COPIA

Source Control, Git, and Industrial Automation

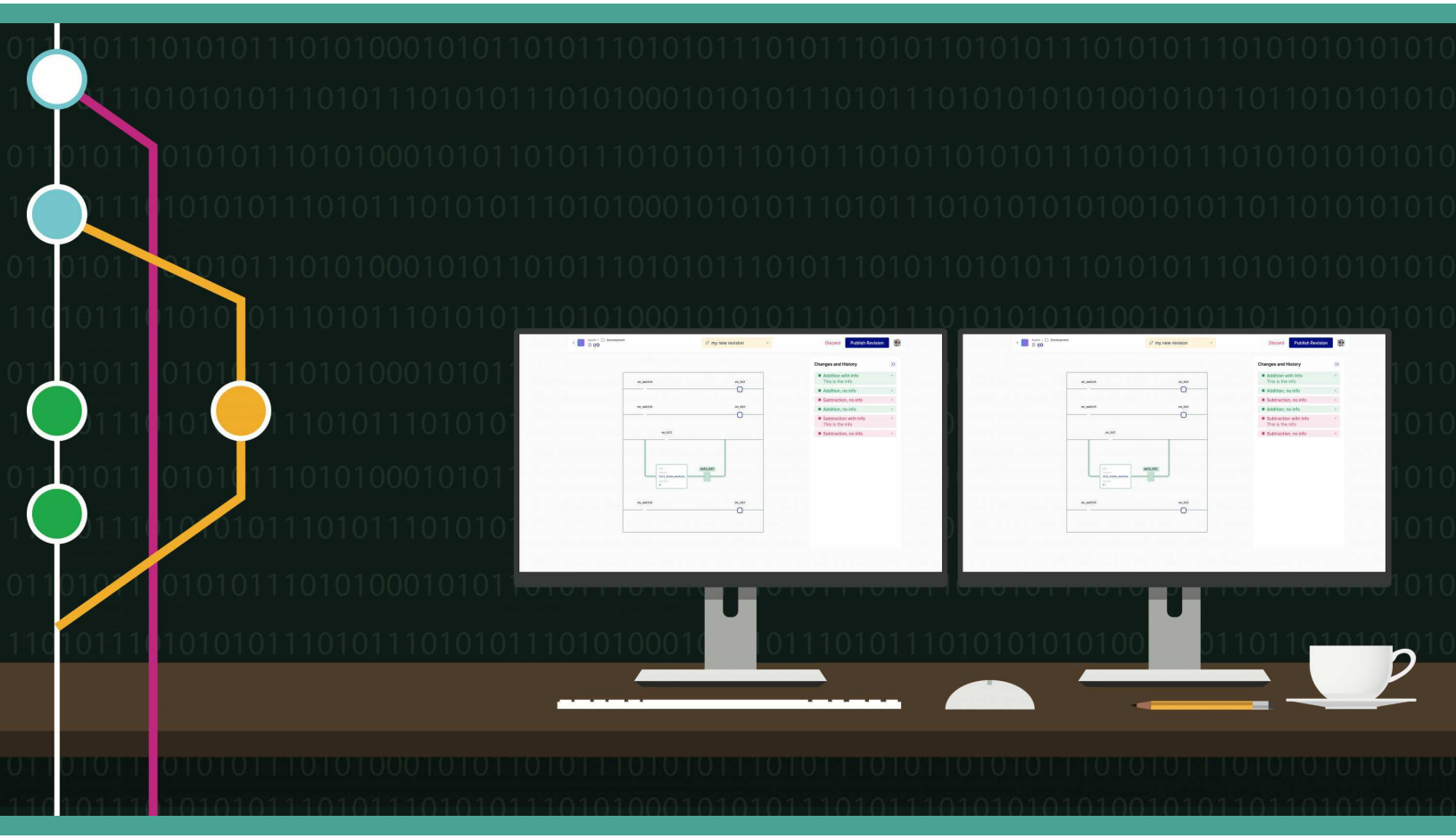


TABLE OF CONTENTS

Overview

...building systems without source control feels like "walking a tightrope without a net."

Source control systems have become ubiquitous in the IT world but have not become a major component of operational technology (in particular, PLC programming).
There is an increasing, popular movement in the industrial controls world to bring modern source control systems to PLC programming. This document will review the benefits of these systems and why people have begun to push for their adoption across many industrial organizations.

We argue here that source control isn't a revolutionary concept, but instead provides multiple tangible benefits to organizations and should be strongly considered given the **relative ease of adoption** and **low implementation cost**. When benefits include **reduced downtime, improvements to engineering efficiency**, and **greatly optimized workflows** these systems easily provide ROI for any organization.

