International Journal of Modern Manufacturing Technologies
ISSN 2067–3604, Special Issue, Vol. XIII, No. 3 / 2021
https://doi.org/10.54684/ijmmt.2021.13.3.177

CAD-BASED AUTOMATED G-CODE GENERATION FOR DRILLING OPERATIONS

Anastasios Tzotzis, Athanasios Manavis, Nikolaos Efkolidis, Panagiotis Kyratsis

University of Western Macedonia, Department of Product and Systems Design Engineering, Kila Kozani, GR50100, Greece

Corresponding author: Anastasios Tzotzis, tzotzis.tasos@gmail.com

Abstract: The automated generation of G-code for machining processes is a valuable tool at the hands of every engineer and machinist. Nowadays, many software systems exist that provide automated functions related to G-code generation. Most free software require the import of a Drawing Exchange Format (DXF) file and cannot work directly on a 3D part. On the contrast, the equivalent commercially-available software systems are feature-rich and can provide a variety of automated processes, but are usually highly priced. The presented application aims to supplement the existing free Computer Aided Manufacturing (CAM) systems by providing a way of generating G-code for drilling operations, using already owned commercial 3D Computer Aided Design (CAD) systems such as SolidWorks™. Thus, in the case of 3D part drilling, a standard 3D CAD system is sufficient since the code can be adopted by most modern CAD software with minor changes. Moreover, no specialized CAM software is required. In order to achieve this automation, the Application Programming Interface (API) of SolidWorks™ 2018 was utilized, which allows for the design of visualized User Interfaces (UI) and the development of code under the Visual Basic for Applications (VBA™) programming language. The available API methods are employed to recognize the features that were used to design the part, as well as extract the geometric parameters of each of these features. In addition, an embedded calculator automatically defines the cutting conditions (cutting speed, feed and tool) based on the user selections. Finally, the application generates the Computer Numerical Control (CNC) code for the summary of the discovered holes according to the standardized G-code commands; the output can be a typical TXT or NC file that can be easily converted to the preference of the user if necessary.

Key words: CAD, Application Programming Interface, VBA, Drilling, G-code

1. INTRODUCTION

Nowadays, modern Computer Aided Design (CAD) systems include advanced features and capabilities that can be used to facilitate procedures and tasks that otherwise would require dedicated software systems in order to be carried out. Example tasks are the generation of the G-code for machining, as well as the simulation of the tool path. Implementation of the Application Programming Interface (API) of CAD systems allows for the development of macros and applications that can fulfill the aforementioned roles. Moreover, the API provides the user with tools that can lead to efficient working schemas. In the past years, researchers have utilized CAD systems to perform various tasks without the need of specialized software and to carry out investigations on multiple areas with the aid of CAD-based strategies and techniques. Indicative applications of the CAD-based programming include the automated operation of various design procedures, which in most cases can be embedded into a User Interface (UI), the interaction between different software modules and programmable machines, as well as the simulation of engineering problems.

Vijayaraghavan and Dornfeld [1] presented a method to create accurate models of two-flute standard drills using solid-modeling techniques and Boolean operations to mimic the manufacturing process of drills. The accuracy of the drills generated by this method have been used in Finite Element (FE) simulations. Similarly, Tzotzis et al. [2] described an application that can be used to generate CAD models of standardized turning inserts. The development was

