ON THE SPECIFICATION OF DISTRIBUTED COMPUTER CONTROL SYSTEMS

J. Ludewig and H. Sandmayr

Brown Boveri Research Center, CH-5405 Baden, Switzerland

Abstract. The computer science group at the Brown Boveri Research Center, Baden, Switzerland, is aiming at a technique for all levels of process control system development. Specification systems are a rather new, but promising field for improvement. The paper describes the current situation, and the tools which are now available. These existing tools, which are powerful, but limited in range, will be combined and extended to obtain an integrated environment for the development of process control systems.

Keywords. Distributed systems. System specification, Levels of abstraction, Software Tools.

INTRODUCTION

After countless discussions about system (or software-) specifications, we still do not have a generally accepted definition of the term "specification". So, everybody who intends to talk on this topic must first provide his personal definition, in order to be as clear as possible.

The Technical Committee on Application Oriented Specification of EWICS has agreed in the following definition (Kramer, 1981):

> A specification is a description of an object stating its properties of interest. It usually implies that the description should try to be precise, testable and formal. It is recommended that 'specification' be used with some attribute, e.g. requirement specification.

The objects which are typical for us are what we call in German "Leittechniksysteme", i.e. systems for industrial process control applications, as used for controlling electric power distribution, gas pipelines and steel mills. "Process control system" will be used in this sense throughout this paper.

The properties of interest are primarily the requirements, which either are supplied, or must be agreed, by the customer. Later on, the specification will also contain information about the design. Even in the beginning nobody can strictly distinguish design from requirements, because the systems are very complex, and the design will often limit the range of possible requirements. Therefore, we will use the term specification for all information which is relevant for the customer, when the system is being planned.

SPECIFICATIONS

The Importance of Sound Specifications

Boehm (1976) has shown that the early errors, i.e. those which are committed in an early stage of software development, are the ones which are most difficult to detect, so they are usually not detected before the system is being tested or even in operation. The importance of this observation is amplified by a second one: The total costs of a software error are the higher the later it is detected; Boehms results even suggest an exponential rise. There is no reason to believe that these results are valid for software only.

Therefore, almost every expenditure for finding errors nearly as soon as they are committed, or avoiding them at all, will finally pay. Sound specifications, which are well understood both by the customer and by those who translate them into reality, are the most powerful means to achieve this goal, and all provisions for validation of design and implementation are based on the specifications.

Specification Languages

Traditionally, natural languages, mixed with tables and graphics, were the only specification languages. Their advantages are obvious: Everybody can write and read them (at least in his native language), no special training is needed, and tools (like text editors) are readily available.

Experience exhibited, however, several deficiencies of "natural specification languages":

- The syntax is very complex, and virtually undefined. Tools which can not only process character-strings but parse them, according to their grammar, are not very advanced.

- The semantics are even more difficult to handle. Beside the ambiguities caused by the syntax, the meaning is often vague. Since no criteria can be found for logical completeness, the specifications cannot be checked to detect incompleteness.

The uncertainty of the meaning causes many more problems: the communication between implementer and customer, the validation of the product, the preparation of manuals and educational material, they all depend from a correct understanding of the specifications.

These difficulties can be overcome by using formal languages instead of natural ones. A formal specification language can be clear and precise, so the implementer cannot misunderstand it, and tools can be constructed to perform various checks on the specifications.

General specification languages, which resemble general purpose programming languages, are, however, not likely to be accepted by a large fraction of possible users. They want to use their particular jargon, in a most natural way. If they can formalize or abstract something, they prefer tables and graphics, which support their imagination much better than any written language does. Average users also do not fancy strictly formal specification languages, because they are not able to formalize all their ideas, which are necessarily fuzzy when the work starts, in one step.

Thus, a specification language should be a compromise between formality and naturalness, in order to be both acceptable for the user, and advantageous with respect to his productivity and to the quality of what he develops.

