their velocity of delivering secure innovative applications in our increasingly connected yet fractured industry. To advance our industry, ActiveState pledges to offer a new framework by which open source languages can be built, certified, deployed and resolved continuously and automatically to help organizations leverage their polyglot environments and deliver innovative applications to differentiate against competitors and drive desired business outcomes. By supporting Open Source Language Automation, ActiveState will help companies decrease risk to deploy applications across polyglot environments, enable engineering teams to deploy robust applications with speed and security, and free up developers to spend time on high-value work."

As we look ahead we will be working with the industry to build the solutions and awareness that will drive change in how we build, certify and resolve open source languages. We look forward to collaborating with you on this important initiative to benefit our open source industry.
This is the advent of Open Source Language Automation.

Bart Copeland
CEO & President, ActiveState

ACTIVESTATE OPEN SOURCE TRAJECTORY

ActiveState is the leader in open source languages: packaging, distribution and management; and has been so for over 20 years. As such, ActiveState has created the category of Open Source Language Automation. The blueprint to implement Open Source Language Automation comprises four phases: define policies, centralize dependencies, automate builds, deploy and manage artifacts.

VALUE STREAM OF OPEN SOURCE LANGUAGES

The value stream is the foundation for realizing the benefits of Open Source Language Automation. ActiveState created three pillars upon which organizations could measure and build the value of their open source languages. These pillars will enable enterprises to gauge risk and increase developer velocity, and optimize the value open source languages provide.

The three pillars are:

1. Enterprises should create an **application development and delivery lifecycle** approach specific to open source runtimes.

2. Enterprises should create an **application deployment and management approach** specific to open source runtimes.

3. Enterprise should employ **open source value stream management.**

1 Runtime Development and Delivery Lifecycle

A five-step application and delivery lifecycle approach to open source runtimes could comprise: Design, Code, Build, Secure, Deliver/ Release. The components for each step immediately follow.

i. Design: Cleanse existing open source modules by creating a database of deficiencies in the open source modules, packages and libraries under consideration

ii. Code: Modernize open source modules/packages by providing incentives (personal, financial) to identify and improve on the open source package inefficiencies. Enable access to a platform that modernizes and packages open source software for the more popular language communities.

Create machine readable open source development policies. For example, license types such as GPL would not only be in the policy document but also in machine readable code. This would increase development velocity and improve security awareness by facilitating writing tests against the machine policies.

Organizations can define their requirements based on environment and use case. For example, banks have higher security thresholds, if considering threshholds for security versus innovation, open source communities can express security attributes. A usage match could then be produced.

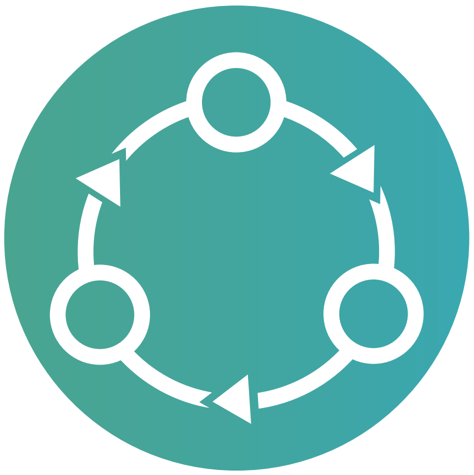
Centralize open source lifecycle management to create a single source of truth for all open source code. This will enable development efficiency and improve deployment quality. Issues created with the high fragmentation of dev tools, communities, and code sources could be addressed. And code versioning and policy management would be improved across all of the apps in an enterprise not just within a singular application CI/CD workflow.

iii. Build: Modernize Open Source Build Management. Build systems are 20 years old. Build processes and systems need to be enhanced by considering the specific needs of authored open source software.

iv. Secure: Create correct by construction security approaches when using open source software in the app dev and design lifecycle.

