Software Engineering for Control Technology - Definitions and Requirements

Storr, A.

Prof. Dr.-Ing.

Institut für Steuerungstechnik der Werkzeugmaschinen und

Fertigungseinrichtungen, Seidenstr. 36, 70174 Stuttgart/Germany

Tel: 0711/121-2420 Fax: 0711/121-2408

Abstract

Software for control technology is a significant part of a machine. The necessity of applying systematically procedures and methods for the development of control software is emphasised. The use of CASE tools and the specification of reusable software modules can increase the productivity. When selecting the methods and procedures, various constraints have to be considered. The selection is influenced by the hierarchical level of the control architecture, the kind of control function that is to be implemented and the required quality features. An important basic design method is the decomposition of control software according to the decomposition of hardware features in mechanical engineering. This will lead to an object-oriented structure. Additionally, using the description method of the state graph, the reuse of control software is supported.

Keywords

Structures and demands of control technology; software engineering procedures and methods; CASE (computer-aided software engineering) tools

1 INTRODUCTION

Nowadays, the term 'software crisis' is often heard. What is generally implied is that there is a shortage of software engineers. The outcome is that engineer-like systematical procedures and methods are often not applied to software development and especially not to software design. The significance of software as a time and cost factor as well as a product and manufacturing component and consequently as machine element has not yet been completely recognized.

1. Software development is differentiated from the software application cycle. Application includes also maintenance and service i. e. further development. Experience has shown that the expenditure concerning the maintenance can be higher than that concerning the software engineering.
2. The engineering cycle consists of different actions. The sequence of these actions reflects a systematical process or procedure. It is important that much time is dedicated to the "early" actions and that they are characterized by thorough and extensive analyses.
3. Furthermore, the figure illustrates that errors often occur relatively early in the process but are usually discovered late. This results in cost-intensive corrections.

Fig. 1 shows some features of software development.

Fig. 1: Phases of software engineering, time and cost (according to Hering and others)
(104 348)

The figure does not explicitly demonstrate the necessity of re-usable software and the resulting advantages as, for example, cost reduction and a reduced number of errors. Furthermore, systematical engineer-like software development requires methods e.g. for design modelling, in addition to the above mentioned process. CASE (Computer Aided Software Engineering) tools - are based on procedures and methods (fig. 2). The development and the application of function-specific CASE tools is required urgently.

Fig. 2: Components of software engineering (104 258)

2 DEFINITIONS

2.1 Software Engineering - Software Technology

Presently both terms are used. Software engineering is according to IEEE "the systematic approach to the development, operation, maintenance and retirement of software". According to Fig. 3, software engineering is principle-oriented, whereas software technology is understood as application-oriented software engineering. Here it is clearly seen that software technology is, to a high degree, an engineering discipline and serves for the systematical production of software. In this paper, software design in regard to the application "control technology" stands in the forefront.

Fig. 3: Comparison software technology and software engineering (acc. to Göhner)(104 323)

2.2 Activities and Methods

Methods can support the execution of single or multiple tasks. When supporting multiple tasks, a method is comprehensive and continuous. Fig. 4 compares methods according to the mentioned differentiation. Furthermore, function-, data- and object-oriented methods are distinguished. It is important that a method is supported by a graphically describing or representation function. In Fig. 5 additional methods and such functions are listed. Both figures show clearly the problems with the selection of methods, especially since these do not cover the broad range of requirements. They must be assessed by their performance in regard to the application area - in our case control technology. A single method is not sufficient, but, for example, a data-oriented and a function-oriented method have to be combined.

2.3 Decomposition

Decomposition - whether it be top down or bottom up - leads to modularity and hierarchy of systems. It is a basic design method and has for a long time been applied to hardware features in mechanical engineering. The objective is the definition of control functions in regard to this decomposition. This results in a communication basis for various departments of an enterprise, e.g. for mechanical and electrical design and software development. Sometimes the term machining objects is introduced. They represent an integration of a mechanical function unit and software i. e. control functions. Re-usability is the goal and thus configurability of software becomes more and more important.

