Whitepaper

Disaster Recovery for Industrial Automation Starts with Source Control

Adam Gluck
Founder and CEO
Copia Automation

Industrial Operations are Vulnerable

Threats to industrial operational technology (OT) – the hardware and software that monitors or controls equipment, assets and processes – are on the rise. Floods, fires, earthquakes and the like are increasingly causing physical disruption, while cyberattacks have evolved from simple data theft to process disruption, such as shutting down a plant for ransom or even compromising the integrity of industrial environments with intent to create serious harm.

Many industrial sites do not yet have a cybersecurity or disaster recovery plan in place.

> *Natural disasters, human activity, and cyberattacks disrupt the day-to-day operations of industrial sites when no contingency plan has been put into place.*

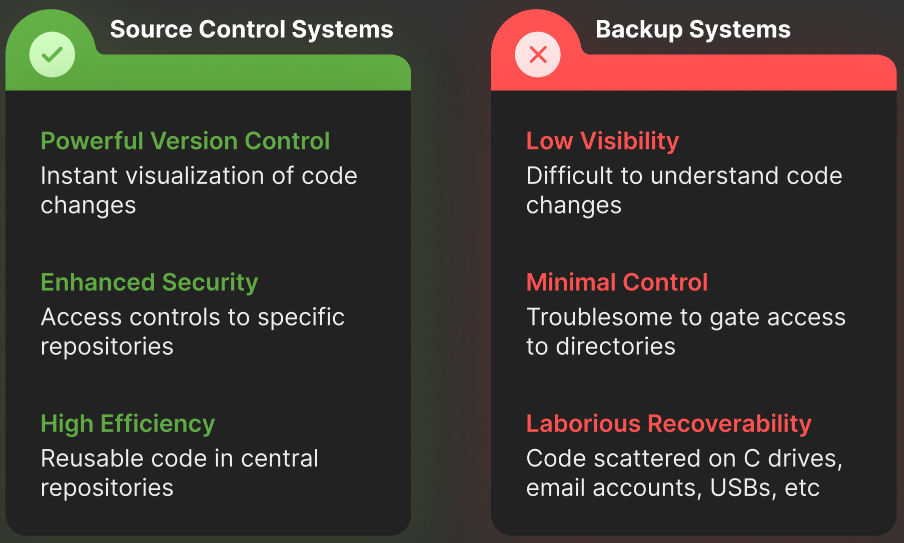
Wam Voster, the senior research director at Gartner, argues that *"security and risk management leaders [in operational environments] should be more concerned about real world hazards to humans and the environment rather than information theft."* Of course, with a cybersecurity or disaster recovery plan put in place, industrial sites could mitigate such hazards. But as Voster identifies, *"organizations in asset-intensive industries like manufacturing, resources, and utilities struggle to define appropriate control frameworks."*

Although the foundation of any process control system is the code written to run it, the struggle often includes poor practices for source-code change management, particularly for programmable logic controller (PLC) programming.

Source Control Offers a Solution

The old way of doing things just isn't sustainable in an industry undergoing the much needed digital transformation. There are several differences between backup systems and source control systems. A backup system provides regular backups of a production system (see **Figure 1**). This is useful, but it's difficult to know what has changed, who has changed it, and when it changed.

But with a proper source control system, it becomes easy to trace meaningful changes and thus root-cause regressions and reduce production downtime. Good source control practices save engineers when they need it the most.



Source Control Systems

✓

Powerful Version Control
Instant visualization of code changes

Enhanced Security
Access controls to specific repositories

High Efficiency
Reusable code in central repositories

Backup Systems

✕

Low Visibility
Difficult to understand code changes

Minimal Control
Troublesome to gate access to directories

Laborious Recoverability
Code scattered on C drives, email accounts, USBs, etc

Figure 1
Source control systems enhance traceability and recoverability, increasing efficiency and productivity for PLC workflows.

Gartner recommends:

> “ *[The OT should] ensure proper backup, restore, and disaster recovery procedures are in place. To limit the impact of physical events such as a fire, do not store backup media in the same location as the backed-up system. The backup media must also be protected from unauthorized disclosure or misuse. To cope with high severity incidents, it must be possible to restore the backup on a new system or virtual machine.* ”

Why Source Control?

> *Source control systems often provide powerful mechanisms for code sharing and collaboration, which streamline operations and save engineering time.*

What is Source Control?

