OVERVIEW OF DISTRIBUTED CONTROL SYSTEMS FORMALISMS

P. Holečko

Department of Control and Information Systems, Faculty of Electrical Engineering, University of Žilina
Univerzitná 8216/1, SK 010 26, Žilina, Slovak republic, tel.: +421 41 513 3343, e-mail: holecko@fel.uniza.sk

Summary This paper discusses a chosen set of mainly object-oriented formal and semiformal methods, methodics, environments and tools for specification, analysis, modeling, simulation, verification, development and synthesis of distributed control systems (DCS).

1. INTRODUCTION

Increasing demands on technical parameters, reliability, effectivity, safety and other characteristics of industrial control systems initiate distribution of its control components across the plant. The complexity requires involving of formal methods in the process of specification, analysis, modeling, simulation, verification, development, and in the optimal case in synthesis of such systems.

2. DISTRIBUTED CONTROL SYSTEM

A common general definition of a distributed system describes it as a system consisting of several intelligent devices cooperating for common purpose. Intelligent devices (microcomputers, workstations, robots etc.) support processes, which coordinate activities and information exchange via a communication network. In order to call a device "intelligent", it must fulfill following requirements (Fig. 1):

- Contain some kind of processor or CPU for processes realization and decision making;
- Dispose sufficient memory capacity for information storage.

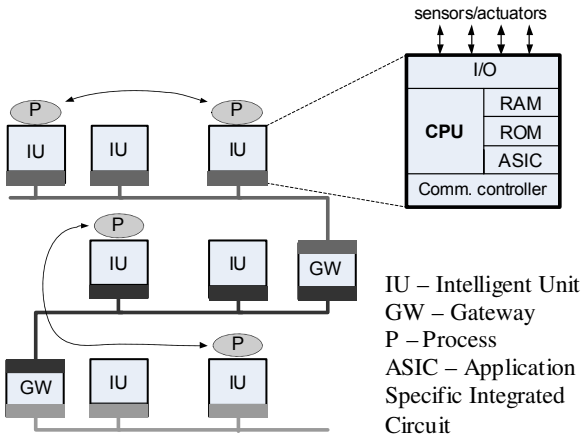

Fig. 1 Example of distributed control system and intelligent device structure

The term Distributed Control System (DCS) denotes a control system, usually of a manufacturing system, process or other type of dynamic system, in which the control elements are distributed and interconnected by a network for the purpose of communication and monitoring.

In the next section the problem of formalizing the processes of DCS's life-cycle will be discussed.

3. FORMAL METHODS

The main motivations of using formal concepts are [9]:

- In the process of formalizing informal requirements, ambiguities, omissions and contradictions will often be discovered;
- The formal model may lead to hierarchical semi-automated (or even automated) system development methods;
- The formal model can be verified for correctness by mathematical methods;
- A formally verified subsystem can be incorporated into a larger system with greater confidence that it behaves as specified;
- Different designs can be evaluated and compared.

In general, formal methods for distributed control systems must address the following problems:

I. Modeling - Select appropriate models and formal notations for adequately describing controlled and control system. These notations must deal with the dynamic and reactive nature of the controlled system, and allow for the proper expression of timing properties.

II. Verification - The verifier is presented with a formal mathematical model of the system, and a specification *S* of how the controlled system should behave. The verification problem involves demonstrating that the model of the system satisfies the specification *S*.

III. Development - In controller development a specification *S* is given that the plant must satisfy (the controller is not given). A disciplined method is sought whereby designers can be helped to construct a controller so that system model satisfies *S*. In development the controller should be built in a modularly structured compositional fashion (controller architecture)

IV. Synthesis – If controller development is fully automated, then such process is called synthesis.

A chosen set of methodics, environments and tools fully or partially satisfying the specified

requirements related to formalisms will be introduced.

3.1 Sequential Function Chart

The primary objective of Sequential Function Chart (SFC++) is the implementation of graphical modeling formalism for design, validation, simulation and automatic code generation of industrial systems, graphical language for programming its control software and supervisory level control tool for control, monitoring, diagnostics etc. [10]. The aim of the running project is also creation of visual programming environment, which integrates the advantages of object-oriented modeling for design and simulation and the performance of distributed control systems (i.e. computers with real-time operating systems interconnected via industrial networks). To bypass the differences between object-oriented model and implementation level, on which run several parallel tasks, a standard formalism (IEC, 1988; UTE, 1992) is used for describing of system dynamics and programming of control system based on Sequential Function Chart (SFC).

