CULEGERE DE LUCRĂRI ŞTIINŢIFICE
TEHNOLOGII MODERNE, CALITATE,
RESTRUCTURARE
Chişinău, mai 2007

DEVELOPMENT OF RECONFIGURABLE SOFTWARE MODULE FOR CNC MACHINE TOOLS

FLORIN BOGDAN MARIN, IONUT CLEMENTIN CONSTANTIN,
VASILE MARINESCU, CIPRIAN CUZMIN, ALEXANDRU EPUREANU

Key words: reconfigurable machine tools, open-architecture, interpreter, condition instructions set

Abstract - After reconfiguration the new hardware architecture implies also software architecture changing. Reconfigurable machine tools performance is determined most by ramp-up-time. A important part of this is the time needed for software reconfiguration. This paper presents a reconfigurable interpreter, developed as new software approach, which provide a fast software reconfiguration to be operated by the machine user itself. The interpreter design methodology and corresponding software presented in the paper was developed at the "Dunarea de Jos" University – Galati. Results achieved confirmed the efficiency of this approach.

1. Introduction

Market demands for products tend to change rapidly. To respond to such changing mix and volumes, manufacturers must modify the structure of their machine tools, to be able to change their production schedule. There are different manufacturing systems: dedicated, flexible and reconfigurable.

A Reconfigurable Machine Tool (RMT) is a concept that describes capacity of a machine to modify its structure, in order to allow a fast change of both hardware and software parts according to new demands of the market [1][2].

In this paper we propose a reconfigurable interpreter for CNC machine tools allowing implementation of RMT hardware taking into account Open Architecture concept [3], which we consider to be a must have condition in designing software for RMT.

The goals of this report are: a) to define the concept of reconfigurable interpreter; b) to define a methodology for designing a reconfigurable interpreter; c) applying

II. ARCHITECTURE OF RECONFIGURABLE INTERPRETER

In order to achieve the goal of development of an interpreter we can divide the problem into the following basic parts: a) choosing a language to be interpreted; b) writing an execution code for the language; c) providing a debugger; d) providing a GUI interface.

Software module represented by interpreter was implemented on a turning machine in order to proof the concept of reconfigurable interpreter. The language used in the software interface is well known *machine tool programming language ISO CNC Code.* However, the interpreter was designed in order to adapt to any new programming language as is shown below.

The information circuit in a CNC starting with the user and ending with the effectors is described conceptually in figure 1.

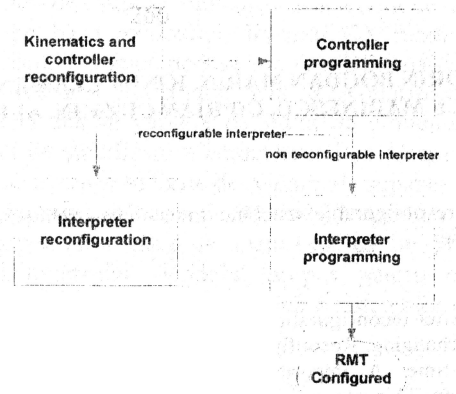

Figure 1. Information circuit in CNC machine

When preceding to reconfigurate the machine, a new hardware configuration is builded including new controllers and a different kinematics configuration. As a consequence, the control component and software interface are required. Whereas software interface for basic command is not an issue, building the interpreter, as a matter of fact the main core of the command for the user is a challenging task. Nowadays interpreters are designed to execute implementation of a language, as long as machine tool produces. The RMT paradigm requires designing reconfigurable software modules, including interpreters. Furthermore is needed to be taken into account reconfigurability, as new programming languages are proposed. We are here taking into account high level language implementation.

Each kinematics configuration of machine tools determines new parameters as machine travel limits, feed rate, spindle feed, etc. Moreover, some of the programming functions may not be available in the new

configuration.

Of a paramount interest are the conditional relations between function, as some functions are deactivating others, and these conditions depends on kinematics configuration.