More than that, they ultimately improve the **development experience** for controls engineers and automation experts. Once familiar with these systems, we have heard from engineers that the idea of building systems without source control feels like **"walking a tightrope without a net."**

What is Source Control?

Git - a source control system created by the creator of Linux.

Put simply, a source control system is really just an advanced file management system that enables people and teams to manage the revision history of their documents.

Source control systems tend to feature:

REVISION HISTORY

This enables people to take meaningful snapshots of their files which they can easily retrieve or return to.

VERSION CONTEXT

This enables teams to understand **who** changed **what** and **why**. Often these systems require credentials that enable traceability.

SHARED REPOSITORIES

Centralized storage systems (either in the cloud or on-premise) which allows teams to access relevant files.

ACCESS CONTROLS

Mechanisms to enable access to different files with different permissioning, such as read and write access for different stakeholders.

FILE REVIEW

Mechanisms to enable teams to review changes before they are committed to the shared repository.

...the source code files for the application you are reading this in are more than likely stored in a Git repository.

These are common sense features that build on top of 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 in the last 15-years. In fact, the source code files for the application you are reading this in are more than likely stored in a Git repository.

Source Control Benefits

The adoption of a strong source control system provides numerous benefits to organizations that can be mapped to ROI.

OPERATIONAL UPTIME

Possibly the biggest benefit of strong version control practices is the ability to understand what code changed in the case of unplanned downtime. Investment in version control systems can enable teams to easily recognize who made a change to a system, revert that change, and even understand the context of that change. When the cost of unplanned downtime is on average $250,000 / hour, every minute counts in terms of root causing and mitigating an issue.

ENGINEERING EFFICIENCY

Source control systems unlock new efficiencies between engineers within the same team and organization. For example, we've seen organizations invest in extensive AOI libraries, templated "master" copies of their machine builds, and share reusable components that enable them to quickly kick off new projects. A strong version control practice enables teams to ensure that all of their members are using the correct version of these reusable components and can even provide a centralized system for documentation.

AVOID COPY + PASTE

Good version control systems will integrate directly with a computer's file system so that you do not have to copy and paste files in order to store back-ups. They'll automatically pick up on and store meaningful changes. This means you're less likely to lose a file. One controls engineer told us "my hard drive failed literally the day after adopting a version control system." Fortunately, they were backed up in a central repository!

AUDITING + TRACEABILITY

Increasingly many contracts and regulatory environments require strong version control practices. Organizations that seek SOC 2 compliance have to have a system for code review. Many industrial contracts require suppliers to provide first-order traceability as part of their contract. We've heard of organizations who've considered implementing version control practices for auditing for multi-million dollar contracts.

While high degrees of security and the need for self-hosted solutions can make version control systems expensive to implement for some enterprise manufacturers, most reasonable version control systems cost in the hundreds of dollars a month for a single team to adopt.

Adopting Source Control

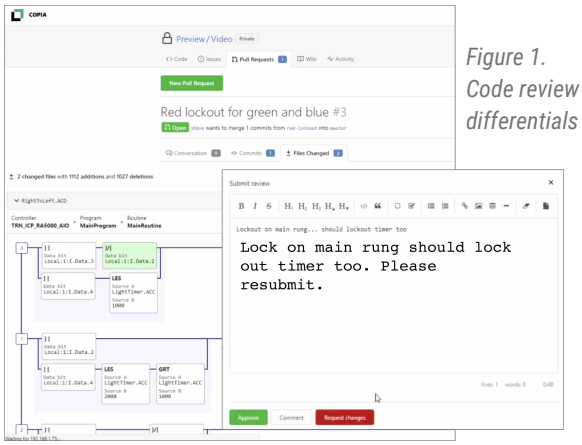


Figure 1. Code review with differentials of routines.

One of the major benefits of source control systems is that they typically integrate at the file system layer. This means that technical adoption is straightforward. We've seen companies fully migrate to a modern source control system in less than an hour.

REGULAR COMMITS

Engineers need to develop a best practice of always committing changes to their source control system before deploying to production.

CODE REVIEW

Engineering teams can begin to develop a code review best practices that creates a record of feedback on code changes and enables deeper understanding of what changed and why certain decisions were made *(see figure 1).*

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 but also allow them to accelerate and improve their collaboration.

WORKFLOWS

Very advanced engineering teams will invest in automated practices around code changes such as internal reporting and even integration with their deployment model.

Overall, the basics of source control are extremely 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.

