MUL'TIPROTSESSIST system [7]. We assume that the implementation file already contains the implementations of the new elementary operators and conditions. Otherwise, they must be created using a text editor. After the program has been compiled in the object language, the user may view the SAA-schema of the transformed program.

LITERATURE CITED


FORMALIZATION OF SOFTWARE DESIGN FOR PROCESS CONTROL SYSTEMS IN FERROUS METALLURGY

V. G. Akulovskii

The operation of control systems in bar-rolling mills is examined and software design tools for such systems are proposed. A system of algorithmic algebras for these design tools is constructed and its capabilities are considered.

Despite the importance of ferrous metallurgy in the national economy, the state of process automation in this industry is still unsatisfactory [1]. The automation is often below the required level [2] and is substantially lower than in the most industrial countries, in particular Japan [3]. This lag is apparently attributable to the difficulties with the development of process control systems in black metallurgy, on the one hand, and to insufficient attention devoted to the problem, on the other.

The dominant position of so-called local control systems (LCS) for processes in ferrous metallurgy has been noted in [1] and the characteristics of these systems have been considered in [4]. The large scale adoption of LCS requires reduction of development costs and delays, while focusing the attention on computer-aided design of such control systems. It is noted in [4] that the availability of computer-aided software design systems is the most effective factor influencing the LCS introduction rates. The well-known multilevel structured program design (MSPD) method [5, 6] has been proposed as the core of such systems, and a system of algorithmic algebras (SAA) for LCS software design has been constructed.

Section 1 briefly describes the object of control using the example of a bar-rolling mill and examines the operating conditions of control systems for these mills. The main software requirements for LCS operating under these conditions are identified. The requirements that must be satisfied by software design tools for producing quality software are also identified.

Section 2 proposes a system of algorithmic algebras (SAA-L), which formalizes the LCS software design process. SAA-L includes design and transformation tools for parallel and sequential algorithms. Section 3 analyzes and demonstrates the capabilities of the algebraic system for satisfying the requirements from the automatic program design system. In the concluding section, we discuss the efficiency of MSPD and the scope for the use of an extended version in a new application domain.

1. PROCESS CONTROL SYSTEM IN BAR-ROLLING INDUSTRY

As an example of the automated object, we consider a small-bar rolling mill, which is a large complex of interacting technological equipment. In the simplest case, the complex includes a number of main units: heating furnace, several groups of rolling cages, tipper, industrial shears, roller path, one-sided cooler, and a host of supplementary and auxiliary mechanisms and systems. The rolling process can be described in elementary terms as follows. An ingot 7-10 m long heated to around 1200 °C is unloaded from the heating furnace by the tipper and reaches the rolling cages. The rollers, driven by 600-800 kW motors, spin inside the cages, compressing and conveying the ingots. At the output from the last (cleaning) group of cages, we have a finished bar up to 1000 m long, moving with a velocity of 25 m/sec. The bar is cut by shears into strips up to 100 m long, which are conveyed by the roller path to the cooler. The rolling process in the mill is usually controlled by several control systems [7].

This description does not provide a full picture of system operation. However, even this brief description shows that the control system is required to function under extreme conditions. In particular, these include harsh climatic conditions with air temperatures fluctuating between wide limits, strong electromagnetic fields acting as a source of interference for the automatic systems, large quantities of electrically conducting dust, vibrations, large geometrical size of the automatic system, extended communication lines, etc. These extreme operating conditions often cause failure of the LCS, distort the information in communication lines, and lead to operator errors.

In continuous operation, the ingots are delivered to the rolling mill in a virtually uninterrupted stream. Therefore, because of the high rolling speeds, any delay in generating the control signal beyond the specified limits is tantamount to failure of the control system.

Finally, consider another important feature of equipment control systems under the above-described conditions in bar-rolling mills. The control system is "responsible" for the control quality in the sense that it must ensure workers' safety and avoid the heavy material losses associated with poor control.

Leaving aside the problems of reliability and speed of control system hardware, we reach the following conclusions from the above discussion. The LCS software for bar-rolling mills must satisfy more stringent requirements than large commercial and scientific systems, information systems, etc. In addition to the standard requirements for most process control systems, such as maintainability, flexibility, portability, etc., the software for the bar-rolling control system must also meet the following requirements:

1) reliability — the ability of the software to ensure correct results under all possible combinations of input signals and input data in the presence of hardware faults and operator errors;

2) reactivity — the property of the program to generate, under any combination of input signals and for any conceivable arrival rate, output (control) signals with a delay that does not affect the quality of the technological process.

As we have noted above, the MSPD method is proposed for program design, because it meets most of the requirements presented to software design tools. In particular, the MSPD method provides a unified design framework for the entire complex of programs in all development stages, allows software optimization by given criteria, uses a design language which is accessible to all developers, permits selecting among different languages in the coding stage, generates simultaneous design documentation, etc. We will consider the tools that the MSPD method uses in order to meet these requirements, which are highly characteristic of the application domain considered in our paper.

Of central importance in this case are reliable debugging tools. A necessary condition for the creation of these tools is the possibility of developing the program together with its debugging tools within the framework of a single design. In this case, each design level is equipped with adequate debugging tools. In the introduction stage, when program errors, hardware faults, etc., may lead to failure of the controlled plant, debugging tools should be replaced with tools for handling emergency situations and failures. Both the debugging tools and the failure-preventing tools must satisfy the following conditions:

— they should not obstruct the development of a clear (readable) design;

— they should ensure execution of debugging and failure-preventing operations in such a way that program running time increases only when errors and failures actually occur.