Source control is a file management system that enables people and teams to manage the revision history of their files, typically featuring revision history, version context, shared repositories, access controls, and file review. In the case of a regression or disaster, source control provides important benefits as teams can always access files and retrieve the last known working version of code on the plant floor, avoiding costly production downtime.

The features of source control systems are plentiful:

Revision History

Take meaningful snapshots of your files, returning to or retrieving them at a later time.

Version Context

Assess who changed what, when, and why across the entirety of your organization.

Shared Repositories

Establish file access from a single source of truth, either in the cloud or on-premise.

Access Controls

Grant specific file privileges and permissioning, such as read/write access for different stakeholders.

Code Review

Implement a code review process and review changes before they’re committed.

Rendering

Quickly render files and scan differences between each version -- all without having to open your IDE.

These common-sense features build on concepts that many controls professionals are familiar with, such as a shared network drive or Cloud-hosted storage. The most popular source control system is Git, which was invented by the creator of the Linux operating system and has been adopted by millions of engineers since its creation in 2005.

Source Control Enables Scalability

The adoption of a strong source control system provides numerous benefits that can lead to a return on investment (ROI) for organizations that deploy them. The most rewarding aspect of strong version control practices is the ability to understand what code has changed in case of unplanned downtime. Version control systems enable teams to recognize who made a change to a system, revert that change, and understand the context of that change.

$250,000/hr

Average cost of unplanned downtime

$22,000/min

Peak cost of unplanned downtime

Source control systems unlock new efficiencies between engineers, teams, and organizations. Organizations have invested in extensive reusable code libraries, templated *"master"* copies of their machine builds, and shared reusable components that enable them to kick off new projects faster and more efficiently.

Reliable version control systems integrate directly with a computer's file system, so users do not have to copy and paste files to store backups. They'll automatically pick up on and store meaningful changes. This means users are less likely to lose a file. Consider the engineer whose hard drive failed the day after adopting a version control system. Fortunately, they were backed up in a central repository.

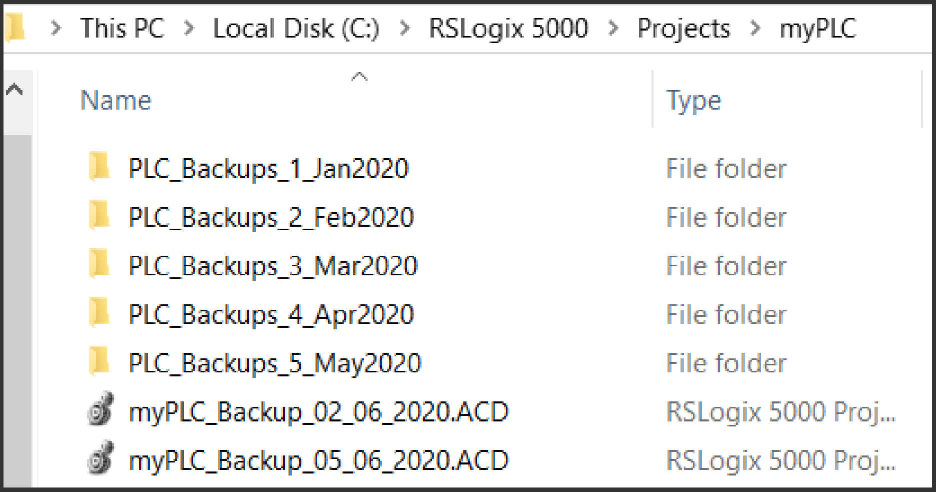
Strong version control is enabling and empowering.

> *Team members can use the correct version of reusable components, capitalizing on a centralized system for documentation. By scaling at new levels, organizations expand their reusable code libraries, grow their IP, and simplify knowledge transfer.*

Ultimately, many contracts and regulatory environments require strong version control practices. Organizations that seek system and organization controls (SOC) 2 compliance must have a system for code review. Many industrial contracts require suppliers to provide first-order traceability as part of their contract.

Ditch Unreliable Archive Folders

To keep track of PLC program versions, common practice has been to have an *"archive"* folder for each project. The old project was named with the date and saved before making edits.



This PC > Local Disk (C:) > RSLogix 5000 > Projects > myPLC

Name	Type
PLC_Backups_1_Jan2020	File folder
PLC_Backups_2_Feb2020	File folder
PLC_Backups_3_Mar2020	File folder
PLC_Backups_4_Apr2020	File folder
PLC_Backups_5_May2020	File folder
myPLC_Backup_02_06_2020.ACD	RSLogix 5000 Proj...
myPLC_Backup_05_06_2020.ACD	RSLogix 5000 Proj...

