Control Engineering Practice 64 (2017) 173–194

Contents lists available at ScienceDirect

Control Engineering Practice

journal homepage: www.elsevier.com/locate/conengprac

Transformation of GRAFCET to PLC code including hierarchical structures

Robert Julius *, Max Schürenberg, Frank Schumacher, Alexander Fay

Helmut-Schmidt-University, Institute of Automation Technology, Holstenhofweg 85, 22043 Hamburg, Germany

ARTICLE INFO

Article history:
Received 28 September 2016
Received in revised form
23 March 2017
Accepted 25 March 2017

Keywords:
Modelling language
GRAFCET
Hierarchical structure
Control code
Programmable Logic Controller

ABSTRACT

GRAFCET is an advantageous modelling language for the specification of controllers in discrete event systems. It allows for hierarchically structuring a control program's specification based on the elements *enclosing steps, partial-Grafcets* and *forcing orders*. A method is already available for the automatic transformation of *Grafcets*[1] into PLC code but this method cannot keep the hierarchical structures due to limitations of the PLC language SFC. In this contribution a systematic approach to automatically transform *Grafcets* into PLC code while retaining the hierarchical structures is described.

1. Introduction

1.1. Engineering of programmable logic controllers today

The five programming languages Instruction List (IL), Ladder Diagram (LD), Function Block Diagram (FBD), Sequential Function Chart (SFC) and Structured Text (ST) of the international standard IEC 61131-3 (IEC 61131-3, 2013) are today de-facto standard for the engineering of programmable logic controllers (PLCs), which involves the following engineering steps: requirements engineering, specification, implementation, commissioning, operation and maintenance. Both engineers and technicians, e.g. field service personnel and maintenance staff, are familiar with these programming languages since IEC 61131-3 is an integral part of the professional education of control practitioners (Braun, Obermeier, & Vogel-Heuser, 2012; Vogel-Heuser et al., 2013). Hence the IEC 61131-3 standard can be considered state of the art and a substantial achievement of the automation engineering domain (Vyatkin, 2013).

In current practice in the manufacturing domain, where PLCs are mostly applied, the engineering steps "requirements engineering" and "specification" are usually carried out only informally by means of sketches, spreadsheets and text documents. The implementation of the control code is based on the manual interpretation of such informal documents (Frey & Litz). This is mainly due to a lack of time and expertise (Johnson, 2007; Ljungkrantz, Akesson, Yuan, & Fabian, 2012) and often results in additional costs caused by the erroneous interpretation of the textual requirements. The main advantages of formal specifications for control design are the facilitation of automatic code generation, the ability to specify the program behaviour in a formal way for documentation as well as the reusability of the specifications independently from the system under development. The possibility to verify the control code behaviour is another important aspect (Ljungkrantz et al., 2012). Automatic control code generation from formal specifications reduces implementation effort and opens a way for a better acceptance of formal specifications for industrial applications. Therefore formal specifications are an important research issue.

* Corresponding author.
E-mail addresses: robert.julius@hsu-hh.de (R. Julius), alexander.fay@hsu-hh.de (A. Fay).

[1] "GRAFCET" (in capital letters) refers to the standard and the modelling language in general, whereas a "Grafcet" is a particular model, i.e. one control specification.

http://dx.doi.org/10.1016/j.conengprac.2017.03.012
0967-0661/

1.2. Related work

Model-driven development (*MDD*) has been proposed to automatically generate control code from formal design models (Thramboulidis and Frey, 2011; Lukman et al., 2013), but these methods and tools still lack acceptance in industrial practice of PLC programming (Vogel-Heuser et al., 2014). The reason is twofold: on the one hand formal methods proposed in the literature are based on modelling languages which are usually not familiar to the designated users (*control practitioners*) (Vogel-Heuser et al., 2014). On the other hand, most model-driven approaches are not feasible in practice since they only allow performing modifications and revisions within the models to keep consistency of code and models. However, in practice, PLC programmers implement required changes directly in the PLC code. As a consequence, formal approaches are not well accepted in practice (Vogel-Heuser et al., 2014) so far. Below we summarize previous work on modelling and code generation for the specification language GRAFCET defined in the IEC 60848 (IEC 60848, 2013) and offer a brief insight into related research fields.

UML statecharts (OMG, 2015) are an established means of generating code from formal models and facilitate hierarchical structured specifications. A suitable approach generating IEC 61131-3 control code from statechart is presented in Vogel-Heuser, Witsch, and Katzke (2005) and Witsch and Vogel-Heuser (2011) and an introduction of using object-oriented extensions of IEC 61131-3 is given in Witsch and Vogel-Heuser (2009) and Racchetti, Fantuzzi, Tacconi, and Bonfe (2014). Several approaches discuss the verification and model checking of statechart and familiar state machines (Helke & Kammuller, 2016; Missal, Hirsch, & Hanisch, 2007). However, the approach presented in this article does not focus on formal verification but the desirable coherence between specification and control program as well as a readable and maintainable code. GRAFCET is superior to statecharts in that it is rooted in an IEC standard and based on Petri nets (Provost, Roussel, & Faure, 2011b) and thus easily comprehensible by control engineers (Reisig, 2016). It also has recently become an inherent part of the professional education of control technicians (Durey, 1997; Marichal and González, 2014). Hence, GRAFCET is well suited for the specification of PLC programs.