Fig. 6: Function-oriented structuring of a machine tool (301 104)

3 DEMANDS ON SOFTWARE DEVELOPMENT

A distinction has to be made between the application-related demands in the field "control technology" and general demands. Control technological tasks are split up in levels, according to functional aspects (Fig. 7). This does not need to correspond with the decomposition of devices; here different structures are possible.

This workshop attends to the levels E3...E5 and also to functions for the user interface. This is shown in the figure as MMI (Man Machine Interface) in several levels. Systematically generated user interfaces are based on the Seeheim model. This level structure model distinguishes between presentation, interactive control, and an application model preparing the data. The control functions of level E3...E5 contain different criteria (Fig. 8) which result in different, graphic-oriented description or modelling methods. Unified modelling causes problems. The figure illustrates the design bases related to levels and the used representation

forms. Modeling or describing the control functions may be function-, data- or object-oriented, further it can be directed at static or dynamic behavior.

Fig. 7: Hierarchical functional control architecture (501 40/41)

Fig. 8 indicates, just like Fig. 5, that there is a problem in selecting and combining suitable methods. However, it should not be denied that different methods may contain similar elements, e.g. state transition diagrams coming from finite automata. This has to be taken into consideration when the selection takes place.

Control functions/data	Description methods	Design bases	Graphically supported representations
control functions	master control/cell level (level E5 and E4)	-state/place changes -if - then - relations -mathematical relations	-state/transition diagram (Z.B. PN, SG, Booch-comp.) -decision tables
	machine control level (level E3, e.g. NC)	-mathematical relations (trigonometry, DG) -closed loop control models	-block diagrams
	machine control level (level E3, e.g. PLC)	-logic relations (Boole) -state changes -sequential operations	-ladder diagram, control system function diag. -state/transition diagram -sequential function charts
data	control levels E3 ... E5	-data structures	-Entity-relationship-model (ERIM)

© ISM 196

DG ... differential equation SG ... state graphs PN ... Petri's networks

Fig. 8: Description methods for control function design (410 124)

Fig. 9: Empirical analysis of the attempts made to solve the software crisis (according to Matsumoto) (104 217)

General requirements - resulting from the already mentioned problems - can be summarized in three focal points (Fig. 9):

1. Improvement and application of methods which result in so-called CASE tools;
2. Improved functional structuring of systems that is also called decomposition. In the field "control technology" this can be corresponding to the physical view at a machine and
3. Reusability of software, for example, by means of adaptability and configurability.

In fig. 10 the reusability of software is also a quality feature of software. Altogether, systematical procedures, methods and CASE tools should help to develop high-quality software. This results in a systematic software design, for which some simple examples will be presented in the following chapter.

Fig. 10: Quality features of software (301 51)

4 CONCEPTS FOR SYSTEMATIC DESIGN

Systematic design has to comply with the generation, the re-usability of software as well as with an interdisciplinary comprehensibility of the design basis. Using the example of a feeding device (fig. 11) as can be found in a transfer line or in a transportation unit, the modelling with state graphs will be demonstrated. The decomposition is illustrated by this figure.

Fig. 11: Hardware decomposition of a feeding device (104 325)

The following picture illustrates the synchronisation of process sequences and function units by means of orders and acknowledgements.

Fig. 12: Synchronisation of function group and function units (104 327)

After defining the entire description, the automated generation of the code may be done by a compiler. One example is the C code that is processed by modern controllers. Continuance up to the code is hereby achieved. Concerning the generation of the code for PLC, the conversion into the programming language of DIN IEC 1131-3 is most apt. State graphs may, as the examples proved, be called object-based because function units or sequences represent an object. Machine objects with the feature "reusability" can thus be defined.

Fig. 13: Software structure of conventional and decentralized structured object-oriented production management systems (104 331)

A second example refers to the master control (fig. 7 and level E4...E5). Contemporary master controls consist of function-oriented software modules (fig. 13). The migration from a function-oriented to an object-oriented structure can be seen, with the physical object view becoming evident. The reusability and systematic design are improved considerably. Object-oriented modeling e. g. according to Booch, and programming support the systematic design by means of objects as a basis. Elements of the Booch notation are state transition diagrams related to state graphs.

