Published: X Workshop on Supervising and
Diagnostics of Machining Systems.
Innovate and Integrated Manufacturing,
pp 96-102. Karpacz 21st-26th March 1999.

object oriented, NC programming, simulation

Oliver HORNYÁK *

OBJECT ORIENTED SOFTWARE ENGINEERING FOR SIMULATION OF NC MACHINING OPERATIONS

Abstract: The growing application area of NC programming systems has increased the importance of computerised simulator offering NC program verification. This paper deals with the NC programming generation and verification processes. It describes the main levels of simulation tasks. The syntactical verification is based on the lexical description of the NC program language used. During semantic analysis certain tool path degeneration can be detected. Recognition of undercut and collision events also requires investigation and evaluation of the geometrical model of the part and tools. Tool life simulation demands a tool database supporting one of the most fundamental activities of monitoring the cutting to make a decision on the tool change. The expected tool life is a stochastic variable function of cutting intensity. In non-stationary case, real time measurement or simulation is required. This requirement necessitates a new generation of simulators. To satisfy complex technological demands (for example time, quality and cost engineering) there are no appropriate simulators therefore new simulators have to be developed with higher performance and capability. The second part of this paper focuses on the object oriented modelling and simulation architecture for simulation of NC machining operation. In this area some new results will be presented.

1. INTRUDUCTION

In course of generating the NC part program setting out from the geometry of part it is to be done the program file interpreting by the given NC controller. There are some general methods of NC programming, they are as follows:

- in manual way,
- by APT based systems for general purpose,
- by integrated NC program development systems,
- by interactive WOP graphical system, built into a CNC device.

Integrated NC program developing system can be defined generally as a system suitable for generating NC program for a specified NC controller. It is not a processor-post processor based system. Because of the continuous growing of process complexity, negative consequences of program errors, e.g. damage of worker, machine, workpiece and tool have become extremely serious. Therefore the demand on computerised simulator offering NC program verification became more and more important.

* PhD Student, University of Miskolc, Institute of Information Science, Department of Information Engineering

The main levels of simulation tasks are as follows:

(1) Syntactic verification
(2) Semantic analysis
(3) Geometrical undercut and collusion test
(4) Tool life simulation
(5) Complex technological simulation.

Most of the up-to-date NC programming systems involve some kind of built-in simulation. The input of these simulation is CL Data file or other data formatting file representing geometrical and technological information. In compliance with it the available services are limited, mainly to the semantic analysis and geometrical undercut and collision test. The syntactic verification has no importance in case of the built-in simulator. The occurrence of syntactical error is prohibited in a computer generated CL Data file.
There are stand-alone simulators to manipulating post-processed NC program in ISO format. Syntactical verification based on the lexical description of NC language. It has significance if NC program has been written by hand.

A syntactically correct part program may contain further semantic errors which must be detected. These types are:

- collision with rapid movement,
- fault of part dimensions,
- intruding a protected area,
- fault on tool path generation.

Recognition of undercut and collision events requires investigation of the geometrical model of the part and tools too. By means of tool path getting from the NC part program considering the tool corrections geometrical model of the finished surfaces can be recovered. This has to be compared with the part geometry. Tool life simulation demands a tool database supporting:

- tool change and analysis of cutting conditions,
- geometrical tool corrections,
- forecast of tool life.

2. SOME NEW TASKS FOR SIMULATION

The latest generation of simulators comprises estimating of tool wears which is the most fundamental activity of machine monitoring systems. The expected life is a stochastic variable, therefore machine tools must be equipped with monitoring system which can conclude the necessity of tool change from measuring main motor current, vibration, force, wear, acoustic emission, etc. Tool life is a stochastic variable function of *cutting intensity* that is also named the *ratio of stock-removal* in special literature. Research work of the late years carried-out at the University of Miskolc has shown the fact that the two main state

variables namely *tool life* and *cutting time* are well manageable functions of cutting intensity:

$$T = T(Q)$$

$$t_m = \frac{V}{Q}$$

where

T = tool life [min],

$Q = d \cdot f \cdot v$ = cutting intensity [cm^3/min],

d = the average depth of cut [mm],

f = feed rate [mm/rev],

v = the average cutting speed [m/min],

V = the allowance volume to be removed [mm^3].

(Remark: Dimensions are given as it is usual and accepted in cutting technology both from theoretical and practical points of view).

Assuming that Q is a quasi-constant (e.g. stationary cutting is boring), tool life could be well-predicted. In non-stationary case, real time measurement or simulation is required.