achieved with the aid of the SolidWorks™ API and the Visual Basic for Applications (VBA™) programming language, by utilizing a parametric-based design technique. Moreover, the generated tools were easily converted to FE-ready file formats. By using similar strategies, Kim et al. [3] proposed a design process for end mills. The design procedure is based on the solid model of the designed cutter along with the computation of the cutter's geometry, wheel geometry, and wheel positioning data. The cutting simulation method is used to obtain the machined shape of an end mill by using Boolean operations between the grinding wheel and the cylindrical workpiece. CAD-assisted approaches were employed by Kyratsis et al. [4] in order to build simple and easy to use tools, that can eliminate repeatable and time-consuming design tasks. Specifically, their work deals with the development of an applet intended for the automated design process of a bicycle. Works that have implemented similar methodologies and techniques, related to the automated design process of products and systems, are available in the literature [5–7]. Oancea and Haba [8] presented a new software tool which allows the user to obtain multiple information such as the manufacturing sequences, cutting conditions for each process from the manufacturing sequence and finally to assist the simulation procedures. Viganò and Osorio-Gómez [9] defined an approach to extract the liaison graph from a 3D CAD model and analyze a method to find the feasible assembly sequences during the stage of the design process of a product based on attributes and parameters of the graph. The employment of CAD-assisted approaches was used for measuring simulations as well in the work of García-Hernández et al. [10]. Authors presented the geometrical and mathematical principles necessary to develop a software application for measuring gears, by utilizing spreadsheet applications and CAD-based simulations of the measuring operation. The aforementioned technique allows for the calculation of the probe positions, as well as the direction of measurement. Another topic where CAD programming resources have been successfully used is the measurement of the produced machining forces and other machining parameters such as surface roughness [11–13]. Tapoglou and Antoniadis [14] presented the HOB3D code, intended for simulating the complex movements involved in gear hobbing. The simulation is achieved by embedding the developed algorithm in a commercially-available CAD system environment. According to authors, the code can calculate and export the total generated machining forces as well as the cutting forces in every cutting edge. In a similar manner, Kyratsis et al. [15] presented the DRILL3D tool, which can be used to calculate the thrust force of both the cutting areas of the tool simultaneously.

The present paper proposes an alternative way of generating CNC codes for drilling, based on CAD system programming. The aim of the present research is to develop an application with features found on CAM systems, with the aid of a commercially-available CAD software.

2. DEVELOPMENT OF THE APPLICATION

2.1 User Interface design

The User Interface (UI) was designed with simplicity and ease-of-use in mind. In order to achieve the aforementioned traits, the embedded toolbox of SolidWorks™ 2018 API was utilized. The form is divided into two sections; the first one "Cutting Conditions" contains two textboxes that allow the input of the cutting speed and feed value respectively. The measurement units for the cutting speed and feed are m/min and mm/rev accordingly. In addition, one listbox was added that acts as a pull-down menu for the user to select a material group. A colored text informs the user for each one of the material groups based on the ISO. For example, blue color (ISO P) represents the Steel group of materials. Specifically, the ISO P group includes non-alloy steels, low-alloy steels, high-alloy steels, cast steels and stainless steels (ferritic/martensitic). The cutting conditions section includes a checkbox also, which when checked enables the automated selection of the cutting conditions, bypassing the default or any user-selected values. This is achieved by scanning for the material of the part that is open in SolidWorks™ and checking the material group in which it belongs. The cutting conditions then are automatically set according to standardized values that derive from well-known cutting-tool manufacturers' recommendations.

Figure 1 illustrates the UI of the application along with the input fields and command buttons.