Layers of Abstraction

In the requirements specification it is stated

- which inputs (as seen from the control system) are available, and where they can be accessed,

- which outputs must be produced, and where they must be delivered,

- how long the output may be delayed with respect to some input, and

- which degree of reliability is necessary.

These informations form the highest level of abstraction in the specification. They should be separated from all others in order to preserve a clear distinction between genuine requirements and design decisions.

Distribution of a process control system may be suggested by two reasons: The process itself is distributed, or one computer is insufficient for achieving the required performance or reliability. Both kinds of information, the geographical distribution of the process and the figures for performance and reliability, must be addressed by the specifications. But, distribution itself is n o t an original requirement.

There is no need to enforce any particular hardware configuration by the top level requirements.

The user will not (or at least: should not) desire to have a certain number of computers. He needs certain functions to be performed within certain intervals, and he also sets limits for MTBF and MTTR. It is the designer who may conclude that, let us say, three machines, linked together in a certain way, will - probably - meet the requirements. The customer should neither anticipate nor be affected by the designer's decisions.

This ideal separation of a specification level and a design level is prevented by mutual dependencies between them. Traditional design is based on the idea of one particular sequential computer. If several ones are used instead, the situation is completely different, and the specifications cannot remain unchanged. That is why the choice of a hardware system is almost inevitably prescribed by traditional requirements specifications.

Our approach to this conflict is to have the same layers of abstraction both in the conceptual description and in the hardware system (see Fig. 1).

technical process	process control system
location	node, local network of nodes
particular function	pool, consisting of interchangeable processors

Figure 1. Hardware layers.

The unit corresponding to a location where some computational power is provided is called a node. For the different tasks to be performed there, one can assign one or more pools, which are the components of the node. The pools again are made of a varying number of processors, which are completely interchangeable and invisible to the software, thus guaranteeing certain figures of performance and reliability.

Though this model of abstraction, which was first presented by Lalive d'Epinay (1979), does not remove all difficulties, it is an important step towards a better separation of layers. The approach produces not only

more concise specifications but also better designs, because the particular hardware configuration can be fitted to the programs and their computational complexity.

The geographical lay-out of the process control system can be done rather early in the design process. In the layered model, the designer does not need to state the size of the computer for a certain site at this time. He just assigns a node to it.

Later on, as the design proceeds, functions are identified, and assigned to nodes. In this stage, a node may be refined to a local network of nodes. Finally, the particular function is related to a certain pool within a node. Independent from the final distribution, the designer may find that certain functions should be executed in parallel, or even on different machines, while others should not (e.g. for reasons of data integrity). Many functions m a y be performed in parallel, but have not to. These properties should be stated for all functions, in order to help the designer identifying candidates for distributed processing.

Finally, when the software has been designed, a certain number of processors is assigned to each of the pools within every node in order to meet the required performance and reliability. Thus, what used to be a first order problem has been reduced to a simple parameter.

Treating distribution this way, there is no need for special specification systems; ordinary ones are sufficient, provided they allow for expressing the possibility of distribution.

Specification Tools and Specification Systems

Small, stable systems can be specified in a purely manual way. But if the system is large, or changing all the time, some tool is necessary. Such a tool will do some checking on the specifications, preserve them in a database, evaluate them, and prepare various reports which can be used for communication, reviewing, and documentation. Finally, the tool will output his accumulated knowledge in a format which is most convenient for the implementer.

It is the tool which can make an otherwise unattractive specification language successful. Only a tool allows for easy management of voluminous documentation. The tool may also enable the user to choose his favorite representations of information. And for many users, representations of a language (e.g. graphics) are even more important than its concepts.

The combination of a specification language and a set of tools which work on specifications written in that language is called a specification system. Computer aided specification systems have only a very short history. In 1971, the ISDOS-Project appeared on the scene, and its product PSL/PSA (Teichroew, Hershey, 1977) remained the only one in the market-place for several years. Only since about 1977, several competitors showed up, and the topic became popular everywhere in the world. Today, many systems are offered, but the advances are still not too exciting, and in future days, those systems may be viewed like cars built in the 19th century are viewed today.