The IEC 61499 (IEC 61499-1, 2012) defines an open architecture for distributed and embedded control with an event-based execution order for program organization units (POUs). The program behaviour for sequence control within the single function blocks is defined by using execution control charts (ECC) (Thramboulidis, 2013) or formal specification languages like Petri nets and will be implemented in an IEC 61131-3 programmable language. Therefore automatic generation of control code from GRAFCET specifications can play a pivotal role in the context of IEC 61499.

According to IEC 60848 GRAFCET is a powerful graphical modelling language for discrete event systems. GRAFCET aims to specify the functional behaviour of sequential parts of a control program. Its hierarchical components (*enclosing step*, *partial-Grafcets* and *forcing orders*) serve to structure a program's behaviour in a clear way and enable control of distributed parts within a single PLC. A hierarchical structure in GRAFCET can be designed by *partial-Grafcets* which are enclosed in other *Grafcets* and by forcing the *situations* of a *Grafcet's steps* from other parts of the *Grafcet* or even from separate *Grafcets*. The modelling features of GRAFCET have been formalized recently as a basis for model-driven development (Schumacher, Schröck, & Fay, 2013.; Schumacher & Fay, 2013; Sogbohossou & Vianou, 2015; Gonzalez, Marichal, & Hamilton, 2016). GRAFCET has been proposed both for code generation (Alvarez, Burgos, Sarachaga, & Estévez, Marcos, 2012) and for conformance test purposes (Provost, Roussel, & Faure, 2011a). A survey on GRAFCET-related research can be found in Schumacher and Fay (2014).

Several GRAFCET design tools are available and in use today, such as *FluidSIM 5*, *SFCEDIT*, *OFT2* and *WinErs*. They allow to graphically designing control specifications, but none of these tools can transform the specification into PLC code in an IEC 61131-3 programming language.

In Schumacher and Fay (2014), an approach has been presented for the automatic generation of IEC 61131-3 compliant PLC code from GRAFCET specifications. This approach is based on an exhaustive formal model of GRAFCET as Control Interpreted Petri nets (Schumacher & Fay, 2014). The target language is Sequential Function Chart (SFC), one of the five PLC programming languages defined in IEC 61131-3. SFC shows many structural similarities with GRAFCET, e.g. steps, transitions, actions, parallel and alternative branches, and therefore seems to be a natural target language for code generation from GRAFCET.

However, the IEC 61131-3 standard does not offer possibilities in SFC to consider hierarchical structures in the way IEC 60848 does in case of GRAFCET. Therefore, in the approach described in Schumacher and Fay (2014), a so-called “normalization” had been applied before the SFC code generation. Devroey et al. (2014) surveys methods for flattening hierarchical structures in state machines. By means of “normalization”, the hierarchical structures of a *Grafcet* are dissolved. The result of the “normalization” is one single *Grafcet* that can be transformed into an equivalent IEC 61131-3 control program in SFC, using unambiguous transformation rules. By “normalization”, the number of *steps* and *transitions* in the *Grafcet* increases, and so does the number of *steps* and *transitions* in the resulting SFC.

But the strength of GRAFCET is the possibility to hierarchically structure a control program. This facilitates the modelling of concurrent behaviour and significantly increases the readability of the specification. Therefore, the fact that the hierarchical structures are removed in the transformation approach described in Schumacher and Fay (2014) is disadvantageous, as it results in relatively large and difficult to read SFC control programs. A more legible hierarchical structured control program enables changes directly in the PLC code during commissioning and maintenance of software. This benefit is not present in a normalized SFC control program.

The authors' main goal was therefore to develop a transformation approach that preserves the hierarchical structures specified in GRAFCET during the generation of the IEC 61131-3 control code. This approach is described in this manuscript. To encourage the practical use and acceptance of the approach, the concept of a software tool is presented which offers the basic infrastructure for an automatic generation of control code, according to the IEC 61131-3 standard, even for hierarchical *Grafcets*.

1.3. Outline of the article

Section 2 discusses the selection of a target language and a target POU for an automatic transformation approach. To our best knowledge this is the first closer look at a suitable IEC 61131-3 target language for GRAFCET preserving hierarchical structures. Subsequently, in Section 3, a formal model of GRAFCET is defined that summarizes previous work in the field. This forms the foundation for a GRAFCET transformation algorithm for sequential systems. Section 4 presents the complied transformation algorithm which transforms *Grafcets* to PLC code in the IEC 61131-3 language ST including method-oriented elements. In this section a new set of transformation rules