Hereby the close relationship of methods for modelling various control functions of different levels becomes evident. The same applies to functions for the user interface.

5 SUMMARY

Concepts for computer-aided systematic design of control software and the reasons why they are so necessary were presented. Initial steps for the implementation of CASE tools were already taken in close cooperation of industry and university. The joint research work is to be continued in order to computer-aided design and generation of software by means of engineering. The demands on the different control levels must be considered as well as the computer-aided configuration of reusable modules (with classification features) and complete control systems. Furthermore, it has to be taken into account that software and information systems should contain more and more supporting elements, for example, in the field of diagnosis and if manufacturing alternatives are applicable. In addition they should be able to absorb experience and know-how of the user and to make it available if needed.

Organisational measurements for the planning of time, cost and personell with software development were not taken into account. These are necessary on top of the technical requirements mntioned in this paper, in order to execute an software development effectively.

6 REFERENCES

- Ludewig, J. Software Engineering und CASE-Begriffserklärungen und Standortbestimmung. it 33 (1191) H. 3., S. 112 ...120.
- Davis, A. M. Software Requirements. New Jersey: Prentice Hall 1990.
- DeMarco, T. Structured Analysis and System Specification. New York: Yourdon Press 1978.
- Ward, P.T.
Mellor, S.J. Structured Development for Real-Time Systems. New York: Yourdon Press 1985.
- Chen, P.P. The Entity-Relationship Method - Towards a Unified View of Data. ACM Transactions on Database Systems, Vol. 1, No. 1 (March 1976), S. 9 ... 36.
- Booch, G. Booch Method of Object Analysis and Design. Santa Clara (CA): Rational 1992.
- Rumbaugh, J. Object-Oriented Modeling and Design. Englewood Cliffs, New Jersey: Prentice Hall 1991.
- Pritschow, G. u.a. Studie über die Auswirkung von einheitlichen Entwurfs- und Entwicklungswerkzeugen zur Softwareentwicklung und durchgängigen Softwaredokumentation für eine Fertigungszelle. Frankfurt: VDW-Forsch.ber. 1012, 1991.
- Pritschow, G. u.a. Pflichtenheft und Bewertung von CASE-Tools zur durchgängigen Software-Erstellung und Software-Doku-mentation. Frankfurt: VDW-Forsch.ber. 1013, 1993.
- Storr, A. u.a. Simultan zur SPS-Software - Neue Ansätze zur effizienten SPS-Programmierung. ELEKTRONIK 43 (1994) 23, S. 124 ... 136.
- Fleckenstein, J. Zustandsgraphen für SPS - Grafikunterstützte Programmierung und steuerungsunabhängige Darstellung. ISW 63. Berlin: Springer 1987.

- Brantner, K. Adaptierbares Leitsteuerungssystem für flexible Produktionssysteme. ISW 96. Berlin: Springer 1993.
- Siewert, U. Systematische Erstellung adaptierbarer Leitsteuerungssoftware am Beispiel der Durchsetzungsplanung. ISW 100. Berlin: Springer 1994.
- Pritschow, G. Offene Systeme für die Automatisierung in der Produktion.
- Siewert, U. GMA (VDI/VDE-Gesellschaft Meß- und Automatisierungstechnik) - Kongreß'93 Automatisierungstechnik. Dresden, 20.9.1993.
- Storr, A. Objektorientierte Leittechnik: neue Perspektiven und Lösungen.
- Uhl, J. CIM-Management 11 (1995) 1, S. 30 ... 34.

7 BIOGRAPHY

Prof. Storr is the deputy director of the Institute of Control Technology for Machine Tools and Manufacturing Units at the University of Stuttgart. His main fields in research and development work are master control systems, information systems and NC programming systems, furthermore the automation of the technical information flow. These fields are accompanied by systematic software engineering.