Figure 2
Archive folders are limited in that they only convey changes in how they are titled.

The typical archive folder is limited in its capability.

Aside from displaying file name changes, there is no context about changes made within archive folders.

However, a true PLC software source control system is much more powerful than an archive folder because it:

- **Keeps** a history of the changes and who made them
- **Saves** only the differences between versions, thereby saving space
- **Allows** multiple people to work on different parts of a project at the same time and merges them together without manually splicing code
- **Requires** users to provide a message to help keep track of why a change was made

Collectively, these benefits lead to increased ROI in operational uptime and reduced risk of costly unplanned downtime.

Source Control is Easy to Adopt

Another benefit of source control systems is that they integrate at the file system layer. Ease of implementation features include regular commits, code review, reusable code, code sharing, and workflows.

1 hr

The typical migration time for organizations.

- **Regular commits:** Engineers need to commit changes to their source control system before deploying to production.
- **Code review:** Engineering teams can begin to develop code review best practices that create a record of feedback on code changes and enable deeper understanding of what changed and why certain decisions were made.
- **Reusable code:** Engineers can invest in centralized repositories and documentation to provide reusable code for their team and organization.
- **Code sharing:** Engineers can invest in access management protocols to share their code with external and internal stakeholders in ways that protect their IP while allowing them to accelerate and improve their collaboration.
- **Workflows:** Advanced engineering teams will invest in automated practices around code changes such as internal reporting and even integration with their deployment model.

Adopting a proper source control system is an easy, cost-effective solution to streamline workflows without creating downtime.

Git is the Differentiator

For many industries, the standard since 2005 has been Git, a distributed source control system that makes collaboration around code extremely easy. Git is available for free as an open-source product, but can be complex to use and difficult to host. Major Git services such as Github and Gitlab have become multi-billion-dollar companies as they have solved these issues for millions of engineers.

Git is an industry standard.

> *The basics of source control are easy to adopt, but strong version management systems (like Git) can scale to organizations of virtually any size with virtually any sort of collaborative workflow.*

However, the problems specific to industrial automation have thus far impeded broad adoption in industrial automation. Industrial file types were designed well before modern version control systems. Oftentimes, common file types like ACD files for Allen-Bradley PLCs are not comprehensible or still entirely in binary (see **Figure 3**).

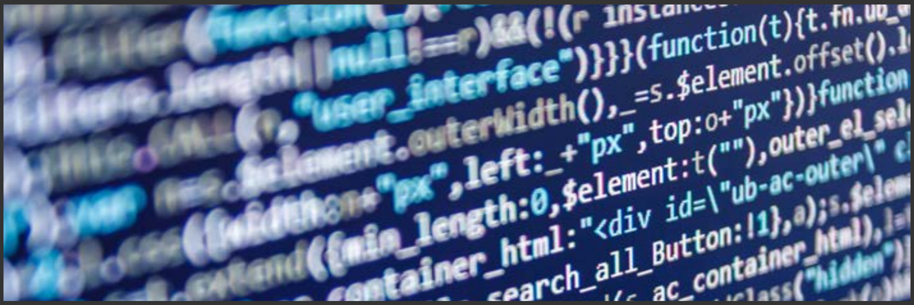


Figure 3
While some systems have moved to XML, these file types are still hard for engineers to parse and understand.

While it’s possible to introduce a Git-based workflow on top of Github or Gitlab, only recently have companies begun to solve this problem for controls professionals. Companies like Copia Automation provide a Git-based workflow built directly to support control engineers with first-order support for IEC 61131 languages and vendor-specific industrial file types. Copia eliminates complexity and hosting issues that traditional industrial automation companies could face, which differentiates our solution from other version control systems on the market.

Copia is the only source control solution built on Git.

> *Other version control tools available today are not specifically designed for industrial or PLC workflows.*

Looking Forward with Copia

As we move into a new decade of industrial automation, there is a growing need to adopt powerful, more responsible source control tools for the foundational systems that power the industrial economy.

Source control systems are easy to adopt, are affordable, and provide immediate benefits to organizations in terms of operational uptime and efficiency, and provide a robust disaster recovery solution for industrial manufacturing.

About Copia

Copia Automation is dedicated to building developer tools to improve efficiency, operational uptime, and agility for organizations.

To learn more visit us at copia.io and feel free to email with questions or comments at contact@copia.io.

COPIA