> Create guidelines for the balance between open source security and innovation. As noted in **"Open Source Language Automation Primer: Industry Insights"**, speed and agility can trump security and conversely "the Department of No" can block innovation. Organizations need to find balance between these two spectrums. The suggested path is to have security and privacy built in and not added on at the end of the application development cycle.

v. Deliver/Release: Raise awareness of wasted time managing open source dependencies. Implement value stream management practices and tools for open source based on application development and delivery. As noted in **"Open Source Language Automation Primer: Industry Insights"**, management does not have visibility or awareness of how much time is wasted managing open source dependencies (onboarding, bootstrapping, etc.)



2 Runtime Deployment and Management

Similar to section 1. above, a lifecycle approach to the deployment and management of open source runtimes could be implemented to optimize the value in open source runtimes.

The three-step lifecycle approach of: Deploy, Manage, Update is detailed below.

i. Deploy: Create consistent processes to manage open source runtimes across any public and private cloud.

ii. Manage: Enable depth in the visibility of open source runtime code through an application management platform. Companies would be able to understand, debug, patch/update, control performance, and control drift of their open source code.

iii. Update: Enable open source infrastructure change management for code to answer questions like: Who put what in the code? Who made the change? When should updates be made? The industry would benefit with versioning in all aspects of the underlying language and its dependencies, not just to the actual application code. e.g. source code dependencies, build versions, data versions are not captured well.

> Create a security vulnerability update process/platform which can certify the deployed open source code and resolve against where code is deployed, notify when policies or vulnerability thresholds are exceeded and alert relevant stakeholders.

3 Value Stream Management

There is a special class of tech debt and process inefficiency due to open source usage. Organizations need to identify the waste in the open source development experience. Industry is delegating a large amount of work to developers to fix the same problems that other developers in other organizations have already solved. e.g. an enterprise might have 2/3 of their dev team maintaining code versus building new code.

Enterprises should employ open source Value Stream Management to address inefficiencies. A value stream is a design process that helps teams visualize and map the steps necessary to take an application from concept through production. The use of value stream analytics, metrics, and data/tool integration is still in its early days in the DevOps design and deploy lifecycle. Nevertheless, value streams are an extremely powerful tool for project teams to determine the true state of their application delivery and deployment.

Value streams provide the ability to improve the percentage of real work performed versus time waiting for inefficient people-based processes, machines and software.

Enterprises should extend their application development and delivery value stream initiatives to include open source value stream metrics. This would incorporate the opportunities and challenges when using open source languages.

Teams must define the open source based value stream, its major process steps, identify issues and areas of team improvement and focus. The result would be the means to analyze and improve open source value streams.

By implementing all of the former, along with value stream metrics, an organization will have successfully implemented value stream management.

CONCLUSION

The Open Source Language Automation Primer provides the foundational knowledge to support the case for implementing Open Source Language Automation. The primer covers:

- 8 industry mega-trends
- 5 industry insights
- 3 value stream aspects

Open Source Language Automation is the new category ActiveState has created so that enterprises can track when applications were vulnerable and respond by automatic enforcement of their open source language policies.

The category is supported by ActiveState's 20+ year trajectory in open source and learnings that will: i) enable organizations around the world with the means to implement Open Source Language Automation; ii) and optimize the value organizations derive from open source languages.

The ROI of implementing Open Source Language Automation will be measured by:

- Developer freedom & value, i.e. are coders spending time on the right stuff?
- Security & stability gain by managing risk of open source language code.
- Cost and time-savings in the management of open source language code, specifically related to maintenance & discovery.

Organizations will be able to automatically build their language distribution, automatically enforce their open source licensing policies, and automatically resolve dependencies, vulnerabilities and other discrepancies as they arise.

Through Open Source Language Automation ActiveState aims to address a marketplace void and resolve the challenges enterprises and developers face with polyglot environments. ActiveState will offer continuous, automatic open source language builds at scale.

Please click here to schedule your readiness assessment for Open Source Language Automation.

Build, Certify, Resolve...
Automatically and Continuously.

Contact us to find out why 97% of
Fortune 1000 use our software:

solutions@activestate.com

ActiveState®

website: www.activestate.com | Toll-free in NA: 1.866.631.4581 | email: solutions@activestate.com