Preliminary Evaluation Results of Static Code Analysis of Control Software in an Industrial Context

Juliane Fischer, Eva-Maria Neumann, Jan Wilch, Birgit Vogel-Heuser*
Technical University of Munich
TUM School of Engineering and Design
Institute of Automation and Information Systems
**Core Member of MDSI, Member of MIRMI*
{juliane.fischer; eva-maria.neumann; jan.wilch; vogel-heuser}@tum.de

Martin Obermeier, Thomas Kellhammer
Krones AG, Neutraubling
{Mar.Obermeier; Thomas.Kellhammer}@krones.com

1. Motivation for static code analysis of control software

The control software in automated production systems – primarily implemented on Siemens platforms as the European market leader – must be maintainable for a multitude of machine and plant variants over decades [1]. This can only be realized efficiently with the help of a suitable module structure supporting reuse and maintenance. In reality, however, implementing new requirements often leads to uncontrolled software growth. In many cases, the software is not systematically evolved but is enlarged by extensions in the code to implement new requirements as quickly as possible – the result: historically grown, unmanageable legacy software that is difficult to maintain and reuse.

Since control software has become the decisive factor for a company's success and the differentiation of its products from its competitors, methods that support the revision of existing code by refactoring for improving software quality, including reusability, are required. For software quality assessment and the identification of refactoring potentials, static code analysis has high potential as it enables the examination of software without executing the code [2]. More precisely, it allows checking the program structure, the contained software elements and their dependencies in early development phases. Although the use of static code analysis is not yet state of the art in machine and plant engineering [3], it has already been successfully applied to control software, for example, by [4], [5] and [6]. For the application of static code analysis in an industrial context, an analysis procedure enlarging [7] and a prototypical tool for the analysis of control software developed in Siemens TIA Portal have been developed. The prototypical tool parses source code files exported via TIA Openness into an internal data model for further analysis [8]. First evaluations with industry experts are promising and presented in the scope of this whitepaper.

Acknowledgement: Parts of the results presented in this whitepaper were achieved in the *advacode* project [9] (funded by *Bayerische Staatsministerium für Wirtschaft, Landesentwicklung und Energie* and *Zentrum Digitalisierung Bayern (ZD.B)* under grant number DIK0112/04).The authors from TUM would like to thank *Krones AG* for the active support during the evaluation and the valuable feedback.

2. Evaluation with industry experts

The applicability of static code analysis in an industrial context was evaluated in an online workshop with experts from a German internationally operating plant manufacturing company in the food & beverage and intralogistics sector. Overall, the company has over 16,000 employees worldwide. It mainly uses automation hardware, including programming platforms, from Siemens, Rockwell Automation, and B&R Industrial Automation. Control software projects are programmed in graphical and textual programming languages while considering company-wide programming guidelines and utilizing library modules, code generation and templates. The participating experts were primarily from the company's software development department, but some also had an electrical engineering background. Most of the participants were control software developers, including a few participants from the management level.

The workshop was conducted as a web meeting. After a short welcome, the intended analysis procedure was introduced to the participants as a mixture of presentations and tool demonstrations. The procedure introduction was divided into four blocks (each with 15 min of presentation and 5 min for participants to voluntarily and anonymously answer one or two single-choice question(s)).

- *Block 1*: Overview of the means for static code analysis, including an introduction to the proposed analysis procedure
- *Block 2*: Preparations for conducting a static code analysis using interview guiding questions
- *Block 3*: Conducting the analysis under consideration of company-specific boundary conditions, e.g., checking the conformance of the analyzed software with company-specific programming guidelines
- *Block 4*: Documentation of analysis results as a basis for deriving refactoring recommendations

The four blocks were a mixture of presentations and live demonstrations using the prototypical code analysis tool and the institute's *Self-X Material Flow Demonstrator* as an application example. In total, 46 participants joined the workshop meeting, whereby 36 remained in the web conference for the entire duration.

