24th COBEM - 2017

24th ABCM International Congress of Mechanical Engineering
December 3-8, 2017, Curitiba, PR, Brazil

COBEM-2017-2709
UNIVERSAL PARAMETER LANGUAGE FOR THE PROGRAMMING OF NUMERICAL COMMANDING MACHINES

Marco Aurélio da Fontoura Gonçalves
Moacir Eckhardt
Cristiano José Schever
Federal University of Santa Maria, UFSM, Industrial Technical College, Av. Roraima nº 1000, Santa Maria, RS, University City, Neighborhood Camobi, Zip code: 97105-900. Brazil
marcoctism@gmail.com, moacir@ctism.ufsm.br, cristiano.scheuer@gmail.com

Abstract. *This work has as objective the development of a parameterized universal language for the programming of numerical control machines. For the interpleading and compilation of this language a computational application is developed called parameterized programming module MPP. In order to test the system, an angular, transversal and longitudinal thinning cycle is programmed, showing the operability of the algorithm. The proposed parameterized language has a differentiated structure of the languages provided by the command manufacturers. It is a complementary tool in the use of parameterization in numerical programming. The algorithm proved to be reliable in the interworking and compilation of the numerical code.*

Keywords: *CNC programming, Advanced language, machining strategies.*

1. INTRODUCTION

Despite the efficiency of CAM Computer-Aided Manufacturing systems, basic programming is largely employed, especially in the programming of basic geometries. In this programming form, there are functions that aid the programmer and are available in the respective Computer Numeric Control (CNC) commands. Examples of these functions are machining cycles for roughing and cavity operations (Davim, 2008; Groover, 2011).

Parameterized programming is a form of advanced manual programming that allows you to include in CNC programs mathematical calculations, computational variables and conditional deviations. This allows implementing algorithmic logic in the operation of the numerical control machine. The numerical command manufacturers provide the parameterized programming methodology, being specific for each type of command (Lynch, 1997; Smid, 2003).

The Parameterized Programming Module (MPP) is composed of a universal hybrid programming language, elaborated with Pascal programming fundamentals (Manzano and Yamatumi, 2007) and incremented with the G code (General or preparatory) by the ISO Standard (Smid, 2003).

The purpose of MPP development is to facilitate programming, not needing the knowledge and mastery of multiple programming languages, since each command has its own language. With this, the learning and operability in the elaboration of specific machining operations, based on the parameterization, tends to become more accessible. CNC machines, where they occur to the application of parameterized programming resources, in the development of specific machining cycles, tend to increase their productivity (Gonçalves, 2007). With this study we have a reference to the application of a programming technique that although little diffused, when properly used allows to generate routines for standardized or parameterized geometries.

2. COMPUTATIONAL PROCEDURE

In this chapter we will describe the hybrid programming language of the proposed system containing basic functions applied to the machining tool movement requirements and the characterization of the developed algorithm.

2.1 The proposed language

The proposed language is based on Pascal fundamentals with the insertion of G (General or preparatory) Codes by the ISO Standard.

2.2 Definition of variables

The variables are of the actual type and declared at the beginning of the program after the "VAR" declaration, as the syntax shown below.

```
Var
Variable1: real;
Variable2: real;
```

Assigning values to variables

To concatenate values in the variables is used ": =" according to the syntax:

```
Begin
Variable1: = 20;
Variable2: = 30;
end;
```

Note: The command block is represented by "begin" and "end";

2.3 Arithmetic Operators

In the proposed language one can perform arithmetic operations on variables. Table 1 shows the available operators.

Table 1. Arithmetic Operations

Operator	Meaning
*	Multiplication
/	Division
+	Sum
-	Subtraction

You can perform an arithmetic operation between two variables and assign the result to a third variable by adding a line of code with the following syntax:

Variable3: = variable1 variable2 operation;

2.4 Relational Operators

Table 2. Relational Operators

Operator	Meaning
=	Used to test whether two values are equal.
<>	Used to test whether two values are different
<	Used to test whether a value is smaller than another.
<=	Used to test whether a value is less than or equal.
>	Used to test whether a value is greater than another.
>=	Used to test whether a value is greater than or equal to

2.5 ISO Code

The programming code is entered numerically controlled interleaving with the algorithmic structure, characterizing the proposed language as a hybrid structure for Interpretation compiler. Table 3 shows examples of accepted codes in programming.

