



**OPEN SOURCE  
LANGUAGE  
AUTOMATION  
PRIMER:**  
INDUSTRY INSIGHTS

# TABLE OF CONTENTS

- Message from Bart Copeland, CEO & President, ActiveState ..... 3**
- ActiveState Open Source Trajectory ..... 5**
- Industry Insights..... 5**
  - Companies Need to Automate their Application Lifecycle..... 6
  - Enterprises Need to Invest in Tools and Processes for Application Delivery ..... 7
  - Resolve, Own, Accept, Mitigate (ROAM) for Security vs Development Tension..... 8
  - Containers & Serverless Architectures Need Runtime Management ..... 10
  - AI Needs Single Source of Truth ..... 11
- Next Steps..... 11**



## 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.

## INDUSTRY INSIGHTS

After working closely with enterprises to build engineer open source language distributions, ActiveState is sharing its five insights which provide opportunities for building value from open source languages.

What follows are details on each of the five industry insights listed below:

- 1.** Companies need to automate their application lifecycle to overcome the issues with open source DevOps lifecycle management.
- 2.** Enterprises need to invest in tools and processes for application delivery to resolve gap in awareness of time wasted managing open source languages.
- 3.** Resolve, Own, Accept, Mitigate (ROAM) should be implemented to address the differing goals of application security (control, risk management) and development (speed, agility and leveraging open source).
- 4.** Runtime Management needs to address issues with containers & serverless architectures.
- 5.** AI needs a single source of truth.

---

## 1 Companies Need to Automate their Application Lifecycle

Companies are faced with a number of challenges when it comes to open source lifecycle management. The challenges are widespread and revolve around visibility, security and control. Ultimately, companies are struggling to rapidly analyze and repair non-proprietary code, e.g. open source software.

Companies understand open source code is available freely to use but have difficulty with maintaining security, innovation cycles, and version control within their DevOps cycle. Companies may know a vulnerability exists in their code but they don't know where the vulnerability came from and where it's deployed. Not surprisingly, code visibility and manageability is decreasing. Further, companies can't answer questions like: What team is constantly putting it there? How do I communicate patches and updates to other dev teams? Into production? Enterprises need to improve management of attributes (versions, packages), and answer questions like: Who is going to remediate threats? And what's happening in operations?

In the post Equifax world corporations are demanding even higher security standards. Software vendors that service security-conscious customers have a new security hurdle to overcome. And companies within verticals like banks, insurance, healthcare, etc. know the negative impact a breach can have on their reputation and valuation. Previously, common vulnerabilities and exposures (CVE) weren't considered to be a great risk such that companies didn't resolve them for many years. Now there is a new urgency and sensitivity to not be 'vulnerable', a fix is necessary because of money is on the line.

Separately, the management of open source code can add complexities ranging from third-party services to open source libraries. For example, GitHub is useful as it drives community and open source proliferation but it leads to a host of issues like: Where's the project? Who is in the project and does the developer have the rights to grant the code in the project? How is the GitHub code related to other corporate code behind the firewall?

And open source software oftens contains a bucket of libraries that need to be organized into specific use case packages. But the necessary actions to implement, maintain and resolve usually go unaddressed. e.g. AI needs different Python packages depending on the use case. Who does the packaging? Who versions it? Who maintains it? And who provides notification and alerts for it? These questions require well-thought out answers and actions in order to package open source libraries into an application.

Maximizing application business value and user satisfaction is deeply dependent upon deployed code performance: scalability, reliability and security. There exists a huge need to improve MTTR (mean time to repair). When enterprises use non-vendor supported open source for their applications they can't see what's under the hood to diagnose and repair issues. Poor application performance is experienced by users and results in costs to the enterprise providing the application.

Ultimately enterprises need to automate more of the application lifecycle steps to address their struggle with open source lifecycle management. This is especially true when using open source as a foundation. Organizations can't waste weeks going through an open source review. Automation will decrease wasted cycles and provide rapid analysis and ability to identify what is required to be repaired or replaced.

---

## 2 Companies Lack Awareness of Time Wasted on Open Source Languages

There is a lack of corporate awareness of time wasted managing, securing and administering open source languages

Companies don't realize how much of their business is run on open source or the associated risk they are injecting in their organization. And management does not have an awareness of how much time they are wasting managing open source dependencies (onboarding, bootstrapping, etc.) and what security vulnerabilities exists in which of their apps, or even an understanding of the time lost remediating said vulnerabilities.