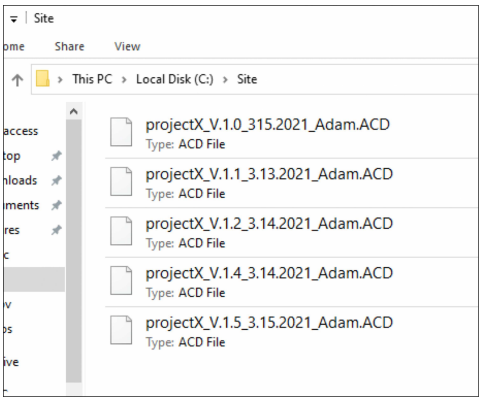
Source Control vs. “The Archive Folder”

In a recent Reddit post on source control systems a commenter made this point:

> *“I don’t get it - this is so you can keep track of versions of your PLC program? At my plant, we have a folder named “archive” for each project. We just name the old project with the date and save it before making edits, etc.”*

A responder gave a “quick and dirty” explanation as to why a true source control system is so much more powerful than an “archive folder.”

1. *Only saves the differences between versions saving space.*
2. *Keeps a history of the changes and who made them*
3. *Allows multiple people to work on different parts of a project at the same time and merge them together without manually going through and splicing in code*
4. *On commits must provide a message so helps keep track of why a change was made.*



In a typical archive folder, there is no context about changes except for file name changes.

When we asked [...] “how did you get people to finally adopt [source control]?” He simply responded, “bad things happening.”

To expand on their excellent answer, we distinguish between a “backup system” and a “source control system.” A backup system provides you with regular backups of a production system. This is useful, but it’s difficult to know what has changed, who has changed it, and when it changed.

With a proper source control system, it becomes easy to trace meaningful changes and thus root cause regressions and **reduce production downtime**. Furthermore, source control systems often provide powerful mechanisms for code sharing and collaboration which **streamline operations** and **save engineers time**.

Ultimately, these sorts of arguments against source control **are not new**. We spoke with a principal software engineer at a major technology company who explained the history of adoption of Git at Microsoft. **When we asked him “how did you get people to finally adopt [source control]?” He simply responded, “bad things happening.”**

Good source control practices save engineers when they need it the most.

Not a Revolution, Just Responsible

Recently an engineer commented in an online discussion about source control, “I have been using my own system to manage files for years, I don’t understand why this is revolutionary.”

We look at this through a different lens. It’s best captured by an embedded systems programmer who over the course of a year convinced their team to adopt source control practices. He eventually convinced his team by telling them that it was **“frankly irresponsible”** to not use version control.

In a world where controls professionals are often responsible for **multi-million dollar production environments**, and where the **cost of downtime** can easily be **hundreds of thousands of dollars** or **even human life**, we believe that organizations simply cannot afford to not adopt systems that will easily allow them to identify **who changed what, where, when, and why.**

Available Tools + Challenges

For many industries, the standard for 15-years 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.

However, Git by itself has some major downsides that make simple adoption by controls engineering orgs difficult:

COMPLEXITY

Git is an extremely powerful tool with a very intense learning curve. For most control system professionals, just learning the command line is a barrier to entry, especially on windows machines which require special plug-ins to work with a tool originally designed for Linux.

HOSTING

To make Git truly work, it’s important to have a remote server that hosts a Git repository. Maintaining and managing a Git server for a standard team is expensive, and often not worth the benefit that it provides.

VISUAL PROGRAMMING

Whereas many programming languages are entirely text-based the IEC 61131 specification provides definitions for multiple visual programming languages. Ladder logic and FBD are nearly ubiquitous in the controls engineering world, and that is likely not going to change any time soon. Certain features in Git, such as showing the “diff” or “difference” between text based files entirely breaks in the context of visual programming paradigms. This makes features such as version history that presents diffs to understand changes less meaningful.

INDUSTRIAL FILES

Industrial file types were designed well before modern version control systems. While some systems have moved to XML, these are still hard for engineers to parse and understand. Furthermore, common file types like ACD files for Allen-Bradley PLCs are still entirely in binary, and are not human comprehensible.

From left to right: GitHub, GitLab, and Copia

The complexity of using and difficulty of hosting Git is a well-known problem, with major Git services such as Github and Gitlab becoming multi-billion dollar companies as they have solved this problem for millions of engineers. However, the problems specific to industrial automation have thus far impeded adoption in industrial automation broadly.

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. In particular, companies like Copia Automation provide a Git-based workflow built directly to support controls engineers with first order support for IEC 61131 languages and vendor-specific industrial file types.

Conclusion

We hope that [...] we begin to see major adoption of powerful source control tools for the foundational systems that power the industrial economy.

Source control systems are easy to adopt, affordable, and provide immediate benefits to organizations in terms of **operational uptime** and **efficiency**. Adoption of source control systems have been curtailed by the lack of support for industrial use cases. We hope that as we move into the new decade of industrial automation, we begin to see major adoption of powerful source control tools for the foundational systems that power the industrial economy.

Copia Automation

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

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