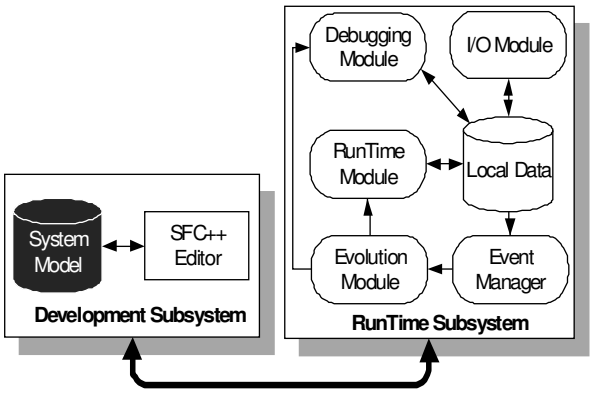

Fig. 2 SFC++ logical architecture

SFC++ is intended for control systems using single or multiprocessor computers or PLCs with real-time operating systems ((RTOS) interconnected by local networks and with control buses connecting the process and other devices.

3.2 The Crisys project

The Crisys project aims at improving, unification and formalization of the actual methods, techniques and tools used in the industries concerned with process control, in order to support a global system approach when developing DCS [2],[3]. The main result of Crisys project is quasi-synchronous approach based on synchronous language Lustre [7] and associated tool SCADE (Safety Critical Application Development Environment) [1]. This approach is dedicated to a special class of DCS, which are organized as several periodic processes, with nearly the same working period, but without common clock, and which communicate by means of shared memory through serial links or field busses.

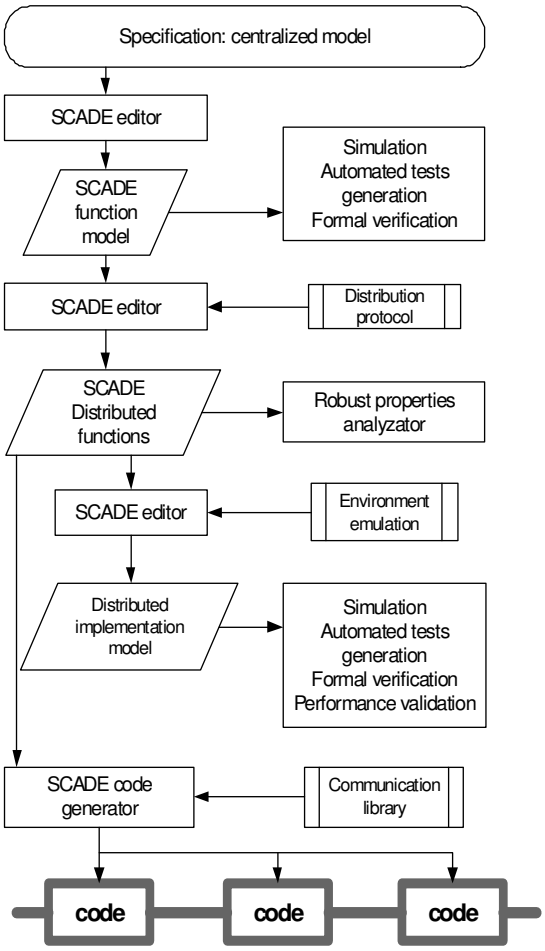

Fig. 3 CRISYS methodology scheme

3.3 The ISILEIT project

The ISILEIT (Integrative Specification of Distributed Control Systems for the Flexible Automated Manufacturing) project aims at the development of a seamless methodology for the integrated design, analysis and validation of distributed production control systems [6]. Its focus is the use of existing techniques which should be improved with respect to formal analysis, simulation and automatic code generation. The integration of SDL block diagrams, UML statecharts and collaboration diagrams formed an executable specification language that allows specifying reactive behavior as well as complex application specific object structures. To ensure the correctness of the design at the earliest stage, validation in form of simulation and formal verification is integrated

into the process. The developed simulation environment can be used to prove that the generated executable code meets the requirements.