2.1. Questionnaire-based evaluation

During the workshop, nine single-choice questions were asked via a polling application of the used web meeting tool MS Teams. Workshop questions are referred to as *W#[question number]*, e.g., question 1 is referred to as *W#1*. The originally asked German questions, including their answer options, have been translated to English. The questions and the responses are provided in the following Tables 1 to 9.

Table 1: Answers to W#1: Have you ever used static code analysis methods or tools in the development process in your daily work?

Answer options	**Number of responses**
Yes, means of static code analysis are an inherent part of the development process	2 (5.4%)
Means of static code analysis are used optionally / irregularly in the development process	8 (21.6%)
No, means of static code analysis are not applied	24 (64.9%)
I do not know	3 (8.1%)
Sum (total answers n)	*37*

Table 2: Answers to W#2: Do you find the interview guiding questions helpful for preparing the analysis and identifying the analysis goal?

Answer options	**Number of responses**
Yes, the interview guiding questions are helpful	19 (48.7%)
In part, I find the interview guiding questions helpful	17 (43.6%)
No, the interview guiding questions are not helpful	1 (2.6%)
I do not know	2 (5.1%)
Sum (total answers n)	*39*

Table 3: Answers to W#3: Do you consider the analysis procedure to be successfully applicable in your company with regard to the boundary conditions (such as unchangeable design decisions)?

Answer options	**Number of responses**
Yes, definitely	9 (22.5%)
Partially the procedure can be used	16 (40.0%)
No, the procedure is not applicable	4 (10.0%)
I do not know	11 (27.5%)
Sum (total answers n)	*40*

Table 4: Answers to W#4: Do you think the approach can sufficiently address the constraints of your application sector and would therefore be applicable?

Answer options	**Number of responses**
Yes, definitely	4 (10.5%)
Partially the procedure would be applicable	14 (36.8%)
No, the procedure would not be applicable	3 (7.9%)
I do not know	17 (44.7%)
Sum (total answers n)	*38*

Table 5: Answers to W#5: From your point of view, is the documentation of the analysis results on different levels in the context of your own software helpful to identify anomalies as well as disadvantageous software elements?

Answer options	**Number of responses**
Yes, the documentation is helpful	15 (38.5%)
Partially I think the documentation is helpful	18 (46.2%)
No, the documentation is not helpful	1 (2.6%)
I am uncertain	5 (12.8%)
Sum (total answers n)	*39*

Table 6: Answers to W#6: Does the documentation enable the derivation of recommendations for action and a rough estimate of effort?

Answer options	**Number of responses**
Yes, the documentation of the analysis results enables this	7 (20.0%)
Partly the documentation enables this	21 (60.0%)
No, the documentation does not allow this	4 (11.4%)
I am uncertain	3 (8.6%)
Sum (total answers n)	*35*

Table 7: Answers to W#7: After an introduction to the procedure, would you be able to independently transfer and apply the code analysis procedure to your control software?

Answer options	**Number of responses**
Yes, after an introduction, I could use the approach	7 (18.9%)
Partly I could use the approach after an introduction	11 (29.7%)
I am unsure, as this depends strongly on the scope of training	13 (35.1%)
No, even after an introduction, I could probably not use the approach	5 (13.5%)
I do not know	1 (2.7%)
Sum (total answers n)	*37*

Table 8: Answers to W#8: In principle, could you imagine integrating the described procedure for static code analysis of control software into your company workflow?

Answer options	**Number of responses**
Yes, the entire procedure can be integrated in principle	7 (18.9%)
Yes, parts of the procedure can be integrated	18 (48.6%)
I am unsure if the approach would be integratable	11 (29.7%)
No, the procedure cannot be integrated at all	1 (2.7%)
No answer	0 (0.0%)
Sum (total answers n)	*37*

Table 9: Answers to W#9: From your point of view, is the application of static code analysis easier with the shown approach than without the approach?

Answer options	**Number of responses**
Yes, the application seems easier to me with the procedure	22 (64.7%)
No, the procedure does not make the application easier	4 (11.8%)
I do not know	8 (23.5%)
Sum (total answers n)	*34*

After the presentations and questions, the workshop participants were divided into three groups (Group 1 with 13 participants, Group 2 with 11 participants and Group 3 with 15 participants) to discuss the applicability and usefulness of the presented and demonstrated concepts. Each of the three groups was moderated by a member of the AIS Institute.