Table 3. ISO code

ISO (code G)
G90
G54 X -150 Z -30
G00 X 100 Y 400 Z150
T2 D2 M3 S1200
G01 X2.5 Y45 F200
Y30
Z-80.15
G91 X10
G90 Z7 F100
X10 Y10
G02 X-10 Y-10 I -10 J -10 F300 M8
G01 Z10 F200
G03 X10 Y10 R20 F400
M30

2.6 Structure conditional if-then-else.

There are situations where you want to execute a piece of code only if a certain condition is true and another code snippet, if the condition tested is false, the structure conditional if-then-else statement has the following syntax:

```
if (condition)
then
  begin
            GO.....
            G1.....        Block commands executed with CNC code if
            G3.....        the condition is true.
            M14
end
else
  begin
            GO.....        Block commands executed with CNC code
            G1.....        if the condition is false.
            G3.....
            M14
  end;
```

2.7 Repetition structures while loops

This repetition structure is used when you want to execute a block of commands to be repeated while a certain condition is true, the repeating structure has the following syntax:

```
while < condition > do
begin
            GO.....        Block commands executed with CNC code
            G1.....        while the condition is true.
            G3.....
            M14
end;
```

Figure 1 shows a listing of a program developed with the proposed language for holding the longitudinal roughing operation. The initial parameters are: initial diameter at X = 24 mm, diameter end at X = 12 mm, depth Z = 40mm, the initial position Z = 0 mm and 1 mm depth of cut.

```
var

xinicio : real;
xfinal : real;
zinicio : real;
zfinal : real;
xatual : real;
prof : real;

begin

xinicio := 24;
xfinal :=12;
zinicio := 0;
zfinal := 40;
prof :=1;
xatual := xinicio;

while (xatual>xfinal) do

begin

G0 X(xatual-prof) Z(zinicio+1);
G1 Z(zfinal);
G0 X(xatual);
G0 Z(zinicio +1);
xatual:= xatual -prof;

end;

if (xatual<=xfinal) then

begin

G0 X(xinicio) Z(zinicio +5);

end;
end;
end.
```

Figure 1. Structure with the proposed language

2.8 Characterization Algorithm

To interpret the proposed language was developed a strategy for the preparation and compilation of the program. Fig. 2 illustrates the graphical interface developed, composed of two areas of editing, the first left is the environment to edit the language proposed in the parameterized system the second is the right to edit the generated CNC code.