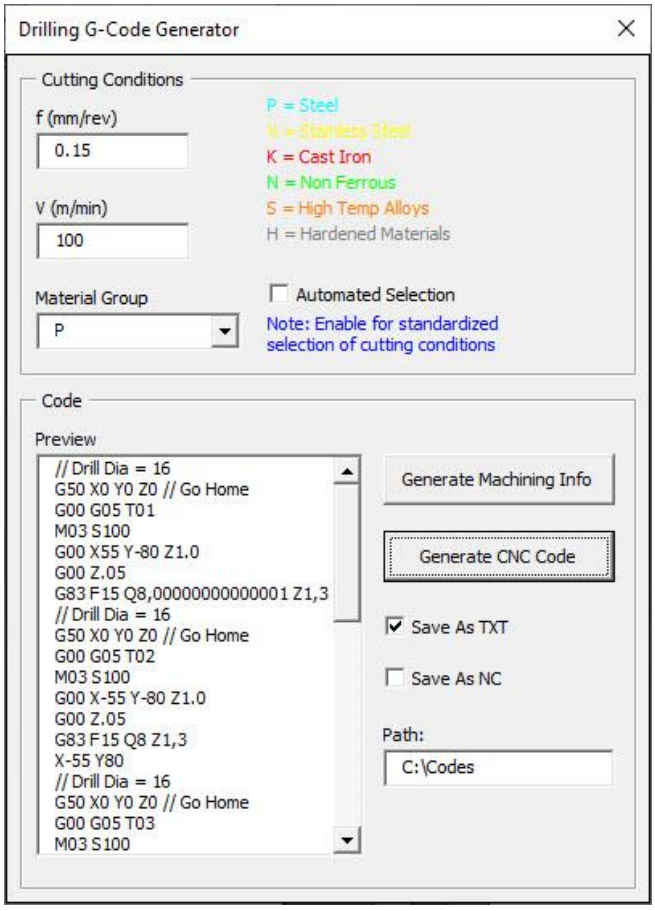

Fig. 1. User Interface (UI) of the developed application

The second section of the application, namely Code, is used for the generation of the drilling G-code. This section contains one textbox and one listbox, the listbox is intended for the code preview, whereas the textbox is used for the input of the code file saving path. The code preview windows provides a first glance at the code. Two additional checkboxes are used to allow the user to select whether the file is saved as TXT, NC file format or both. Either way, the files are editable and changes can be made if necessary. Finally, two command buttons are included for the machining information generation and the code generation respectively. When the first button "Generate Machining Info" is pressed, a procedure that scans for hole features initiates and creates attributes on each of the found holes. The second one "Generate CNC Code", is used to generate the G-code based on the found holes and their attributes, as well as the set cutting conditions.

2.2 Workflow of the application

The code of the application is divided into two parts; creation of the attribute definitions and generation of the G-code. The complete process is illustrated in Figure 2 with the form of a workflow. The first part begins with the declaration of the necessary variables such as the cutting speed, the feed, the hole and drill diameters, the hole depth, the workpiece material group, the attribute parameters and finally the topology objects. Next, these variables are linked with the process options that can be accessed by the user via the UI. For example, the textbox entitled "f", which is used by the user to input the feed value, is linked to the corresponding variable, namely feed, so that the process can take place. The next step is the access of the open document. In the case that no part document is open, a message window appears prompting the user to open one. Part documents must contain holes generated either by a dedicated hole feature or by an extruded cut feature. Upon accessing the part document, the creation of the attribute definitions is realized for the next parameters: cutting speed, feed, position in X axis, position in Y axis, position in Z axis, hole depth and hole diameter. Consequently, the traversal of the solid bodies that are available in an open document is performed, in order to find the cylindrical surfaces of the holes. Next, the information of each cylindrical surface found is assigned to an attribute and then an instance of each attribute is being created. In addition, the parameter values for each attribute are set and an equivalent callout for each attribute is created, so that can be displayed as soon as the "Generate Machining Info" button is pressed. The first six steps of the workflow comprise the first part of the code.

The second part of the code begins with a feature traversal on the part [16], searching for the defined attributes. Next, the parameter values (cutting speed, feed, position in X axis, position in Y axis, position in Z axis, hole depth and hole diameter) of the found attributes are extracted. Finally, the CNC code is formatted and added to the listbox for review. Moreover, it is generated according to the used cutting conditions and stored as TXT or NC file depending on the user selection. The CNC code can be generated with the "Generate CNC Code" button. Besides the generation and storage of the CNC code, the listbox intended for the preview of the code is also filled by pressing the "Generate CNC Code" button. For this purpose, the appropriate G and M commands are used, such as the "G00" and the "M30" commands, which are required for the rapid positioning of the tool and for the end of the program respectively. The last three steps are the ones that comprise the second part of the code.

