IEC 61131-3 control applications vs. control applications transformed in IEC 61499

Monika Wenger, Reinhard Hametner, Alois Zoitl *

* *Automation & Control Institute,*
Vienna University of Technology, Austria
(e-mail: {wenger, hametner, zoitl}@acin.tuwien.ac.at)

Abstract: Since distribution of control applications, exchangeability as well as modularization becomes more and more important for the creation of flexible and competitive automation systems we analyze a concept which is supposed to provide these properties. The presented concept is based on a model-driven transformation of IEC 61131-3 projects into IEC 61499. Therefore we transform a small IEC 61131-3 example project into IEC 61499 and compare the simulation results with each other to examine first results on its usability in the automation domain.

Keywords: Transformations, Control Applications, Models, Automation, Simulation

1. INTRODUCTION

Distribution of control applications is becoming more and more important for todays automation systems. Also exchangeability and modularity are considered necessary functionalities within the automation domain. These properties allow the building of more flexible and competitive automation systems since they are supposed to provide the functionality to combine hardware components and software tools of different vendors within one system as well as the reuse of code. Within the industrial automation sector currently the IEC 61131-3 standard (IEC 61131-3, 2003) is used for small as well as very complex systems. This standard provides modularization but through the use of global variables the reuse of code is not given in such an extensive way as it is needed. Equally distribution is only possible in a very static manner through the use of the Access Path and global variables. The insufficient control of the execution sequence is also identified as disadvantage. Under consideration of the mentioned topics, the desired support can be obtained by the IEC 61499 standard (IEC 61499-1, 2005). This standard has been developed as advancement of IEC 61131-3 and offers support for the distribution of control applications. Since its event triggered approach has been designed for distribution and also allows the explicit control of the execution sequence and last but not least is able to rebuild cyclic and procedural approaches, it combines the current functionalities as well as the new properties.
But the necessity to redesign the currently used control software within the IEC 61499 standard is pointed out as great disadvantage since the modality of software design is quite different and can also be very cost and time-intensive. Therefore we developed a model-driven transformation between these standards to make available the desired properties for flexible automation systems while avoiding a redesign of the currently used IEC 61131-3 control software within IEC 61499. Presently the transformation concept has only be tested according to its transformation functionality. Therefore this work aims at the testing of the transformation outcome by the use of simulated devices.
For checking and verifying the model transformation from IEC 61131-3 to IEC 61499 an industrial automation approach for testing control systems which is called soft-commissioning can be used. The basic idea of Soft-Commissioning (SC) is to test industrial control software by connecting a controller (e.g., Programmable Logic Controller) to a discrete event simulator. The greatest advantage of SC is that it can be used and executed on standard office workstations with round trip times smaller than 100ms. A further approach could be Reality in the Loop (RIL) which also allows coupling simulation models to real world objects. Schludermann et al. (2000) and Auinger et al. (1999) show several combinations between reality model and simulation model configurations. Hegny and Zoitl (2010) explained a component-based simulation framework on basis of IEC 61499 which supports several simulation scenarios (e.g., fully simulated plant, integration of external simulation tool, hybrid simulation and fully operational).
Since an example application in IEC 61131 has already been analyzed according to its modes and functionalities we want to verify the transformation by comparing it with the behavior of the transformed IEC 61499 application.
In the following the general transformation concept described in the work of Sünder et al. (2008), and Wenger et al. (2009b,a) are summed up. Section 3 introduces an IEC 61131-3 example which is used to test the transformation concept. It also presents the transformation outcome according to the transformation concept. Finally the IEC 61131-3 source as well as the IEC 61499 target are simulated and the results are compared to each other.

2. TRANSFORMATION CONCEPT

The transformation concept is divided into four parts. The first part describes the transformation process in general. The second and third part illustrate the general

transformation rules whereas two sets of rules are suggested. And the last part contains information about common parts of both transformation sets.