In case of integrated process planning and scheduling, one of the most important integrating factors is connection between *make-span* and *operation times*. A new area of pragmatic verification of NC part program is the analysis of operation times on cutting conditions. To achieve the objectives (goals) of production the operation times must be known before the start of scheduling. Computing of the operation time is the basic service of the modern simulators. The operation time is a function of the tool path generation method. By means of simulation some different versions can be tested, and the optimal method can be selected. Considering the term of average or extreme program running conditions the technology bottlenecks of the manufacturing process can be identified. At the bottlenecks, the cutting intensity must be increased. The operation time is the sum of machining (or: main) time and auxiliary times:

$$t_{op} = t_m + \sum_i t_{a,i} \, .$$

The effect of the NC override of cutting is that the operation cost change in a non-linear way. Because of the increased cutting intensity, the tool life decrease. It causes more frequent tool or cutting edge change. Balancing the machining and tool cost is an optimal finding problem. My research topic is to develop a new NC program simulator which can support process planning and scheduling optimization.

The complex technological simulation includes the simulation of quality assurance and quality control as well. The major factors affecting quality are as follows:

- force
- vibration
- quantity of heat
- tolerances
- deviations
- surface roughness.

All of these factors can retraceable to elementary volumes and integration of these. The simulation of quality assurance can be executed on this. This requirement introduces a new generation of simulators.

An inverse approach is to use NC part program simulation when geometrical and technological information are gained from archive NC part program. The purpose of this can be:

- verification of the correctness of a part program,
- modification of the existing part program generating a new version,
- detection of collision, break, waste, etc.

In this case simulator supports the shopfloor or machining cell dispatching in direct way.

3. SOFTWARE TECHNOLOGY FOR ADVANCED SIMULATION

Development of a new simulator software is a complex problem. My goal has been to develop a premium quality software. The major quality criteria are:

- correctness
- user-friendly behaviour
- robustness
- compatibility
- efficiency
- standardization
- maintainability, serviceability
- upgradeablility, extensibility
- controllability
- portability.

Considering the quality assurance of software products ISO 9000-3 has been referred. This standard does not specify the applied development methods. Nevertheless, it puts to use well defined, manageable methods down.

My choice was the object oriented method. Simulators transform the real world's object to the plane of Information Technology. All the areas of technical life can be treated in an object oriented way with adequate decomposition. We can say Object Oriented Programming have became a paradigm. This acceptance is reflected in the popularity of object oriented programming languages such as Ada, Smalltalk, Java and C++.

Systems constructed in object oriented way are build from cooperating objects instead of reducing these objects to a procedural set of steps. The object oriented paradigm expounds three major ideas that are necessary for an object oriented programming language to support:

- *Encapsulation* - the hiding of an object's implementation details
- *Inheritance* - the ability to reuse existing class in the creation of new, more specialized objects
- *Polymorphism* - the ability of code to exhibit multiple behaviours depending on the object being used.

The encapsulation makes the ability to decompose the system into cooperating object. The object internal implementation, which is code and data is nobody's business but its

own. The object is a sort of „black box”. The object has a public interface through which the data can be accessed. This interface constitutes the contract between the object and its users.