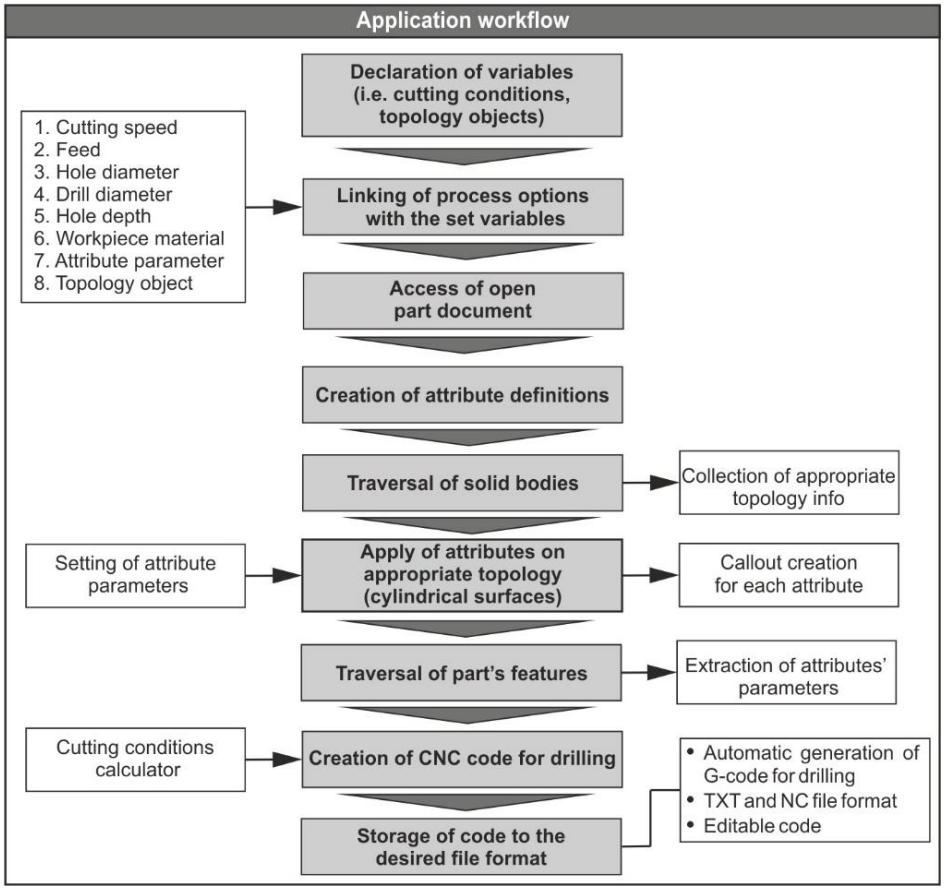

Fig. 2. Workflow of the application

2.3 Coding procedure

The code of the application is divided into two sub procedures according to the functions of the application [17,18]. Each of these procedures is called as soon as the corresponding button ("Generate Machining Info" and "Generate CNC Code") is pressed in the UI. The procedures are structured according to the syntax norms of the VBA™ programming language. Moreover, the appropriate methods of SolidWorks™ API are used. While the "DefineAttribute" method is used to create an attribute definition, the "AddParameter" method is used to add the feed, the cutting speed, the X position, the Y position and the Z position of the hole center, as well as the depth and the hole diameter to the created attribute definitions. In order to traverse the solid bodies that are available in the part document, a "For" loop is utilized. At first, a collection is created that can store the found cylindrical surfaces of the part. The "GetBodies2" method is used to get the bodies that are available in the part and the traversal of the bodies is carried out with the aid of the loop. The "GetFirstFace" method is called with the execution of the loop so that the first face of the solid body can be found. Next, the "GetSurface" method is required to get the surface in the case it is a cylinder, whereas the "CylinderParams" is used to gather the information of the current surface. Upon finalizing the information gathering process, the "CreateInstance5" method is utilized to create an instance of the attribute described in the userform's activate handler. In addition, to set each parameter value of the attribute using the information from the cylindrical surface, the "GetParameter" method is required, whereas to set the values of the parameters the "SetDoubleValue2". Consequently, to display to the user which faces got attributes, the "CreateCallout2" method is used. Thus, when pressing the "Generate Machining Info" button, the according callout objects are created on the part with the aid of the "CreateSelectData" method. Finally, to allow the code to be executed for all other faces that are available on the model, the "GetNextFace" method is employed, which gets the next face in the solid body. Most of the SolidWorks™ API methods that were used for the first part of the code and their function are included in Table 1.

