<Industrial Automation>

<Why you need to get it right>

Tibor Lapikas MSc.

Software Improvement Group

Industrial Automation Software: Why you need to get it right

Author

Tibor Lapikas MSc.

Practice Lead Industrial Automation at Software Improvement Group

Contents

Industrial automation: A key success driver in industrial projects	**4**
Measure and improve the technical quality of industrial automation	**9**
Improve development methods and developer education	**13**
Engage suppliers in fact-based collaboration and decision making	**16**
Next steps	**18**
About SIG	**19**

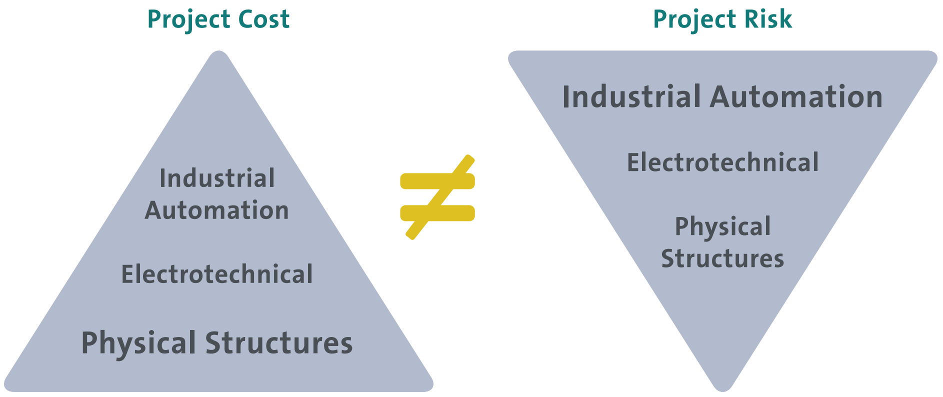
Industrial automation: A key success driver in industrial projects

As demand rises for innovative and efficient software solutions in industrial projects, the need for high-quality industrial automation is also increasing rapidly. However, the current day-to-day practice in the industry shows a very different reality with frequent project delays, cost overruns and inadequate functionality.

Even though industrial automation is increasingly recognized as a vital part to any industrial project, the source code is still often treated as a black box with the actual technical quality of the product never properly known or inspected. Such negligence makes project outcomes uncertain and unpredictable, increasing its risks.

Financial projections: Cost versus Risk

From a financial perspective, industrial automation is a small part of a project's total investment. Only about 5% of the total value of an industrial project is spent on automation. However, when it's neglected, serious project delays and cost overruns are not uncommon. This shows an inverse relationship between the cost of automation and the associated project risks. Therefore, industrial automation, which makes up a small part of an industrial project's investment, is associated with significant risk.



Discrepancy between industrial automation project cost and risk

The key to addressing these risks and leveraging the benefits of industrial automation? Prioritizing the technical quality of industrial automation.

Software quality assurance

Measuring and improving the technical quality of the software in industrial automation projects has significant benefits compared to classic, process- and document-based quality assurance. These advantages include:

- ✓ Risk reduction in delivery time and cost
- ✓ Improved time-to-market
- ✓ Better reuse of existing software
- ✓ Increased innovation strength

These benefits make a strong business case for investing in technical software quality in industrial automation projects. The case is strengthened by the relatively low cost of implementing quality-focused automation compared to the associated risks in the case of its negligence. This e-book explains **the best practices for measuring and improving technical software quality in industrial projects**.

Measuring and improving industrial automation

Measuring the technical quality of industrial automation offers insights into the quality of the software product delivered in an industrial project. This step is crucial for a transparent and predictable delivery of any project. Thanks to these insights, stakeholders have a better understanding of the project scope and progress to mitigate risks at an early stage. The question is, how do we measure the technical quality of automation, and how can we improve it?

Let's dive into 3 main strategies to answer this, highlighting best practices for each to improve the quality of industrial automation:

1. Measure and improve the technical quality of industrial automation

2. Improve development methods and developer education

3. Engage suppliers in fact-based collaboration and decision making

‹Measure and improve›

1. *Measure and improve the technical quality of industrial automation*

Customers and suppliers often evaluate projects in industrial automation from a purely functional point of view. This means they focus on fuctionalities that are mere results of the automation software, or in other words, whether the software does what it is supposed to do. But this approach covers only one aspect of a successful project. How software is designed and built is of equal importance. In comparison, other components of industrial projects, such as hardware and technical installations, are inspected and certified before they are put into use. Why would this be any different for industrial automation?

By measuring the technical quality of the automation, we can identify and mitigate risks early on. These risks include, but are not limited, to project delays, long-term maintenance issues and low productivity.

Measuring software quality starts with the ISO/IEC 25010 standard and its implementation[1]. This international standard for software quality is applicable to industrial automation. The standard defines the core aspects of the software quality that influence the maintainability of software in the long term. The advantage of using an international standard is that programming languages, such as Structured Text and Ladder Logic, can

[1] *www.sig.eu/resources/sig-models/*

be evaluated against a well-defined and standardized framework. It also ensures that technical code quality is analyzed in an objective, repeatable way. Therefore, the ISO/IEC 25010 standard goes far beyond other standards that either focus on documents and processes instead of the technical quality, or they do not assess architectural quality metrics.

Having addressed the technical assessment, it's time to improve the quality of the analysed system. This next step will enhance the maintainability of the system, reducing the time to fix defects and also add new functionalities. In the following pages, we'll walk you through the guidelines to improve the technical quality of the software.