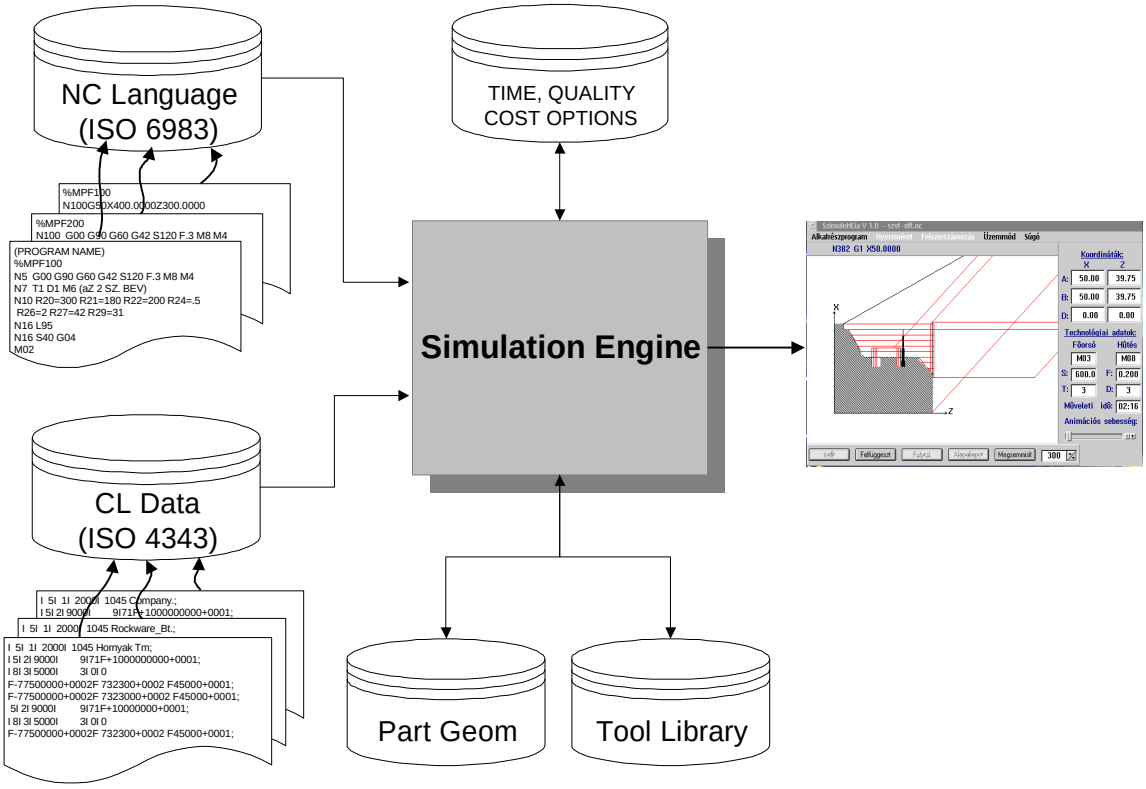

Fig. 1. Simulator for graphical verifications of NC part program

The most important step of the object oriented design is to build-up the object hierarchy. The nature of simulation problems is that the processes temporal proceeding is examined. From a software technological point of view simulators are aggregations of static and dynamic objects. Such a static object is the *simulator* object. This is the central object of the simulation. It contacts the auxiliary objects for example the *user interface* object, and manage the graphical animation.

Other object can arise only dynamically, in runtime. The object of simulation should only be dynamic objects. During the object oriented design I have made up a hierarchy of class to decompose the NC program into sequence of *entities*. Such entities are the movement cycles (fast feed, linear interpolation, circular interpolation), coolant, spindle on/off and even the end of the sentence sign. The entities visualization one after the other results the graphical animation. The ability of object's polymorphism is observable in this example. The visualization method of the object *coolantOn* different to the object *linearInterpolation*. The main purpose of the graphical animation is semantic analysis. The use of the object oriented technology makes it possible to imply additional simulation levels without modifying the code. Unvarying existing interfaces of the existing, well written objects the software can be extended adding further interfaces and components.

A significant problem of simulator software is graphical environment. Developing a new standalone graphic library is a notable task. It can not be a part of development of the new simulator software. The basic graphic libraries only supports elementary 2D objects. This is the code reusing. Performance of the modern computers permits the use of real time 3D animation. The OpenGL graphical library regards as a quasi standard in this area. It is accessible to most of the operating systems. To improve performance, some of the video accelerators support hardware OpenGL acceleration. An other possible and popular method is using the Virtual Reality. The Virtual Reality Modeling Language is standardized, platform-independent and suitable for network.

In the course of developing a simulator software the network environment must be examined. The access of tools database and download of the NC programs from the DNC server are through network in the real machine tools environment. The simulator must be equipped with the ability of sending and receiving standard manufacturing messages, accessing databases, communicating with other computers.

4. CONCLUSIONS

Requirements set against the simulators have recently been increasing. The simulators having multiple services support multiple CAxx functions, for example CAPP, NCP, PPS, etc. Developing a complex simulator requires object oriented technology.

5. ACKNOWLEDGEMENTS

The scientific investigation results obtained so far and summarised in this paper are belonging to the PhD (Doctoral) Programme of the University of Miskolc, Subprogramme Production Systems and Information Science. The author is indebted to the Academic Staff of the Department of Information Engineering, especially to *Dr. Ferenc Erdélyi* and to *Prof. Tibor Tóth* for their advice, encouragement and useful discussion.

REFERENCES

[1] AVGOUSTINOV N.: *Virtual Shaping and Virtual Verification of NC-Programs (Simulation of Part Manufacturing by Means of NC Machines in Virtual Reality)*

[2] EDDON G., EDDON H.: *Inside Distributed COM* pp. 11-18.

[3] HORNYÁK, O.: *NC program ellenõrzõ szimulátor fejlesztése*, 1997

[4] TÓTH T., ERDÉLYI F.: *The Role of Optimization and Robustness in Planning and Control of Discrete Manufacturing Process.* The Second World Congress in Intelligent Manufacturing. CIRP Budapest, Hungary, 1997., pp. 205-210.