ESPRESO

ESPRESO is a system for computer aided specification of process control software (Ludewig, 1981). It was developed from 1977 at Nuclear Research Center, Karlsruhe, Federal Republic of Germany. Its fundamental features are:

- a set of concepts dedicated to the modeling of process control software. The emphasis is on communication and coordination between parallel processes. Hierarchical decomposition is supported. Communication between separate units ("modules") is restricted to paths which must be declared explicitely.

- a formal, PASCAL-like specification language. The language is defined by an Extended Attribute Grammar (Watt, Madsen, 1977; Ludewig, 1981b), thus avoiding any ambiguities.

- a set of tools to check, accumulate, manage, and evaluate specifications. The most important tools, which are all written in PASCAL, are operational, others are currently being implemented at Karlsruhe.

The first application of ESPRESO started late in 1980; a nuclear reactor protection system is being specified.

ESPRESO was a first attempt to exploit experiences and ideas about software specification by designing a new specification system. It was very successful in some respects: The concepts seem to be useful, and the complete formal definition is a major improvement, compared to other specification languages. Originally, ESPRESO was also intended to cover not only the software but also the technical process, thus allowing for a complete description of the whole. This goal was not reached, because we did not find t h e typical process. Possible applications seemed to be too different to fit into one single model.

PROCESS CONTROL SYSTEMS

Properties of the Process

Processes like gas pipelines and steel mills, which are to be supervised by the process control systems, are obviously distributed systems. The distances between their components range from a couple of meters up to 1000 km or more. They are made to be in operation for very long periods, typically some decades, which is much longer than the life time of today's computer systems. During that time, the process control system is subject to frequent changes, because the process must be adapted to new requirements and to improved technologies.

Though the process control system is but a comparatively small subsystem in terms of money, it is of crucial importance for the operation of the whole. Even a breakdown of some minutes may be intolerable. During normal operation, a poor control system may prevent optimal performance, and deficiencies of the man machine interface can even cause serious risks. Therefore, one can hardly overestimate the importance of reliability and correctness with respect to the user's needs.

Current Situation

Dispatching systems manufactured by BBC can either be broken down into hardware and software, or into components performing different tasks. From the former view, such a system consists of Indactic hardware systems (Demmelmair, 1979) and BECOS (Brown Boveri Energy Control System) (Blum, Muheim, Weiss, 1979). Larger systems additionally use computers like PDP 11 or VAX.

From the latter view, SCADA (Supervisory Control and Data Acquisition), a data acquisition system, can be distinguished from the power application software (PAS), which contains programs for determination of current network topology, state estimation, and many other purposes (Reichert, 1979). A similar structure can be found in systems made by other manufacturers.

The software, which is similar in many projects, is fairly large; its development takes some 20 or more man-years. So, it would be very nice if it could be used several times, tuned by some parameters which reflect the particular conditions of the project.

To date, some fraction of the software must be rewritten for many projects (in particular for large projects), because the requirements differ significantly, and system structure depends partially on these changing conditions. As mentioned above, the choice of computer configuration is a key decision, because not only the price but also the behaviour is strongly influenced, and the analyst cannot agree on certain requirements without knowing the configuraiton. Many other decisions are mutually dependent in a similar way, and it takes not only a lot of experience but also some courage to submit an offer to the customer.

Thus, a specification system for process control applications might have some advantages beyond the ones listed above:

- availability of old data for new projects,
- easy identification of differences between several similar projects,
- fast elaboration of offers,
- optimal structuring of systems in order to separate those parts which are most likely to be influenced from the requirements particular to a project or from the hardware configuration,
- simple retrieval of reusable software components.

Today, simple tools for text management are used; a real specification system will greatly improve flexibility and abilities.