Table 1. Basic API methods for the solid body traversal and definition of the attributes

Method	**Function**
DefineAttribute	Creates an attribute definition
AddParameter	Adds a parameter to the attribute definition using the specified name and default value
GetBodies2	Gets the bodies in this part
GetFirstFace	Finds the first face in a body and returns the face
GetSurface	Gets the surface referenced by this face
CylinderParams	Gets the parameters of a cylindrical surface
CreateInstance5	Creates an instance of this attribute on the specified object with the specified options
GetParameter	Gets the specified parameter on this attribute
SetDoubleValue2	Sets the double or integer value of a named configuration option parameter
CreateCallout2	Creates a callout for the selection
CreateSelectData	Creates a ISelectData object to use as argument with Select
GetNextFace	Gets the next face in a body

The second part of the code does a feature traversal and searches for all the attributes on the open model. Once an attribute is found, its values are obtained and then a CNC drilling program is formatted according to the data stored in the attributes parameters. For the feature traversal, a "For" loop is utilized as well. The "FirstFeature" method is the one that is responsible to get the first feature that is available in the model, whereas the "GetSpecificFeature2" method is used to get the interface of the current feature as long as it is an attribute. The "GetParameter" method is used in a similar manner as in the first part of the code. The purpose of this method is to get the specified parameters (feed, cutting speed, X position, Y position, Z position, hole depth and diameter) of an attribute. To fill the lines of the preview listbox with the G-code and to get an attribute value of type Double, the "AddItem" and the "GetDoubleValue" methods are used respectively. Finally, with the "GetNextFeature" method it is possible to get the next feature in the part document, so that it can be checked whether it is an attribute or not. In the case that the user enables the automatic selection of the cutting conditions, the "GetMaterialPropertyName2" method must be utilized in order to obtain the material name that is applied on the workpiece part. To realize the automatic process of the cutting conditions selection is realized via a control structure with the "Select Case" statement, which is discussed in the next paragraph. Table 2 contains the aforementioned API methods used for the realization of the feature traversal and the CNC code generation.

Table 2. Basic API methods for the feature traversal and the G-code generation

Method	**Function**
FirstFeature	Gets the first feature in the document
GetSpecificFeature2	Gets the interface for this feature
GetMaterialPropertyName2	Gets the names of the material database and the material for the specified configuration
AddItem	Adds the specified advanced component selection criterion to
GetDoubleValue	Gets an attribute value of type double
GetNextFeature	Gets the next feature in the part

To enable the automatic selection of the cutting conditions, user can check the corresponding checkbox that is intended for this purpose. Once the CNC code generation process begins with this function enabled, the feed and the cutting conditions are selected based on the material of the workpiece model and according

to the recommendations of well-known tool manufacturers that are available in their catalogues. To realize this process, the algorithm that is illustrated in Figure 3 is utilized. In the case that the user chooses to use custom values for feed and cutting speed, the corresponding textboxes are used; these are the "f.TextBox" and the "V.TextBox" respectively. In contrast, when the "CB.CheckBox" value is set to true, the "GetMaterialPropertyName2" API method is utilized in order to obtain the material that is applied to the workpiece. Consequently, a control structure is performed, checking the material group in which the obtained material of the workpiece belongs. According to the catalogues, six groups exist; the ISO P, the ISO M, the ISO K, the ISO N, the ISO S and the ISO H that represent Steels, Stainless Steels, Cast Irons, Non-Ferrous materials, High Temperature alloys and Hardened materials respectively. The aforementioned structure is carried out with the "Select Case" statement, which runs one of multiple groups of statements, depending on the value of an expression. In this case, depending on the material group that the obtained workpiece material belongs, the values of the variables that represent feed and cutting speed change according to the set values of feed and cutting speed for that material group. The material groups along with their materials are stacked into an array, namely "mat". Due to the large number of material groups and materials combinations, the use of arrays is imperative.