Best practices for improving technical software quality

1. ***Reduce code complexity***
   When you reduce the complexity of the code, your code will be **easier to analyze, modify and test**. Reduced complexity is accomplished by keeping your basic units, i.e. functions, simple and short, and by limiting the amount of logic each function contains.

2. ***Avoid code duplication***
   Copy-paste code is inefficient and error-prone, which frequently results in the need for later adjustments. In addition, duplication also

increases the total source code volume which has a further negative impact on the maintainability of a system. Avoid duplication **by writing reusable, generic code.**

3. ***Reduce time-to-market by generating frequently-used code***
   Developing functionalities for industrial projects often requires laborious and repetitive programming. Therefore, it helps to write generators which automatically create frequently-used code. This **reduces the chance of errors** and makes your codebase smaller and **easier to maintain.**

‹Improve Development›

2. *Improve development methods and developer education*

Modern development practices, such as automated testing, continuous integration and agile project management methods, have become commonplace in office automation. However, these practices can also bring benefits to industrial automation. The market requires high-quality software with shorter time-to-market. Therefore, the industry needs to embrace modern development practices.

Software engineers in industrial automation, who often have an electro-technical background, can benefit from experience and training in software engineering practices. Creating software from a software engineering perspective is different than creating software from an electrotechnical perspective. A software engineering perspective focuses on the abstractions and reusability of the software. Such an approach requires slightly more time up front, but it pays off with improved long-term software maintainability and reduced risk on project delivery.

Best practices for improving development methods and developer education

1. ***Apply modern development practices***
   Automated testing and continuous integration are common development methods in office automation, but they can be equally beneficial to industrial automation. Automated tests enable

immediate feedback on modifications. In addition, automated testing **is scalable** as opposed to manual testing. Furthermore, continuous integration helps to create testable releases more frequently. Combined with advanced integration testing, i.e. against virtual systems, the feedback cycle becomes much shorter. Such an integrated approach will immediately benefit your projects.

2. ***Train engineers in modern software development practices***
   Modern software development practices are not at odds with modern industrial software development. Applying these practices can **enhance the quality** of software, reduce defects and produce **maintainable code in the long run**.

3. ***Develop a software product-line***
   The steps above are easier to implement when developing source code in a product line with your generic product being improved continuously, as opposed to developing single-use source code on a per-project basis. **Thinking from a product as opposed to a project point of view** will serve to identify and develop reusable code. In addition, it will help you focus on the long-term maintainability and functionality of the product. Therefore, developing a product-line can be a significant undertaking, yet it often proves to be a smart investment.

<Collaboration>

3. Engage suppliers in fact-based collaboration and decision making

Software is a fundamental part of any industrial automation project. It is crucial for correct functioning of the hardware, and thus for the success of the whole project. However, measurable software quality requirements are not common in tenders and contracts. Therefore, suppliers can deliver software of unknown quality as long as it functions as intended.

Technical code quality and project success are intrinsically linked. By measuring the technical quality of code during development, you'll uncover risks early on.

By leveraging the results of the technical quality analysis in your collaboration with suppliers, you'll be able to make rational and informed project decisions -- not draw conclusions based on opinions and gut feelings.

Best practices for engaging suppliers in fact-based collaboration and decision making

1. ***Measure technical code quality during development***
   Measuring and reporting on technical code quality will provide **early**

indications as to whether the code meets the highest standards of quality. Therefore, the process of identifying and mitigating risk can start early in the software development cycle. This is one of the key factors to ensure project success.

2. ***Incorporate technical quality requirements into the contract***
   When requirements for technical quality are included in a contract, it is clear to a supplier that technical quality is measured, and **what quality standard is expected**. Furthermore, bids on a project will consider the expected standard of quality, which will have a positive influence on project performance and delivery.

3. ***Measure productivity***
   Oftentimes, productivity is not considered in automation projects. Productivity can be measured by the amount and quality of produced code, and whether the code meets the expected quality standard. Measuring productivity will ensure that both the customer and the supplier can make informed decisions based on **facts**.

Next steps

Putting these guidelines into practice

Interested in learning more about the practical application of software quality in industrial automation? Have a look at one of our case studies that explain the practices of ensuring software quality in industrial automation.

Dunea

At Dunea, a Dutch drinking water company, SIG assessed the software quality of one of their purification plants in collaboration with ABB. Get the full story here:

www.softwareimprovementgroup.com/case-studies/dunea-abb

VanRiet

And SIG supported VanRiet Material Handling Systems, a leading system integrator, in its efforts to generate most of their source code from standardized components and increase delivery efficiency. Learn more here:

www.softwareimprovementgroup.com/case-studies/vanriet

For all further questions, feel free to contact us.

www.softwareimprovementgroup.com/contact

About the Author

Tibor Lapikas is a Senior Consultant at SIG, where he leads the practice for Industrial Control Systems. With more than 10 years of experience in industrial software, Tibor brings new software quality solutions and modern development practices to the industry.

About SIG

Founded in 2000, Software Improvement Group (SIG) helps business and technology leaders drive their organizational objectives by fundamentally improving the health and security of their software applications.

SIG combines its proprietary tools and benchmark data with its consultants' expertise to help organizations measure, evaluate and monitor code quality - whether they're building, buying or operating software.

An independent organization, SIG has the largest benchmark in the industry with more than 10 billion lines of code across hundreds of technologies. SIG expert consultants use the benchmark to evaluate an organization's IT assets on maintainability, scalability, complexity, security and other mission-critical factors. The SIG laboratory is the only one in the world accredited according to ISO/IEC 17025 for software quality analysis.

<Industrial Automation/>

Software
Improvement
Group