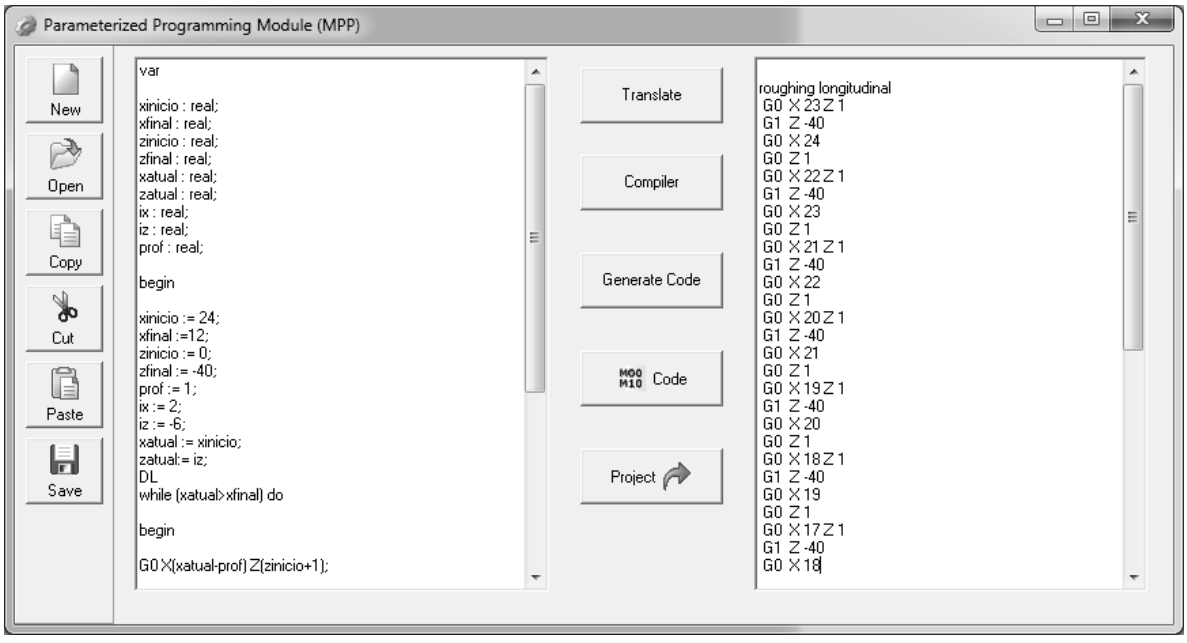

Figure 2. Compilation of MPP

The programming code parameterized after drawn is stored in form of text file. This storage technique is used to transfer data between applications. Figure 3 shows a schematic of the logic employed in the development schedule parametrized module MPP.

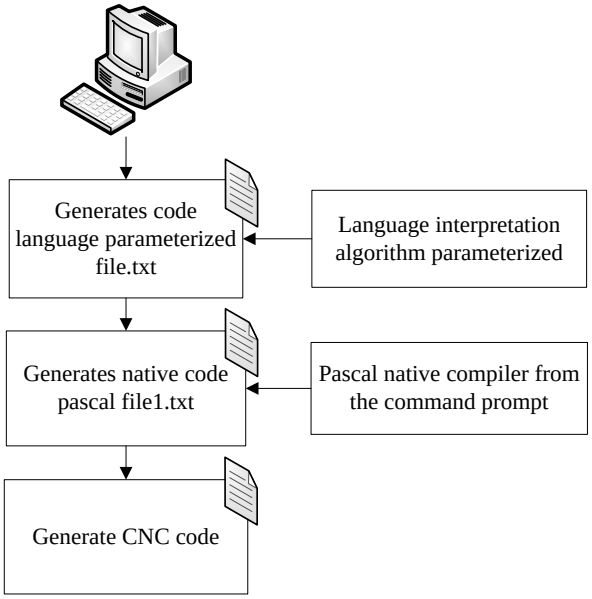

Figure 3. MPP scheme

An algorithm was developed in order to interpret the syntax and semantics of the programming language blocks parameterized proposal. Archiving data in native Pascal compilation unit. Figure 4 shows an excerpt of this algorithm in the programming language Pascal, responsible for interpreting the language proposed.

```
if termo = 'G1' then
     begin
     if (pos('X',texto) <> 0) and  (pos('Z',texto) <> 0)then
        begin
        termo:= copy(texto,(pos('(',texto)),((pos(')',texto))-(pos
('(',texto))+1));
        termo1:= copy(texto,pos('Z',texto),length(texto));
        termo2:= copy(termo1,(pos('(',termo1)),((pos(')',termo1))-(pos
('(',termo1))+1));
       writeln(arq2,'Memo1.Lines.Add('' G1 '' + '' X '' + floattostr' +
termo + ' + '' Z '' + floattostr' + termo2 + ');');
        end
        else
         if (pos('X',texto) <> 0) and  (pos('Z',texto) = 0)then
          begin
          termo:= copy(texto,(pos('(',texto)),((pos(')',texto))-(pos
('(',texto))+1));
          writeln(arq2,'Memo1.Lines.Add('' G1 '' + '' X '' + floattostr'
+ termo + ');');
          end
          else
          begin
          termo:= copy(texto,(pos('(',texto)),((pos(')',texto))-(pos
('(',texto))+1));
          writeln(arq2,'Memo1.Lines.Add('' G1 '' + '' Z '' + floattostr'
+ termo + ');');
          end;
```

Figure 4. Excerpt algorithm interpreter

The result of applying the algorithm interpreter in the file containing the coding language parameterized (file.txt) as Fig. 3, results in a file stored in native Pascal (file1.txt). In this last file is used by the compiler Pascal native prompt system, generating CNC code. Figure 5 illustrates the sequence builds to obtain the final code.