CARMEN

The most powerful tool which is currently used for system development at BBC is called CARMEN, which stands for Computer Aided design of Real time Monitoring and control of Energy distribution Networks (Schmid and co-workers, 1980). CARMEN supports the engineering of process control systems in the following areas:

- efficient collection of process control data from the customer network, including topology and data of switches, transformers, etc.,
- configuration of the telecontrol hardware,
- generation of documents for the telecontrol hardware layout and the connection to the process control objects,
- generation of the process control database.

CARMEN is fitted to the BECOS-system, which was mentioned above. It is built upon a relational database named PRIMO (Koller, Frühauf, 1979). CARMEN has been used in several projects since 1980; an improved version is currently being developed.

For our considerations, CARMEN is an excellent example of what is needed: It is tailored to a particular application area, inputs and outputs are in the language of the people who must supply and use it, and its concepts are sound and well defined.

AN INTEGRATED SPECIFICATION SYSTEM

For the future, a system like CARMEN, but applicable to a wider range of problems and during the whole life time of systems, is needed. It should allow for all kinds of specifications, including the process, the process control system and, in particular, its software. It should be used not only during the development but also when it is tested, corrected and modified. Such a system would in fact be more than just a specification system; it might be called with a word which recently became very popular an environment for system development and maintenance ("ESDAM").

ESDAM consists of

- a project database, to keep as much as possible information about the project in the computer. That includes requirements, design, software code, layout of prints, manuals, and all other information which can be reasonably represented by strings and numbers;
- a specification language, tailored to the application, with options for input in tables, possibly also in graphical form;
- a language-controlled editor for preparation of specifications;
- a tool for validation, which may include a simulation facility;
- a report and documentation generator;
- a man machine interface, for simple interaction with the tools.

The project database is also used to maintain different versions of programs, as they exist during software development, or for different customers.

Concepts for ESDAM

Every language is based on a model (or set of models). In our field, we need at least two models, one for the process, which should e.g. comprise breakers and transformers in case of a power distribution network, and a second one for the control system; this can be the one which was used in ESPRESO, or something similar. Apparently, no single set of models is sufficient.

As a system for an industrial environment, ESDAM must take tables and graphics as important patterns of information into consideration, or otherwise it will fail. Though this is not really a concept, it should be kept in mind, in order to avoid concepts too complicated for these representations. (In general, man cannot understand concepts which cannot be displayed graphically.)

The Data Base

The heart of every specification tool or environment is a data base management system, which releases the user from all kind of clerical work. Since environments will be available as parts of ADA systems, we will try to use such an environment rather than implementing a new one. This approach will also ease the transition from specification language to program code.

Editors

Different from most compilers, which process the source code every time they are invoked, specification systems usually store the information which is derived from parsing the input, thus allowing for piecewise accumulation of a specification, and immediate checking of the input against all former inputs. Therefore, a specification system needs a more sophisticated, language oriented editor, which accomplishs all the conversion between the internal and the external representations.

For tables and graphics, special editoring facilities are required.

The Meta-System Approach

Specification, design, and implementation of ESDAM will take at least 30 or 40 man-years. Such an effort can only be justified if the environment is very likely to be accepted and used for a long period. To date, our knowledge and experience in this field is still far from sufficient, so we cannot guarantee any single concept to be successful. We must try various models and representations, and improve the solution by the feedback we receive.

A possible way out of the conflict between the need for a specification system and our inability to propose mature concepts is to implement not one particular model or language, but a generator which can be used to produce a variety of systems with little effort. This generator will also allow for definition of "dialects", as used by the engineers in different environments. Such a generator was first used at ISDOS; their so called META-Generator can transform a formal language definition into a set of tables, which in turn control the Generalized Analyzer. This toolset was used to define several languages other than PSL, e.g. PCSL (Ludewig, 1980). The range of defineable languages, however, is still rather restricted, and the reporting tools are not yet fully integrated. So, future developments will not be built upon the ISDOS system, but exploit the experiences from its use.