2.2. Challenges regarding the applicability in an industrial context

First, the applicability and usefulness of the presented approach were rated in the groups. A summary of all groups (illustrated with different colors) is depicted in Figure 1.

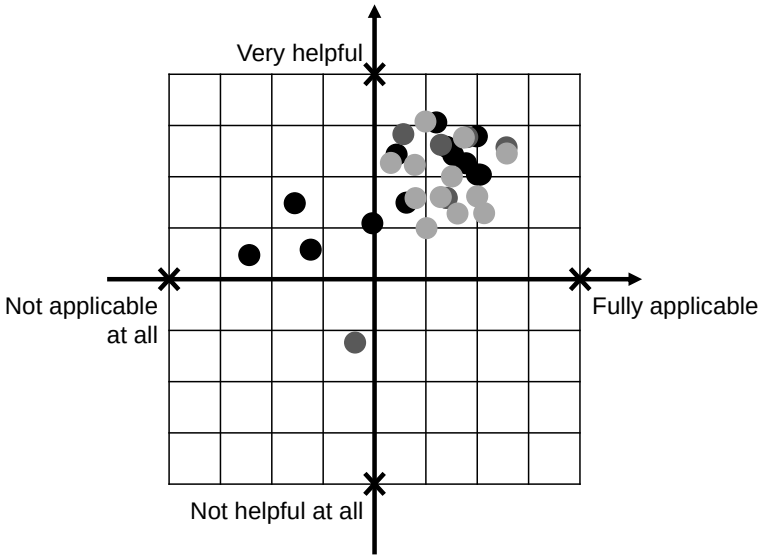

Figure 1: Ratings of the usefulness and applicability of the presented assessment approach (Group 1 = black (12 replies), Group 2 = dark grey (6 replies), Group 3 = light grey (12 replies)).

In the subsequent group discussions, experts indicated challenges hindering the approach's applicability, ways to overcome these and points for supporting the development of high-quality control software. These are listed according to their scope in Table 10.

Table 10: Identified challenges during the discussion sorted by their scope.

Scope	Specifically mentioned challenges
tool-related challenges	• Mix of programming platforms used, including the analysis of the correctness of data exchange between these platforms • Single tool capable of analyzing control software across different platforms • Automated suggestions from an analysis tool are challenging due to the required domain knowledge (including the identification of the highest pain points)
organizational challenges	• Organization of the software development department was seen as an obstacle • Successful application of static code analysis requires assigning a team responsible for the analysis, documentation and subsequent software improvement • Limited capacity in a software developer's day-to-day tasks to resolve the identified weaknesses (appointing resources for refactoring is a strategic question) • Control software developers have little experience with static code analysis • Definition of reusable modules in different disciplines, e.g., mechanics and software development, do not always match since each discipline has different perceptions about module boundaries

Scope	Specifically mentioned challenges
analysis-related challenges	• First trials to analyze software structure in the development environment: Results interpretation, including differentiation between positive and negative software ratings, is challenging due to different stakeholders with different requirements concerning the control software (different stakeholders come to different results) o Suggestion to overcome the challenge: definition of desired software architecture and goals prior to conducting static code analysis (enables setting of evaluation criteria and focal points) • (automatically) analyzing and documenting the software's conformance with company-specific programming guidelines would be beneficial but also challenging since these are partly defined flexibly, leading to different interpretations among software developers o Suggestion to overcome the challenge: differentiating between software parts, which must strictly follow the rules, and less critical parts • The effort to apply the static code analysis to monolithic legacy software is estimated as very high

Despite the raised concerns, according to the industry experts, the integration of the analysis procedure into the company's current development workflow would be possible. Moreover, in the scope of the discussions, various suggestions to support the development of high-quality software and specific application scenarios were mentioned. These are listed in the following.