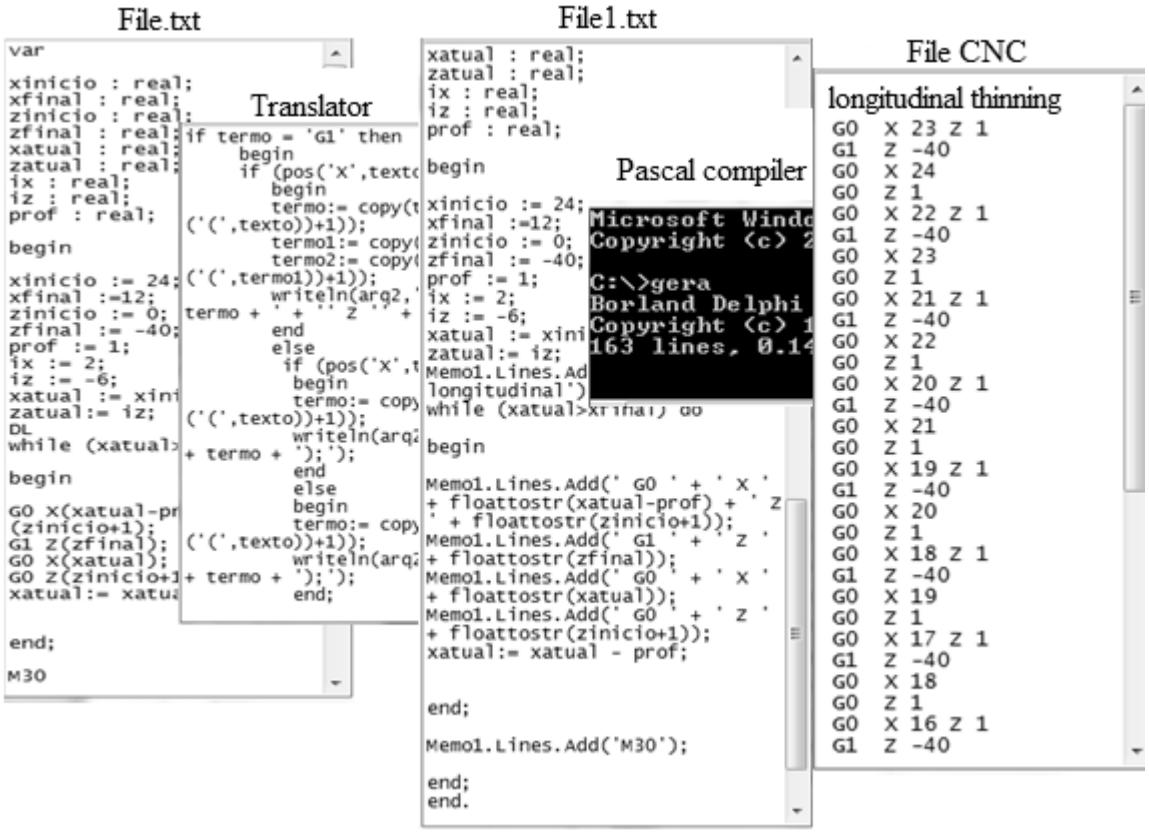

Figure 5. Sequence of phases for CNC file

3. RESULTS AND DISCUSSION

Aiming to demonstrate the proposed methodology presented in this chapter are examples of creating routines machining with language parameterized MPP.

3.1 Cycle for turning

The need for different machining strategies are often, but not always possible, because few possibilities offered by cycles of CNC controls. Figure 6 shows a machining strategy unconventional: thinning at an angle that is not found in the majority commands. Cycles typically used for this procedure are thinning longitudinal and transversal thinning.

