PROTOCOL CONVERSION FOR INTERCONNECTING ENERGY MANAGEMENT SYSTEMS.

J. Luque, F. Gonzalo, F. Pérez, M. Mejías.

Facultad de Informática.
Universidad de Sevilla.
Avda. Reina Mercedes s/n.
41012-SEVILLA. SPAIN.

Abstract.- The difficulty to interconnect telecontrol networks from different power utilities, and even different network components into the same utility, is an important inhibitor to the growth, flexibility and economy of such networks. In this paper the solution reached through an R&D program sponsored by the Spanish Ministry of Industry through the PIE project "CUP" (PIE is the spanish acronym for Electrical Research Plan) is discribed.

INTRODUCTION

The increasing complexity of power networks, and its very high interconnection grade, causes a growing need of communication between different Energy Management Systems. However, such equipment are commonly of different period, technology and vendor. In this situation, the communication among them becomes a serious problem. Many efforts are being accomplished to define a set of standardized protocols, both for the center-remote communication and for the link between centers of the same or different levels. Nevertheless, the role to be played in the future for those standards is not yét clear and the problem of communicating systems working at present remains.

Paper APT 125-02-03 accepted for presentation at the IEEE/NTUA Athens Power Tech Conference: "Planning, Operation and Control of Today's Electric Power Systems", Athens, Greece, Sept. 5-8, 1993.

The typical solution to communicate two incompatible systems is to build a piece of equipment that is able to convert the involved protocols. Such equipment is usually called a protocol converter. This task is commonly performed using ad hoc techniques, that is, solving every peer protocol conversion in a particular way.

As an alternative, Sevillana de Electricidad, the power utility serving the south of Spain, jointly with the University of Seville and several local vendors of control systems, have developed a project to solve the problem in a more general way, building an automatic conversion tool called "CUP", which stands for the spanish equivalent of "Universal Protocol Conversion". During the project design phase, the capabilities of the Formal Description Techniques (FDTs) were considered, in regards to its application to the specification, simulation, implementation and testing of communication protocols. After a detailed analysis, the ESTELLE language, standardized by ISO, was chosen to specify, using Extended Finite State Machines (EFSMs), the behaviour of the incompatible protocols to be converted. In the following text, the CUP tool is described and the results of its application to real-world situations are discussed.

THE CUP TOOL

General description

If two incompatible telecontrol protocols can be stated using an FDT with the conversion

algorithm being described in the same way, the full protocol converter can be formally written. In this case, all the design, test, simulation, implementation and conformance techniques applied to the protocols, can also be applied to the protocol conversion. As an aiding tool in this task, the CUP system has been developed. The CUP tool takes the formal converter specification, and provides support to develop all the tasks from the design to the implementation phase.

The low range platform to run the whole system has intentionally been chosen. More precisely, it is able to run in PC-compatible systems with 640 Kbytes of main memory, 20 Mbytes hard disk and, optionally, graphic adapter CGA or higher; everything running under the MS-DOS operating system. Although many of the tools for processing formal specifications require systems with higher performances, typically workstations under UNIX, a big effort should be made to reduce at a minimum the hardware requirements, in order to popularize the use of these techniques. For that reason the high number of PC-compatible systems installed at present, invites to develop tools for this platform.

The CUP system accepts in the input the formal description of both the protocols and the conversion process, specified in ESTELLE [1,2]. There are thrre reasons why we chose ESTELLE as the specification language: a) it is a technique standardized by ISO; b) it is easy to use and learn, even for people who are not highly specialized. This is due to the fact that it is very similar to a conventional programming language (PASCAL); and c) it is easy to automatically convert the original specification in an executable code because ESTELLE has been designed with an implementation approach. However, none of these reasons are definitive and, therefore, different solutions based on techniques as LOTOS [3] , SDL [4], or some others, are also possible.

The present version of CUP implements a wide subset of the standard, which is highly significative for the needs usually shown in the specification of protocols and their converters. It is foreseen that later versions will implement the full standard. Besides, and as a variant according to the standard, it is possible to express some of the specification following the C language syntax, instead of using the recommended PASCAL syntax. This is consistent with the present trends to choosing programming languages for communication software.