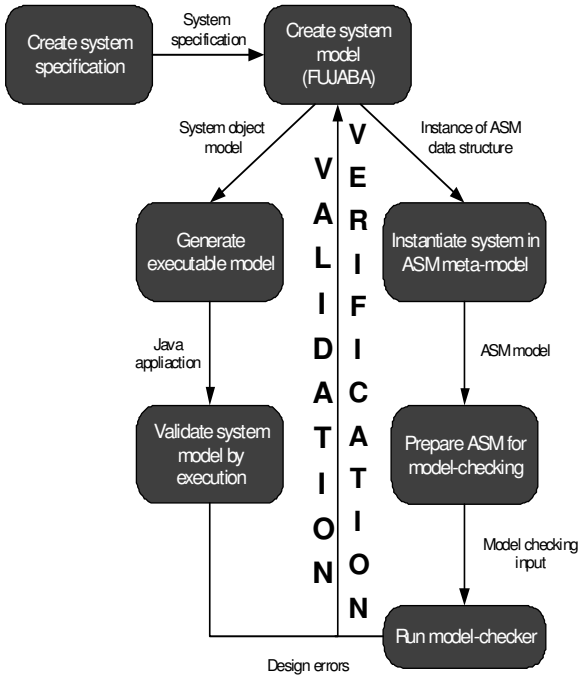

Fig. 4 ISILEIT methodology evaluation activities

3.4 DCS Modeler, DCS Simulator

During design of object-oriented simulation model of a DCS several requirements were considered [4]:

a) Dynamic behavior of each component device within DCS can be considered to be a Finite State Machine (FSM);
b) Overall state of simulation model of a DCS is changing and is determined through communication events among component device models;
c) Component device models of simulation model of a DCS should correspond with physical sensors and actuators in order to simplify the construction of DCS simulation model;
d) Device model consists of diverse combinations of device component models;
e) Events among component device models are transmitted at a level within device or at a level among devices.

The DCS simulation model should be combined with network model in order to evaluate the event communication at the inter-device level. To solve these requirements principles using Design Patterns are widely used. This approach specifies reusable mechanisms for cooperation and interaction between classes or between objects for the purpose of solving common object-oriented problems in general domain.

3.5 Architecture Description Language

The Architecture Description Language (ADL) is defined as a language which disposes of capabilities for modeling conceptual architecture of both hardware and software systems [8]. The language provides models, notations and tools for description of components and its interactions, particularly with regard to large-scale high-level designs. It supports the selection of principles, application of architecture paradigms, abstraction and designs implementation.

The main properties of the language include explicit specification of:

- components,
- connectors,
- interfaces,
- configurations.

A component represents a computation unit or data store and forms loci of computation and state. A connector is a construction block used for modeling of interactions among components and for modeling or rules, which govern those interactions. The interfaces ensure correct connectivity and communication of components. Architectural configuration or topology is connected graph of components and connectors which describes architectural structure.

3.6 IEC 614 99

The IEC 61499 standard modified the Function Block (FB) concept of the IEC 61131-3 standard taking into account the FB concept in field-bus standardization IEC 61804 [5]. Thus the elementary model of IEC 61499 is a function block, which forms the basic structural block of the entire application. There are two types of function blocks: basic function blocks and composite function blocks. Composite function blocks contain other composite blocks and/or basic function blocks. Basic function block contains algorithm and an Execution Control Chart (ECC). Even though the IEC 61499 has some similarities with its predecessor IEC 61131 regarding structural hierarchy and atomic structural construct, function blocks concept, it established a special different concept. Primarily the standard introduced an event-driven approach of interaction among function blocks, whereas existing standards and languages use data or signal communication among elements with assumption of cyclic execution. The standard is defined as a generic standard hence not limiting user to apply of a specific implementation language, communication protocol or hardware components. This enables

generation of heterogeneous networks of distributed control applications.