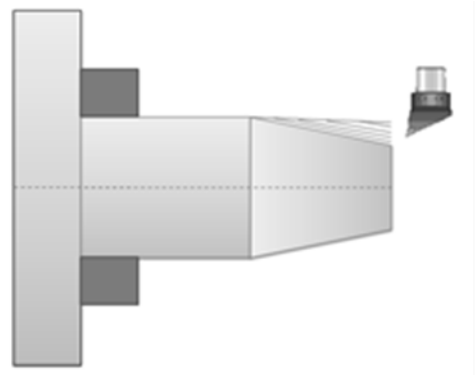

Figure 6. Roughing angle

This type of operation can bring benefits to finishing the workpiece, reducing the variation in cutting force on the removal of metal finishing. Moreover, the reduction of the formation of scaling may be advantageous to minimize machine vibration. Figure 7 shows the result of the thinning longitudinal extension, forming the "rungs", taken at the last pass.

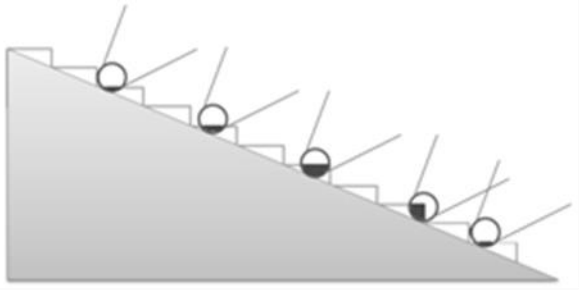

Figure 7. Variation of the contact area of the tool

In preparing the CNC program parametric in order to perform thinning angle is established the necessary variables to the system Tab. 4.

Table 4. Variables for roughing angle

Variables	Definitions
xinicio	Diameter greater part
xfinal	Minor diameter part
zinicio	Initial length
zfinal	Final length
xatual	Counter at x
ix	X-increment
iz	Z-increment
zatual	Counter z

The tool is positioned (ix xatual and zinicio +1) (start point for the displacement of the tool), shown in flowchart form in Fig. 8. The movement is accomplished by determining the angle offset point (xinicio and zatual). Brokers are updated and the tool returns to the starting point and makes the next move.

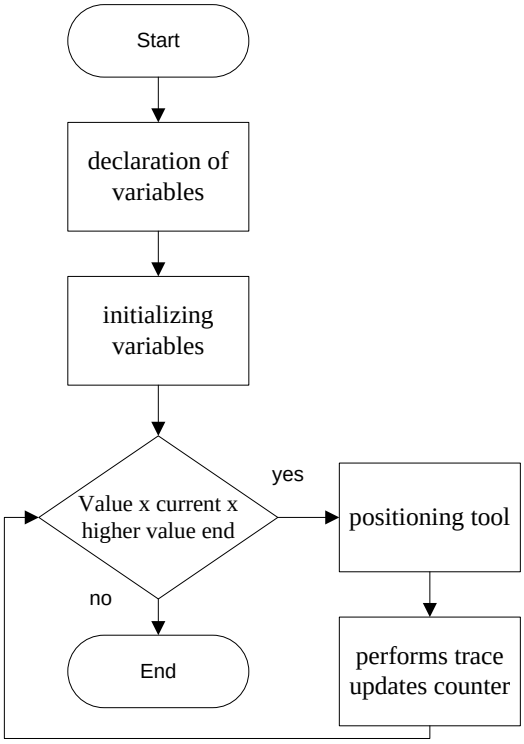

Figure 8. Flowchart thinning angle

The program code developed with the proposed language with comments is illustrated in Fig. 9. Compiling the editor MPP program we get the equivalent CNC interface illustrated in Fig. 10.

```
var

xinicio : real;
xfinal : real;
zinicio : real;                    //Declaration of variables
zfinal : real;
xatual : real;
ix : real;
iz : real;
zatual : real;

begin

xinicio := 24;                     //Initialization of variables
xfinal :=12;
zinicio := 0;
zfinal := 40;
ix := 2;
iz := -6;
xatual := xinicio;
zatual:= iz;

while (xatual>xfinal) do           //repeat until true
                                   condition
begin

G0 X(xatual-ix) Z(zinicio+1);      / / starting position
G1 X(xinicio) Z(zatual);           / / first pass
G0 X(xatual) Z(zinicio +1);        / / starting position
xatual:= xatual - ix;              / / correct counter x
zatual:= zatual + iz;              / / correct counter z

end;


end;
end.
```