2.1 Transformation Process

The transformation process is predetermined by the chosen transformation framework. We use the Xpand generator framework (originally developed as part of the openArchitectureWare project) which is part of the Eclipse Modeling Project (The Eclipse Foundation, 2010a). The Xpand framework offers textual languages for checks, code generation, and model transformation which can operate on the same models (e.g., XMI or XML) and meta-models (e.g., Ecore or XSD) (The Eclipse Foundation, 2010b).
Figure 1 illustrates the process of the transformation. At the beginning of every transformation a library including a file for each self defined IEC 61131-3 Program Organisation Unit (POU) is built. During this library building part a standard library match is performed since both standard libraries exhibit significant differences which have to be resolved first.
After the building of all FB files (*.fbt) the surrounding structure of the IEC 61499 control application is build and saved in a System file (*.sys). The library and the system part of this transformation process set up the IEC 61499 solution of the provided IEC 61131-3 project. The

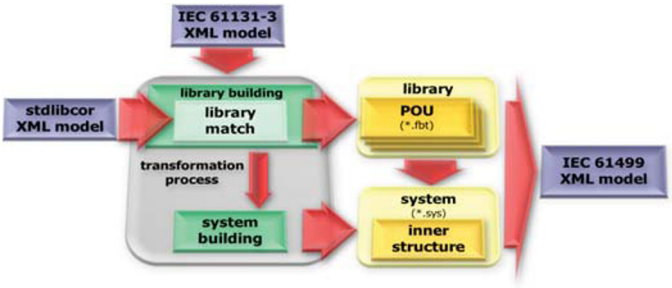

Fig. 1. Model-driven transformation process

library building as well as the system building part of the transformation process are realized by transformation rules. This transformation rules allow to transfer the hierarchical structure of IEC 61131-3 models into IEC 61499 ones. The IEC 61131-3 standard introduces five programming languages Sequential Function Chart (SFC), Ladder Diagram (LD), Function Block Diagram (FBD), Instruction List (IL) and Structured Text (ST). The implemented transformation process currently just supports the transformation of ST and FBD.
Since there are significant differences between IEC 61131-3 and IEC 61499 models it is not possible to just map the elements in an universally valid concept. Currently there are two transformation concepts suggested. On the one hand this is the Name-Driven and on the other hand the Execution-Driven approach.

2.2 Name-Driven Transformation

The Name-Driven transformation concept is based on a name analogy of the IEC 61131-3 and the IEC 61499 standard. Both standards contain a Resource [1] element which is used as starting point for the development of this transformation concept. According to the mapping of a Resource^{61131} to a Resource^{61499} the constraints between the other elements of the two standards can be derived. Therefore every Program Organization Unit (POU) and its corresponding task is transformed to one Application and for every Configuration one Device is created. If there exists a Project element the relevant information is transfered to the System element. Table 1 shows the mapping for this transformation concept based on the analogy of the Resources.

Table 1. Name-Driven Transformation rules

IEC 61131-3	**IEC 61499**
Project	System
Configuration	Device
Resource	Resource
Program & Task	Application

2.3 Execution-Driven Transformation

The Execution-Driven concept is based on the elements responsible for the execution of the control logic. Therefore every Task has to be mapped to one Resource^{61499}. According to the IEC 61499 standard a Resource is responsible for the execution of a specific control part. Each POU is transformed to one Application and each Resource^{61131} to one Device. But having not every POU mapped to a Task those POUs would not be taken into account by the transformation. This is why it is necessary to build one Resource^{61499} for every POU without Task mapping as well. The Configurations are then transformed in an implied manner to the System. Therefore the Configurations become effectively part of the System element. Table 2 shows the mapping for this transformation concept based on the analogy of a Task and a Resource^{61499}.
The use of these concepts depends on the one hand on the

Table 2. Execution-Driven Transformation rules

IEC 61131-3	**IEC 61499**
Configuration	System
Resource	Device
Task	Resource
Program	Application

elements the vendor tool provides and on the other hand the concrete structure of the control application. If the vendor tool just allows one Configuration with one Resource or just one Resource is used in the control application but lots of different Tasks then we suggest to concentrate on the executing elements and therefore use the Execution-Driven concept. On the other hand the Name-Driven concept is easier to understand. And if there are very few Tasks but lots of Resources within the control project then the Name-Driven concept is supposed to be the right choice. Currently there are no tests available which prove the advantage of one single concept.

[1] Resource^{61131} represents an IEC 61131-3 Resource and Resource^{61499} an IEC 61499 Resource