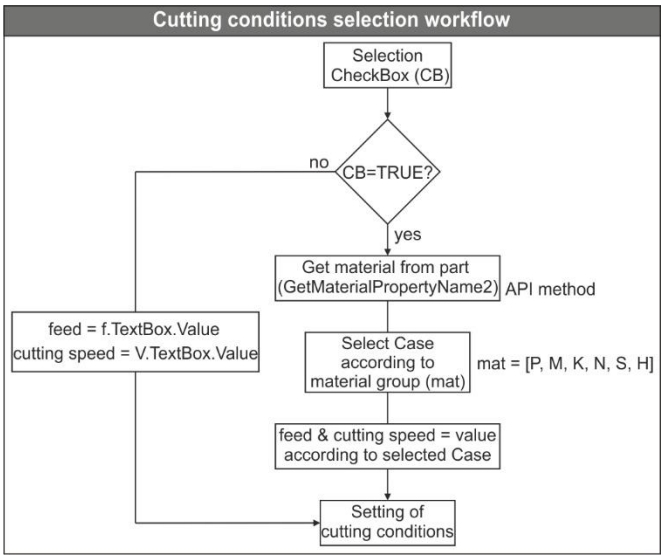

Fig. 3. Selection algorithm for cutting conditions

2.4 Testing procedure

The full process for a sample workpiece is depicted in Figure 4. First of all, a part document must be open in the environment of SolidWorks™, which must be parametrically designed. It is possible to use models that are designed with another CAD system besides SolidWorks™, as long as the models are saved in file formats that retain the used features and can be recognized during the feature recognition process without faults. Usually, hole features are easily recognized since are common features and do not have any particular complexity. Example file formats are the vendor-neutral Initial Graphics Exchange Specification (IGES) and the native for Parasolid geometrical kernel. Furthermore, the model should be designed in such a way so that the upper face of the workpiece is the default Front Plane in SolidWorks™. With this design, the direction of each hole's depth matches the direction of the workpiece's extrusion. Additionally, Z axis becomes collinear to the axis that passes through the center of each hole. Figure 4(a) illustrates a sample workpiece that was designed in SolidWorks™ and is fully parametric. By pressing the "Generate Machining Info" button, the attribute definitions are set with the procedure that is presented in section 2.3 and the equivalent callouts are created for the user to take notice. Figure 4(b) illustrates the created callouts for the sample model. It is noted that the code searches for all cylindrical surfaces that are available in the model. Next, it picks the ones that are related to a hole feature or an extruded-cut feature and finally, creates the callouts. User may use the callouts to preview the positioning of the holes on a face, as well as to check their depth and diameter. Moreover, the application is able to work with cylindrical surfaces that exist on side faces of the workpiece also. This means that the generated G-code can be used with robotic drilling systems with minor changes in the format of the code. Next, by pressing the "Generate CNC Code" button (Figure 1), the full G-code for the drilling process of the found holes is formatted and generated according to the selected feed and cutting speed, in addition to the parameters obtained from the model. Figure 4(c) shows the preview window where the code is listed. This way, user can preview the code and check for any obvious faults. In order to further verify the code and check its integrity, user can copy and paste the code to a path simulator. Since some path simulators are available for use online and without charge, it is easy and fast to perform the code verification. Finally, Figure 4(d) illustrates the saved text file that contains the G-code for the specific workpiece.