Figure 9. Codes and commentary program

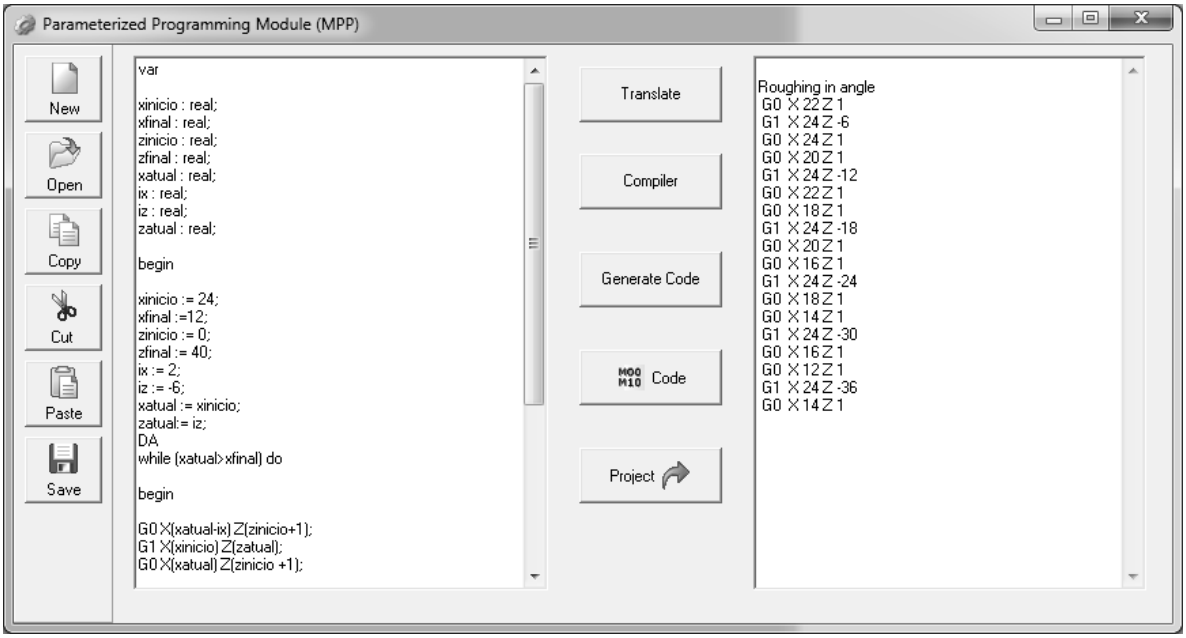

Figure 10. Compilation

The cycles parameterized language developed by MPP can be stored as text files for use by programming editor at the time of program design. Cycles are considered developed additional cycles for machine control, supplementing the resources available by the manufacturer of the command in the preparation of programming. Figure 11 illustrates a scheme for implementing cycles developed in the MPP.