- The presented prototype and analysis procedure were rated as "very strong" to support software quality management once software architecture and evaluation criteria are defined
- Analyzing, visualizing and documenting the dependencies between control software parts to estimate the required effort and cross effects after a modification (addition, modification or removal of software parts)
- Means to support testing if the software is still functional after a change
- Comparison of a machine's control software versions to detect incorrect use of the template, similar to the application scenario described in [7]
- Inclusion of hardware dependency in the static code analysis
- Assessment of a control software's functional correctness by including a model-based description of the automation hardware as information on the required functionality
- Best practices for initially designing high-quality software are required and could be combined with static analysis (identify used design patterns, represent them abstractly and compare them with the planned structure)
- Performing static code analysis within the used development environment to receive instant feedback while editing the control code would be helpful
- User interface to insert company-specific programming guidelines in the code analysis tool for a subsequent conformance check, which would also enable a cross-departmental comparison

(e.g., between machine types) of guideline conformance (meanwhile, the prototype was extended with company-specific rules for assessing the data exchange, details in [10])

Overall, the sub-group discussions provided insights into challenges and improvement suggestions for including static code analysis in an industrial context.

3. Summary and outlook

This paper presents first evaluation results of a static code analysis procedure and prototype for the quality assessment of control software. In the scope of an online workshop with a German plant manufacturing company, the participating industry experts confirmed the general applicability of the presented approach in an industrial context. Moreover, suggestions to overcome the mentioned challenges and improve the development of high-quality control software have been gathered. These will be targeted in future work.

References

[1] B. Vogel-Heuser, A. Fay, I. Schaefer, and M. Tichy, "Evolution of software in automated production systems: Challenges and research directions," *Journal of Systems and Software (JSS)*, vol. 110, pp. 54–84, 2015.

[2] P. Emanuelsson and U. Nilsson, "A Comparative Study of Industrial Static Analysis Tools," *Electronic Notes in Theoretical Computer Science*, vol. 217, pp. 5–21, 2008.

[3] B. Vogel-Heuser, J. Fischer, and E.-M. Neumann, "Goal-Lever-Indicator-Principle to Derive Recommendations for Improving IEC 61131-3 Control Software," in *2020 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, Singapore, Singapore, 2020, pp. 1131–1136.

[4] H. Prähofer, F. Angerer, R. Ramler, and F. Grillenberger, "Static Code Analysis of IEC 61131-3 Programs: Comprehensive Tool Support and Experiences from Large-Scale Industrial Application," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 1, pp. 37–47, 2017.

[5] S. Stattelmann, S. Biallas, B. Schlich, and S. Kowalewski, "Applying static code analysis on industrial controller code," in *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, Barcelona, Spain, 2014.

[6] J. Fuchs, S. Feldmann, C. Legat, and B. Vogel-Heuser, "Identification of Design Patterns for IEC 61131-3 in Machine and Plant Manufacturing," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 6092–6097, 2014.

[7] J. Fischer, B. Vogel-Heuser, C. Huber, M. Felger, and M. Bengel, "Reuse Assessment of IEC 61131-3 Control Software Modules Using Metrics – An Industrial Case Study," in *2021 IEEE 19th International Conference on Industrial Informatics (INDIN)*, Palma de Mallorca, Spain, 2021, pp. 1–8.

[8] E.-M. Neumann *et al.*, "Refaktorisierung von Steuerungssoftware cyber-physischer Produktionssysteme – Potentiale und Nutzen," in *VDI Berichte 2021*, Baden-Baden, 2021, pp. 679–690.

[9] Institute of Automation and Information Systems, *Advanced systems engineering for control software as a prerequisite for flexible, adaptive cyberphysical production systems (advacode).* [Online]. Available: https://www.mec.ed.tum.de/ais/forschung/aktuelle-forschungsprojekte/advacode/ (accessed: Apr. 28 2022).

[10] J. Fischer, B. Vogel-Heuser, F. Haben, L. Beuggert, and E.-M. Neumann, "Towards Configurable Conformance Checks of PLC Software with Company-specific Guidelines," in *5th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS)*, Coventry, United Kingdom, 2022, accepted.