The generation process

Once the relevant formal descriptions have been supplied to the CUP system, the first logical process is the generation of the corresponding executable code. A generation process such as this, is accomplished in three steps:

a) Translating the ESTELLE specification to an intermediate program written in C. Such a translator is built using YACC [5] which, although it is a development tool typically used in UNIX-based machines, can also be found in MS-DOS environments. The translator also produces the "structural tables", where are stored the organization (states, events, conditions, transitions, outputs) for every EFSM described in the specification.

b) Compiling the C intermediate program and obtaining a relocatable module (object file). This process is made invoking any commercial C compiler.

c) Linking the object file obtained previously with some other modules and libraries prebuilt in the CUP system. This process is also made by invoking any conventional linker.

Each of these three steps are fully integrated with each other in

a way transparent to the user, presenting a single man-machine interface. Additionally, an edition step is also integrated into the previous ones, which makes the introduction and maintaining of the ESTELLE specification easier, facilitating the correction of the errors occurred in any of the generation steps. Likewise, the generation phase includes the optional issue of several reports concerning every one of the steps.

The simulation process

The executable code obtained in the generation process will rarely be free of execution or logic errors. To help the designer with the difficult task of debugging the specification, the CUP system includes a simulation process. In this process, the user has access to the following options:

a) Choosing the execution speed. The events and transitions can be fired step by step, at nominal speed (real time), or at user-definable rate.

b) Setting break points. A specification can be run until a break in the normal flow occurs. Such break point can be defined when a module (or module instance) reaches a specific state, or when a certain transition is fired. Whenever a break occurs, the normal execution is stopped, the state of every module can be examined and the specification can be executed step by step.

c) Presenting the execution of the specification. Such a presentation permits testing, for every module and module instance, its state, the last event occurred to the module, the last transition fired, the last output queued, etc. As it can be seen, this presentation can also be made in a graphical way.

At the end of the simulation process the user will be conscious of some of the converter specification errors, and should go back to the edition phase for generating a new converter. This loop will be repeatedly done until a code is obtained satisfying all the designer quality requirements. The path from edition, to generation, simulation, and back to edition, is done with no change in the user environment due to the fact that all the phases are fully integrated into the CUP system.

The conversion process

When the specification is considered error-free, the conversion process can be undertaken, what means the execution of the specified system, transmitting and receiving information through the communication lines, and performing the conversion algorithms for the specified protocols. Although the conversion process usually run in a way transparent to the user, several mechanisms to look inside the system are provided. This will allow the debugging of errors undetected in previous phases. The control mechanisms are:

a) Presenting the information traveling through the line in a tabular or graphical way, and referring the evolution of the specifications internal modules. This type of presentation is quite similar to the one existing in the simulation process.

b) Storing the information in a disk for further analysis.

The use of these real time control mechanisms, mainly those implying any kind of presentation to the operator, are restricted by the real time requirements in the converter execution. Therefore, depending on the performance of the platform where the system runs, and on the number and speed of the communication lines which are being converted, these control mechanisms will or will not run. Anyhow, the system should be sized, according to the most demanding application, so to have access to the function of storing the line information on

a disk.

SPECIFICATION LIFE CYCLE

Although the use of the CUP tool does not imply the adoption of any methodology for developing protocol converters, during the design of the CUP system, and its use to obtain telecontrol protocol converters, a life cycle has been used which, starting from the converter's requirements, assumes the set of tasks to be accomplished by the designer, and his or her interaction with the system. This life cycle can be broken up into the following steps:

1) The designer should obtain enough information to specify the protocol which will be called A.

2) If this information is not written following a formal specification, or the format is different from the one chosen by CUP (it is not written in ESTELLE), the designer should express such a protocol according to the right syntax. In the case that the protocol is multilayer, the designer should specify every one of the layers, processing them separately, and proceeding through the steps described in the following paragraphs.

3) The obtained specification is introduced into the generation system, obtaining information regarding any possible syntactic or semantic errors (static errors). This process will continue until an error-free specification for protocol A (or a layer of A) is obtained.