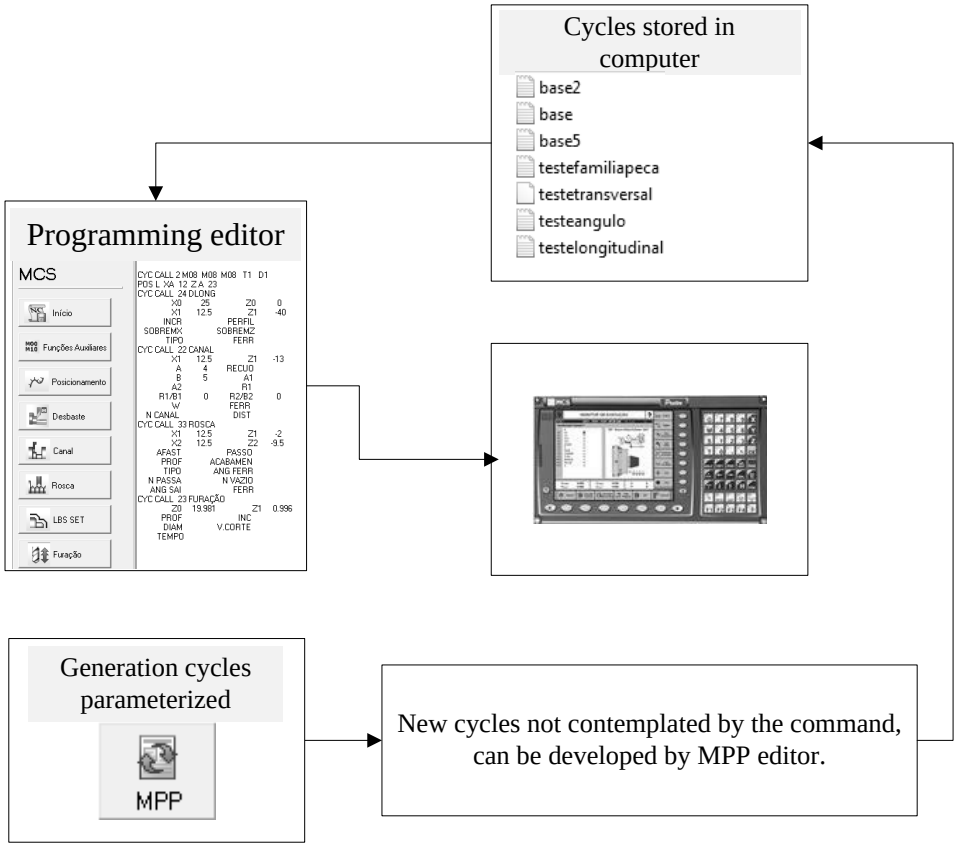

Figure 11. Scheme implementation cycles developed by MPP

With this vision system can also be used in the methodology of group technology, the parametric cycles to perform operations belonging to a family of parts can be stored file groups (Lorini, 1993). When there is the need to run the programming of a part belonging to this group can avail these cycles stored only changing the parameters.

4. CONCLUSIONS

The parameterized programming for CNC program generation module MPP, proposed in this paper, proved effective development cycles parameterized suggested. The CNC code generated by the module is compatible with MPP expected.

The development and use of the editor in future work tends to improve language interpreter suggested and implementation of new functions.

The methodology proposed a universal language parameterized to generate CNC programs cycles influences the programming paradigm parameterized be very difficult because of the need for knowledge of various languages parameterized corresponding to specific commands.

The proposed work is a methodology that can transform language parameterized as another common tool or employed in the production system.

5. REFERENCES

Smid, P., 2003 *Cnc Programming Handbook*: A Comprehensive Guide to Practical Cnc Programming, Industrial Press Inc, 2ª ed, New York.

Lynch, M., 1997. *Parametric programming for computer numerical control machine*, Society of Manufacturing Engineers, Michigan USA.

Gonçalves, M.A.F., 2007. *Um Estudo Sobre Implementação de Ciclos de Usinagem Através de Programação Parametrizada em Máquinas de Comando Numérico Computadorizado*, Dissertação de Mestrado, Universidade Federal de Santa Maria, UFSM, Santa Maria, Brasil.

Lorini, F. J., 1993. *Tecnologia de Grupo e Organização da Manufatura*, Ed. da UFSC, Florianópolis, 105 p, Brasil.

Davim, J. P., 2008. *Manufaturing: Fundamentals and recent advances*, London, Springer, 361 p.

Groover, M., 2011. *Automação Industrial e Sistemas de Manufatura*, Pearson Prentice Hall, 3ª ed, São Paulo, 840 p.

Machado, A. R. e Silva, M. B. e Coelho, R. T. e Abrão, A. M., 2009. *Teoria da Usinagem dos Materiais*, Ed. Edgard Blücher, São Paulo, 384 p.

6. RESPONSIBILITY NOTICE

The authors are the only responsible for the printed material included in this paper.