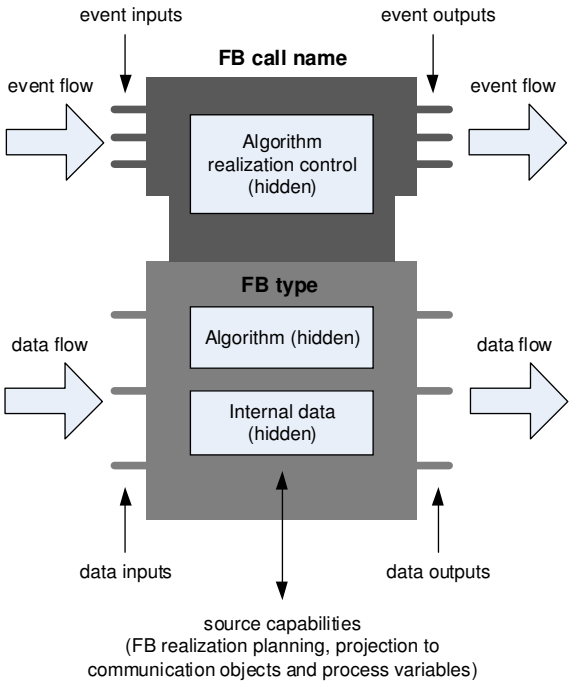

Fig. 5 Function Block (FB) model according to IEC 61499-1 standard

The IEC 61499 standard defines open architecture for design, development, simulation, testing and implementation of distributed control and automation systems.

4. CONCLUSION

The introduced selected set of formal and semiformal methodics, methods, environments and tools is designed to be involved in the process of specification, design, analysis, modeling, simulation, verification, development and synthesis of distributed control systems used in industrial environments. The formalization of these activities enables exact formulation and testing of diverse system requirements, parameters, functionalities, structures etc. The formal outputs of these processes may be subject of appraisal of supervisory authority to guarantee the correctness of the entire procedure and meeting the specified requirements.

Acknowledgement

The work has been supported by KEGA project Nr. K-057-06-00: Innovation of laboratory education methodics on the basis of modelling and simulation in Matlab program environment in combination with educational models using e-learning.

REFERENCES

[1] Bergerand, J.L., Pilaud, E.: *SAGA: A Software Development Environment for Dependability in Automatic Control*. Safecomp'88, Pergamon Press, 1988.

[2] Caspi, P., Curic, A., Maignan, A., Sofronis, C., Tripakis, S., Niebert, P.: *From Simulink to SCADE/Lustre to TTA: a Layered Approach for Distributed Embedded Applications.* Proceedings of the 2003 ACM SIGPLAN conference on Language, compiler, and tool for embedded systems, pp. 153-162, San Diego, California, USA, ISSN 0362-1340, 2003.

[3] Caspi, P., Mazuet, C., Paligot, N.R.: *About the Design of Distributed Control Systems: The Quasi-Synchronous Approach*. SAFECOMP 2001, LNCS 2187, pp. 215-226, 2001.

[4] Tomura, T., Uehiro, K., Kanai, S., Yamamoto, S.: *Developing Simulation Models of Open Distributed Control System by Using Object-Oriented Structural and Behavioral Patterns.* Proceedings of the Fourth International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'01), 2001.

[5] Frey, G., Hussain, T.: *Modeling Techniques for Distributed Control Systems based on the IEC 61499 - Current Approaches and Open Problems*. Proceedings of the 8th International Workshop on Discrete Event Systems (WODES 2006), Ann Arbor, Michigan, USA, pp. 176-181, 2006.

[6] Giese, H., Kardos, M., Nickel, U.: *Integrating Verification in a Design Process for Distributed Production Control Systems*. Proc. of Second International Workshop on Integration of Specification Techniques for Applications in Engineering (INT2002), Grenoble, France, 2002.

[7] Halbwachs, N., Caspi, P., Raymond, P., Pilaud, D.: *The synchronous dataflow programming language Lustre*. Proceedings of the IEEE, pp. 1305-1320, ISSN 0018-9219, 1991.

[8] Medvidovic, N., Colbert, E.: *Architecture Description Languages*. Center for Software Engineering, 2003.

[9] Ostroff J.S.: *Formal Methods for the Specification and Design of Real-Time Safety Critical Systems*. Journal of Systems and Software, Vol. 18, No. 1, pp 33-60, April 1992.

[10] Pardo, X.C., Ferreiro, R., Vidal, J.: *SFC++: A Tool for Developing Distributed Real Time Control Software*. Advanced technologies in manufacturing, pp. 197-202, ISBN 84-95138-08-5, 1998.

[11] Zeigler, B., Praehofer, H., Kim, T.G.: *Theory of Modeling and Simulation*, Second Edition. Academic Press, USA, ISBN 0-12-778455-1, 2000.