To complete the task of reconfiguration of a RMT, three steps are to be followed: a) kinematics reconfiguration; b) controller reconfiguration; c) interpreter reconfiguration. In comparison with nowadays interpreters, the new reconfigurable interpreter lead to reducing the ramp-up-time. Concerning the reconfigurability of an interpreter we make the following assumptions and statements. Firstly, we define a *reconfigurable* interpreter as one with the some or other programming language to be used in all further configurations. During the configuration process, some functions or instructions are disabled; the meaning is modified according to the new machine. Also, an *reconfigurable* interpreter is allowing implementing others languages including a new language that is suppose to emerge, besides the main characteristic, that of allowing changing general settings for the new configuration. Basically all functions of a programming language may be added in the *reconfiguration* process.

The time of implementation new configuration for interpreter is a matter of hours, whereas that of reprogramming the interpreter according to new hardware configuration is not satisfying the condition stated for RMT, that is short ramp-up-time.

III. ALGORITHM DESCRIPTION

The former approach works well for single pass interprets but is, generally, more limiting since with the new reconfigurable concept in mind, the changing of configuration is an issue. An interpreting environment can be made interactive, so that changes can be to speed up the development process by providing quicker feedback on changes to the grammar. As reconfigurable interpreter involves new definitions of functions, the grammar is changing accordingly.

As stated above, each kinematic configuration of machine tools has different characteristics as machine travel limits, feed rate, spindle feed, etc. Moreover, some of the functions may not be available in the new configuration.

The issues that are encountered in the grammar of the interpreter component are: a) some of the functions may not be available in the new configuration; b) functions defining machine travel limits, feed rate, spindle feed etc, are modified/voided; c) the working space is modified.

The code is examined alternatively by running on a *conditions instruction set*, a technique that allows great power in its ability to halt when specific conditions are encountered. These instruction set conditions are the key in this approaching of the debugger. Each issue concerning the relation between function are deal with conditions that are tested on code. As we use ISO CNC

code for programming turning machine, there are four of G functions: a) modal G functions, functions belonging to a family of G functions that cancel one another; b) nonmodal G functions, functions enabled only in the block where they are programmed (cancelled at the end of the block); c) G functions incompatible with the state of the program, functions whose programming is enabled or not according to the state of the current program; d) G functions associated with arguments Functions followed by one or more arguments that are specific to the G function announcing them. These possibilities of relation between functions are treated by *condition instruction sets algorithm.*

The idea of these conditions allows changing the language programming to be interpreted. The software allows fast and easy reconfiguration and defining of functions.

To date we have used the turning machine system to define a wide range of scenario from testing as diverse set of conditions may occur for different configurations. We have successfully supported the execution of interpreter processes and have begun to apply powerful algorithms to the process definitions to prove critical properties of reconfigurability. This experience suggests that our approach to process definition has very broad applicability.

IV. APPLICATION DESCRIPTION

Besides the interface for command and editor for interpreter, the settings panels represent the component of software that allows defining functions behavior. The interpreter has been designed for easy and fast configurations of functions and arguments. The data exchange is based on a set of different information that allows change relations between functions according to their properties at one moment in execution. "Negation Prop" ,"Arguments", "Union" buttons adds the code field, where appropriate, to existing function and adds a new behavior properties for each function. Also the interface allows to define arguments for each function and defining range values of arguments.

ACKNOWLEDGMENT

The authors gratefully acknowledge the financial support of the Romanian Ministry of Education and Research through grants CEEX 22 and CEEX 23.

BIBLIOGRAPHY

[1] Koren, Y., Heisel, U., Jovane, F., Moriwaki, T., Pritschow, G., Ulsoy, G. and VanBrussel, H., (1999), *Reconfigurable manufacturing systems*. Annals of the CIRP,48, 1–14.

[2] Chrin, J. L. and McFarlane, D. C., (1999), *A migration strategy for the introduction of holonic production control*, IFAC Multi-Agent-Systems in Production, Vienna

[3]. Cho Y. S and. Carver D. L, (2004) *A model for software reuse*, Journal of Systems Integration, 6(3), pp. 181–201