Towards Sustainable Industrial Automation Systems

Roland Weiss[α], Heiko Koziolek[α], Johannes Stammel[β], and Zoya Durdik[β]

[α] Industrial Software Systems, ABB Corporate Research,
Wallstadter Str. 59, 68526 Ladenburg, Germany

[β] FZI, Haid-und-Neu-Str. 10-14, 76131 Karlsruhe, Germany

roland.weiss@de.abb.com, heiko.koziolek@de.abb.com, stammel@fzi.de, durdik@fzi.de

Abstract

We identified a set of open issues in the context of software aging and long-living systems with respect to the application domain of industrial automation systems, e.g. process control [7] and SCADA (supervisory control and data acquisition) systems.

Existing systems in the automation domain suffer from expensive evolution and maintenance as well as from long release cycles. One of the root causes for this is that longevity was not considered during their construction.

Most of the solutions that can be found today are not domain-specific, and tend to focus rather on symptoms than on causes. Therefore, we initiated a research project which has the target to define more clearly what domain-specific longevity means, to survey existing approaches, and to derive methods and techniques for addressing the mentioned problem in the industrial automation domain.

In this contribution we present the objectives of this project and outline our state of progress.

1 Introduction

When we compare current research in the context of software aging, long-living systems, maintenance, evolution, and software architecture with our own industrial experience from the automation domain we identify a set of open issues.

In the development and evolution process of automation systems, functional requirements had a main focus whereas longevity has only recently started to become a main citizen.

Starting with a consolidation of state of the art our goal is to synthesize a domain specific method for dealing with software evolution and for providing decision support in project contexts. The approach will be validated in industrial case studies at ABB.

This paper is organized as following: Section 2 describes the application domain, Section 3 defines "long-living" with respect to the industrial automation domain, Section 4 provides an overview of classification and evaluation criteria for the solution strategies, and Section 5 gives conclusions and presents future work.

2 The Automation Domain

In this section we introduce properties specific to industrial automation systems. The increased complexity of today's automation industry is reflected in the increased complexity of control system design and development.

These systems are so-called software-intensive systems, which consist of distributed server nodes, client nodes, and embedded systems (e.g. controllers and field devices). Hence they have to handle a large number of embedded devices of different types which in turn causes problems related to the dependencies of the devices and their evolution due to changing communication and integration standards.

Besides that, industrial automation systems underlie strict requirements on their availability. Thus downtimes have to be minimized, and ideally are aligned with downtimes of machines or plants. A shutdown of these systems is very expensive (and partially dangerous) and software updates should be combined with hardware updates.

Overall such systems have to be constructed following several industrial standards[1] which provide additional requirements and constraints on the architecture and design of the systems.

Process control systems are usually sold only to a limited amount of customers. This means that there is less feedback compared to mass software. Therefore opportunities to learn from mistakes and improve systems are limited.

Compared with other (safety-critical) domains, like military or aerospace, automation systems are very cost sensitive. Therefore, automation systems usually are being developed based on existing ones. This might lead to an "inheritance" of existing problems

[1] E.g. OPC for communication, IEC 61508 for safety, IEC 61850 for electrical substation automation

regarding longevity in new systems.

Automation systems typically have real-time components and have client-server or multi-tier architectures with event-driven communication.

3 Long-living systems and evolution problems

In this section we describe characteristics and special problems of industrial automation systems.

We define systems having a lifecycle of more than 10 years as "long living". The lifecycle ranges from first installation and deployment to final shutdown of the system.

Within this lifespan these systems have to be adapted in order to reflect environmental changes (standards, technologies) or user change requests (bugs, features). The evolution may be caused for example by changes of underlying hardware devices or communication standards. System users expect the system to provide support for more and diverse devices and to cope with increased plant sizes (more devices connected, more control loops executed etc).

Among the common sustainability troubles in automation systems are dependencies to standard information technology, e.g. component systems (COM, .NET) or graphics subsystems. Their evolution cycles are usually shorter than the one of the process control system software.

One has to mention high quality assurance effort involved in system changes. This is usually caused by laws and requirements on safety, development process and / or system environment, and the resulting very high testing effort.

4 Solution Strategies

There are a lot of approaches aiming to solve the software aging problems. We identified four coarse-grained types of strategies.

- Analytical strategies: Approaches for understanding and description of evolution problems using software architecture modelling [8], quality indicators (like metrics [4], bad smells analysis [5]), approaches for analysis of historical data (data mining)
- Structural strategies: Architectural styles and patterns [2], reference architectures [9], variability approaches (software product lines) [1]
- Automation strategies: MDSD approaches [10], generative programming [3], software factory approaches [6]
- Management strategies: Approaches for make or buy decision support, Team organization guidelines

For each category we identified a couple of approaches from literature. One of the challenges is to decide about the applicability and appropriateness of approaches. We have selected several of these approaches based on a criteria catalogue that now have to be customized for efficient application in automation systems.

The criteria catalogue covers properties like applicability, maturity, relevance, positive/negative perspective, degree of formalization, abstraction level, development phase.

5 Conclusion

We have presented our understanding of the software aging problem in the context of automation systems and concluded that until now the sustainability problem as such was not explicitly considered during system development. We have introduced our classification scheme for strategies. Several approaches were briefly mentioned.

We are working on the following next steps: consolidation of strategies in the proposed classification schema; investigation of evaluation criteria and rating for each strategy; selection of the specific strategies for intensive evaluation; customization and recombination of the most promising strategies; evaluation with real world case studies from ABB.

After finishing our research we expect to have a set of methods and application guidelines that enable development and maintenance of sustainable industrial automation systems.

References

[1] J. Bosch. *Design and Use of Software Architectures Adopting and evolving a product-line approach.* Addison-Wesley, 2000.

[2] F. Buschmann. *Pattern-Oriented Software Architecture Volume 1: A System of Patterns.* Wiley, repr. edition, 2001.

[3] K. Czarnecki and U. W. Eisenecker. *Generative Programming.* Addison-Wesley, 2000.

[4] R. Dumke and F. Lehner. *Software-Metriken Entwicklungen, Werkzeuge und Anwendungsverfahren.* Deutscher Unversitäts-Verlag, 2000.

[5] M. Fowler. *Patterns of enterprise application architecture.* Addison-Wesley, 2003.

[6] J. Greenfield and K. Short. *Software Factories Assembling Applications with Patterns, Models, Frameworks, and Tools.* Wiley, 2004.

[7] M. Hollender. *Collaborative Process Automation Systems.* ISA, 2009.

[8] R. Reussner and W. Hasselbring. *Handbuch der Software-Architektur.* dpunkt.verlag, 2. edition, 2009.

[9] J. Siedersleben. *Moderne Softwarearchitektur.* dpunkt.verlag, 2006.

[10] T. Stahl and M. Völter. *Model-Driven Software Development Technology, Engineering, Management.* Wiley, 2006.