First of all, a meta-concept must be chosen. Such a concept may be the object-relation-model. Knuth (Teichroew and co-workers, 1981) showed a different, homogenous model, based

on so called "concepts" only. Then a meta language can be defined, and a processor for this language must be implemented. All other tools like data base systems and editors are table driven.

CURRRENT STATE

The ESDAM, which was outlined above, is currently only a plan, which requires further investigation. Various models are being evaluated, for instance the modeling of process control systems by extended finite state machines (Vitins, 1980). The ESPRESO-tools will be implemented at our research center. They can be used to evaluate the concepts, because they already incorporate some of the ideas on table driven systems. All ideas will be thoroughly discussed with potential users, who will finally decide about success or failure.

CONCLUSION

It was shown that an environment for specification, design, and maintenance of process control systems is both necessary and feasible. If considerations about geographical distribution, realiability and performance can be clearly separated from the functional requirements, a specification model for distributed systems is not significanlty different from one which is made for ordinary process control systems. As shown in the paper, this decoupling can be achieved by a clear distinction of abstract layers, which is supported by a hierarchical decomposition of both software and hardware. In order to get a flexible set of tools, which can be easily modified and adapted to new ideas, a generator will be realized rather than just one particular system.

ACKNOWLEDGEMENTS

Most of the ideas presented in this papers emerged from the joint work of the computer science group at Brown Boveri Research Center. CARMEN and BECOS were developed at BBC.

REFERENCES

Blum, A., Muheim, J., Weiss, J., (1979). BBC software for dispatching systems. Brown Boveri Review, 66, 164-168.

Boehm, B.W. (1976). Software Engineering. IEEE Trans. Comp., C-25, 1126-1241

Demmelmair, K. (1979). BBC Indactic 61 - An automated system for power supply networks. Brown Boveri Review, 66, 161-163.

Koller, H., Frühauf, K. (1979). PRIMO data base management system. Brown Boveri Review, 66, 204-209.

Kramer, J. (Ed.) (1981). Glossary of terms for EWICS TC 11. Up-to-date version, available from J. Kramer, Imperial College, London.

Lalive d'Epinay, Th. (1979). Structure of an ideal distributed computer control system. In T.J. Harrison (Ed.), Distributed Computer Control Systems, Pergamon Press.

Ludewig, J., Streng, W. (1978). Methods and tools for software specification and design - a survey. EWICS TC 7, Paper No. 149, Zürich.

Ludewig, J. (1980). PCSL - A process control software specification language. KfK-Report No.2874, Kernforschungszentrum Karlsruhe, FRG.

Ludewig, J. (1981a). Zur Erstellung der Spezifikation von Prozessrechner-Software. Doctoral dissertation, Technical University Munich. Reprinted as KfK-Report No. 3060, Kernforschungszentrum Karlsruhe, FRG. (in German)

Ludewig, J. (1981b). Specification of a specification language. To appear in Hasegawa (Ed.), Real Time Programming 1981. Pergamon Press.

Reichert, K. (1979). Application Software for power system operation. Brown Boveri Review, 66, 197-203.

Schmid, H.P., Litynski, A., Fransson, F., Koller, H.U. (1980). Unpublished results.

Teichroew, D., Hershey, E.A. (1977). PSL/PSA: A computer aided technique for structured documentation and analysis of information processing systems. IEEE Trans. Software Eng., SE-3, 41-48.

Teichroew, D., Knuth, E., Rado, P., Kang, K.C. (1981). Concept refinement approach (CRA), a system specification technique. ISDOS-Project, University of Michigan, preliminary draft.

Vitins, M. (1980). Requirements specifications for industrial real-time automation systems. Brown Boveri Research Center, KLR 81-5 C.

Watt, D.A., Madsen, O.L. (1977). Extended Attribute Grammar for PASCAL. SIGPLAN Notices, 14, No. 2, 60-74.