This lack of visibility on the impact of open source is not limited to companies. The industry overall lacks recognition on the import and weight of innovation that open source drives. Industry downplays the importance of having a commercial solution around an open source language. This is in contrast with the perception that vendors for other open source components are required, e.g. open source middleware. Yet the truth is that the value/magic of the stack resides in the open source language and there is a gap in awareness of this fact. Ultimately open source languages are what enable innovation. Vendors who sell the databases and operating systems have never solved the open source language value issue.

The lack of corporate awareness of the time wasted managing open source languages can be resolved by an investment in tools and processes for application delivery. Enterprises should consider managing open source through the entire application development to delivery lifecycle. This in turn would decrease both open source vulnerabilities and risk as well as drastically reduce the time wasted managing open source dependencies. Furthermore, companies would be able to drive down the effort and time to remediate underperforming applications. Lastly, the open source management capabilities should be not only for a single application's continuous and delivery pipeline but also address application code interdependencies across multiple deployment servers and clouds.

---

## **3 Resolve, Own, Accept, Mitigate (ROAM) for Security vs Development Tension**

Application security is at odds with the goals of development. Development goals include speed, agility and leveraging open source; these are in constant conflict with application security protocols.

Even with the increasing importance of visibility into vulnerabilities and security threats, these are outweighed by speed, agility, stability and even availability and uptime. Developers often care less about security than application innovation and are pushed to get features to market to drive adoption of a company's technology. This push to get-to-market introduces security risks, especially when using open source. At the same time, organizations will not take their systems down as it injects risk.



In those instances when open source security management is applied, it is heavy handed. Enterprises are not nimble with open source security management, and will structure heavy security review processes (week-long) at odds with the need to get to market faster.

So the tension between application security and the goals of development seems to slide from one extreme of the scale to the other. And regardless of where the weight is held, there is a gap in risk management ownership. Management thinks the team leads owns the risks. In reality the team leads just accept the risks.

Resolve, Own, Accept, Mitigate (ROAM) is a process for managing risk. ROAM should be implemented to address the differing goals of application security (control, risk management) and development (speed, agility and leveraging open source).

When dev teams ROAM, they mostly just Accept it (i.e., when something happens, we'll figure it out how to deal with it), rather than Own it (i.e., spelling out the process for dealing with security issues, new versions, abandonment by author, license change, etc.). In contrast, management believes the dev teams have owned their open source risk and are surprised to find issues with open source persist for months or even quarters after discovery.

A requirement of successful ROAM implementation would be rigour in its implementation, the rules and policies for accountability signed off by both security and development groups. And it would need to specifically address acceptable threat levels for open source code, triggers to resolve threats and sign-off on resolve or ready to deploy code.

---

## 4 Containers and Serverless Architectures Need Runtime Management

Containers and serverless architectures challenge the runtime management of open source code. The use of containers and serverless architectures has presented a challenge with open source code, specifically management of runtime.

Although there are many benefits to containers and serverless architectures they impede open source runtime management. Containers will accelerate the good and bad. e.g. open source vulnerabilities and lack of visible code into production. And since serverless architectures are cloud specific they lock organizations into a cloud vendor, this is sub-optimal.

Companies need to consider containers and serverless architectures not as immune to issues but with their own set of challenges. Company policy needs to consider requirements for the audit, vetting and tracking of open source components running in either containers or serverless architectures.



---

## 5 AI Needs Single Source of Truth

AI has imploded open source code runtime management. AI has many advantages but it is very hard to hire, learn, write code and implement. The explosion of data sources adds to this difficulty.

Surprisingly, the proliferation of AI has created an even bigger open source code runtime management challenge. AI can yield many incompatible design and development approaches and AI packages are use case specific.

Variable use cases and multiple open source community sources for AI packages worsen the ability to manage open source code through the DevOps lifecycle.

To address the variability and lack of controls in the use of open source AI software, runtime management needs to consider and account for AI. Further, a single source of truth, or a de facto standard is needed for AI such that packages aren't use case specific.

### Next Steps

If you're interested in reading the next part in the series "Open Source Language Automation Primer: Value Stream Creation" [please click here.](#)

The next piece details three pillars to measure and build value ActiveState has defined with its 20+ years build engineering open source languages.





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

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

[solutions@activestate.com](mailto:solutions@activestate.com)

**ActiveState<sup>®</sup>**

---

website: [www.activestate.com](http://www.activestate.com) | Toll-free in NA: 1.866.631.4581 | email: [solutions@activestate.com](mailto:solutions@activestate.com)

---

**ABOUT ACTIVESTATE**

ActiveState is a leader in providing commercial level open source language distributions. It provides commercial versions of Python, Tcl, Perl, Ruby and Go. More than two million developers and 97% of Fortune 1000 companies use ActiveState open source language builds including CA, Cisco, Pepsi, Lockheed Martin and NASA. To learn more, visit [activestate.com](http://activestate.com)