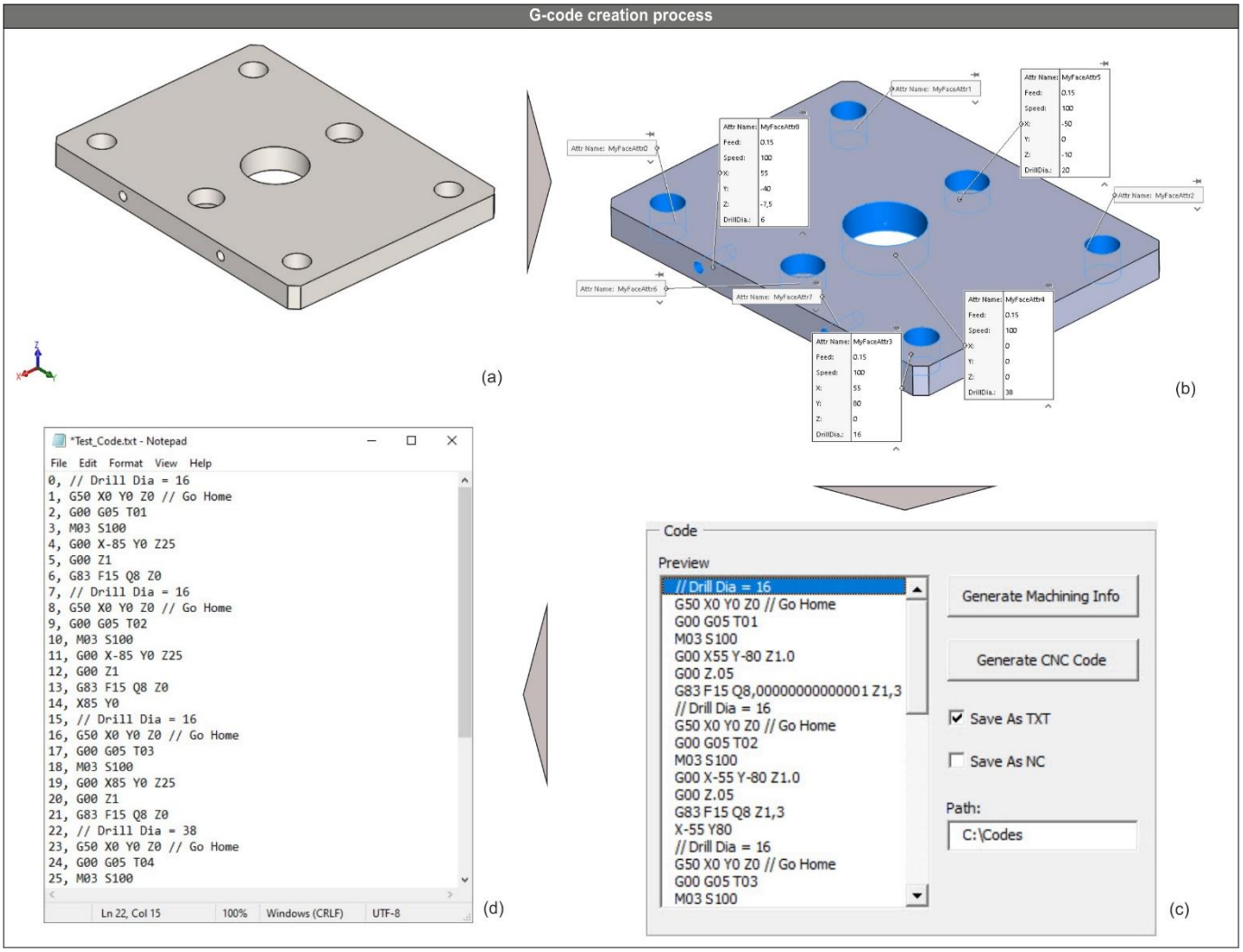

Fig. 4. Example G-code creation for drilling of steel plate

3. CONCLUSIONS

In the present work, the development of a CAD-based application is being demonstrated for the automated CNC code generation for drilling processes, which was realized with the implementation of the API and the programming resources of SolidWorks™ CAD system. The application can be divided into two sections; one that is related to the automated creation of attributes, resulting in the generation of the machining information and another that deals with the automated formatting and generation of editable drilling G-code, optimized for CNC machines. The realization of both parts is based on the traversal of the solid bodies that are available in the part document by means of API programming. The resources used in this research are the SolidWorks™ API and the VBA™ programming language, which allow for the efficient design of customized applications, without the need for installing specialized software packages.

Upon finalizing and testing the application, the following conclusions can be deducted:

The developed application provides an alternative way to generate CNC codes for drilling via a commercially-available CAD system, without the need of costly dedicated CAM software. It is possible to generate the G-code directly from a 3D part without the need to convert the model to 2D. The generated code is editable and can be imported to most freeware or commercially-available CAM systems for advanced editing. Moreover, it is possible to import the codes to simulation software for verification purposes. Finally, it is possible to extend the application, enabling it to generate CNC codes for more machining processes such as milling or even robotic-assisted manufacturing.

Furthermore, it is highlighted that the programming of CAD systems allows for the development of simple macros to complete applications. Moreover, most of the standard CAD-based tasks can be automated, which can lead to reduced development times and production costs of products and systems, hence engineers and machinists can increase their productivity and allocate their time more efficiently.