4) With the previous specification the test modules which allows its execution are included. More exactly, it would be necessary to specify at least both a user and a destination module. The new specification is debugged of syntactic and semantic errors.

5) Using the protocol simulation process, the evolution of the messages in time can be observed. This process is presented, either on the screen or through the printer. The simulation process allows the designer to detect and correct logic errors, obtaining at the end of this repetitive process, the formal specifications for the protocol A (or one of its layer), and an automatic implementation of it.

6) Testing the so obtained protocol A implementation by communicating with a center and/or a remote using such protocol. Correcting the specification until reaching an error-free version.

7) The converter designer should repeat the steps from 1 to 6 in order to specify all the layers of the protocol A and, also, all the layers of the second protocol, which will be called B.

8) The converter designer should determine the correspondence between functions, services and messages for both protocols, reaching the formal specification of the conversion algorithm, taking advantage of the protocols A and B formal specifications previously obtained. In this way, the task of obtaining the formal converter specification is a simple one if protocols A and B provide similar services, which commonly occurs in the Energy Management Systems. The converter specification should also be submitted to the syntactic, semantic and simulation analysis, previously described.

9) Once the validity of every input specification has been tested, the CUP system will generate the executable code required for the real time operation of the specified converter.

10) Lastly, the final performance of the converter could be controlled and, possibly corrected, using the monitoring process included in the CUP system.

CUP has been used to obtain three different protocol converters to interconnect electric control centers and RTUs supplied by different and incompatible networks. The operation of these converters has been succesfully

tested not only in the factory but also in several substations under real conditions.

CONCLUSIONS

In this paper the authors' efforts have been presented to develop a tool which, applying the formal description techniques, efficiently support the design and implementation of protocol converters in the field of telecontrol networks. The results obtained with the prototype version have been highly successful. The tool performances have been tested in actual applications developing a converter for the protocols used in the real time control of the power network in the south of Spain.

REFERENCES

[1] ISO 9074:1989(E) "Estelle: A formal description technique based on an extended state transition model".

[2] S. Budkowski and P. Dembinski "An Introduction to Estelle: a Specification Language for distributed systems". Computer Networks and ISDN Systems. Vol. 14, No. 1, pp.3-23, 1987.

[3] ISO 8807:1989(E) "LOTOS- A formal description technique based on the temporal ordering of observational behaviour".

[4] CCITT "Specification and Description Language SDL". Recommendation Z100-Z104. ITU 1984.

[5] S.C. Johnson, "YACC: Yet Another Compiler-Compiler", Unix Programmer's Manual. Bell Laboratories, 1978.

Joaquín Luque received his Ingeniero Industrial degree in 1980 and the Ingeniero Industrial Doctoral degree in 1986, both from the University of Seville (Spain). Since 1980, Dr. Luque has worked for several companies in the area of SCADA systems for electrical networks, participating in some of the main projects of Energy Management Systems in Spain. He is currently Professor of Electronic Engineering in the University of Seville. Dr. Luque is a Member of the IEEE.

Fernando Gonzalo received his Ingeniero Superior de Telecomunicación degree in 1971 from the University of Madrid. Since then he has worked in planning, design, construction and maintenance on both telecommunication networks and telecontrol systems for Compañía Sevillana de Electricidad. He is currently Manager of Telecommunication and Telecontrol in this power utility. Mr. Gonzalo is a Member of IEEE.

Francisco Pérez received his M.S. degree in 1985 and the Ph.D. degree in 1992, both in Physics, from the University of Seville (Spain). Dr. Pérez has participated in several projects in the field of Robotic, Control and Security Systems. Since 1987 he has been a Professor in the Department of Electronic Technology in the University of Seville, where he teaches Logic Design and Computer Networks.

Manuel Mejías received his Ingeniero Industrial degree in 1982 from the University of Seville (Spain). Since 1982, Mr. Mejías has worked for several companies in the area of informatic systems, participating in some projects of Central Thermic supervision systems. He is currently a Professor of Software Engineering in the University of Seville.