4. REFERENCES

1. Vijayaraghavan, A.; Dornfeld, D.A., (2007), *Automated Drill Modeling for Drilling Process Simulation.* J. Comput. Inf. Sci. Eng., **7**, 276–282, doi:10.1115/1.2768091.
2. Tzotzis, A.; Garcia-Hernandez, C.; Talón, J.L.H.; Kyratsis, P. (2020), *CAD-Based Automated Design of FEA-Ready Cutting Tools*, J. Manuf. Mater. Process., *4*, 1–14, doi.org/10.3390/jmmp4040104.
3. Kim, J.; Park, J.; Ko, T.J.(2008), *End mill design and machining via cutting simulation.* Comput. Des., **40**, 324–333, doi:10.1016/j.cad.2007.11.005.
4. Kyratsis, P.; Gabis, E.; Tzotzis, A.; Tzetzis, D.; Kakoulis, K. (2019), *CAD based product design: A case study*. *I*nt. J. Mod. Manuf. Technol., **11**, 88–93.
5. Kyratsis, P.; Tzotzis, A.; Tzetzis, D.; Sapidis, N. (2018), *Pneumatic cylinder design using cad-based programming.* Acad. J. Manuf. Eng., **16**, 107–113.
6. Mok, H.-S.; Kim, C.-H.; Kim, C.-B. (2011), *Automation of mold designs with the reuse of standard parts*. Expert Syst. Appl., **38**, 12537–12547, doi:https://doi.org/10.1016/j.eswa.2011.04.040.
7. Tzivelekis, C.A.; Yiotis, L.S.; Fountas, N.A.; Krimpenis, A.A. (2015), *Parametrically automated 3D design and manufacturing for spiral-type free-form models in an interactive CAD/CAM environment.* Int. J. Interact. Des. Manuf., **11**, 223–232, doi:10.1007/s12008-015-0261-8.
8. Oancea, G.; Haba, S.-A. (2016), *Software Tool Used in CAPP/CAM Systems for Rotational Parts*. Sci. Bull. Ser. C Fascicle Mech. Tribol. Mach. Manuf. Technol., **30**.
9. Roberto, V.; Osorio-Gómez, G. (2012), *Assembly planning with automated retrieval of assembly sequences from CAD model information.* Assem. Autom., **32**, 347–360, doi:10.1108/01445151211262410.
10. García-Hernández, C.; Gella-Marín, R.; Huertas-Talón, J.L.; Berges-Muro, L. (2016), *Algorithm for measuring gears implemented with general-purpose spreadsheet software.* Measurement, **85**, 1–12, doi:https://doi.org/10.1016/j.measurement.2016.02.013.
11. Wang, L.; Chen, Z.C. (2014), *A new CAD/CAM/CAE integration approach to predicting tool deflection of end mills*. Int. J. Adv. Manuf. Technol., **72**, 1677–1686, doi:10.1007/s00170-014-5760-4.
12. Vakondios, D.G.; Kyratsis, P. (2020), *An innovative CAD - based simulation of ball - end milling in microscale*. Adv. Comput. Des., **5**, 13–34, doi:10.12989/acd.2020.5.1.013.
13. Dimitriou, V.; Vidakis, N.; Antoniadis, A. (2007), *Advanced computer aided design simulation of gear hobbing by means of three-dimensional kinematics modeling.*, J. Manuf. Sci. Eng. Trans. ASME, **129**, 911–918, doi:10.1115/1.2738947.
14. Tapoglou, N.; Antoniadis, A. (2012), *CAD-Based Calculation of Cutting Force Components in Gear Hobbing*. J. Manuf. Sci. Eng., **134**, 1–8, doi:10.1115/1.4006553.
15. Kyratsis, P.; Bilalis, N.; Antoniadis, A. (2011), *CAD-based simulations and design of experiments for determining thrust force in drilling operations.* Comput. Des., **43**, 1879–1890, doi:https://doi.org/10.1016/j.cad.2011.06.002.
16. Kyratsis, P.; Tzotzis, A.; Manavis, A. (2021), *Computational Design and Digital Fabrication*. In Proceedings of the Advances in Manufacturing Systems; Kumar, S., Rajurkar, K.P., Eds.; Springer Singapore: Singapore, pp. 1–16.
17. Tzotzis, A.; Garcia-Hernandez, C.; Huertas-Talon, J.-L.; Tzetzis, D.; Kyratsis, P. (2017), *Engineering applications using CAD based application programming interface*. In Proceedings of the MATEC Web of Conferences; **94**, pp. 1–7.
18. Kyratsis, P. (2020), *Computational design and digital manufacturing applications*. Int. J. Mod. Manuf. Technol., **12**, 82–91.

Received: April 06, 2021 / Accepted: December 20, 2021 / Paper available online: December 25, 2021