International Journal of Modern Manufacturing Technologies
ISSN 2067–3604, Vol. XII, No. 1 / 2020

COMPUTATIONAL DESIGN AND DIGITAL MANUFACTURING APPLICATIONS

Panagiotis Kyratsis

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

Corresponding author: Panagiotis Kyratsis, pkyratsis@uowm.gr

Abstract: Product design and manufacturing needs shorter time to market and lower costs in order to increase the resulted productivity. A series of tools and frameworks have been developed in order to emphasize the use of modern CAD systems in automating a number of design processes and downstream applications. As a result, design engineers increase their efficiency by learning to program CAD systems and implement their ideas in an extremely effective way.
The present paper contributes towards the use of different CAD systems based on their programming for designing and manufacturing products. A number of case studies are presented and prove the originality of the proposed approach. SolidWorks™ and Rhino™ are used for this purpose, together with their programming tools such as Visual Basic for Applications (CAD based Application Programming Interface) and Grasshopper (a graphical algorithm editor tightly integrated with Rhino™'s 3-D modeling tools) respectively. Additionally, several CAD based features are employed, such as the model feature recognition, the object properties extraction and the geometric topology interface. The presented case studies include the development of a fully functioning application for automatic generation of enginnering documents and routine design tasks (e.g. selection of geometric topology) that can be embedded to the user's workstation. Moreover, computer routines for the automatic generation of complex and unique three-dimensional geometries and patterns used in additive manufacturing processes are also included. Finally, testing of the aforementioned case studies dictate that the time conserved, when applying these strategies is remarkable, hence the development of a product or a system can overall be enhanced.
Key words: CAD, Computational Design, Application Programming Interface, Automation

1. INTRODUCTION

Nowadays modern industries aim at automated product design and manufacturing as much as possible. In addition, the use of computer aided design and manufacturing (CAD/CAM) systems is a must tool for design engineers in the industry. Hence, the implementation of application programming interface (API) in routine tasks is a topic that draws the attention of many researchers related to industry in the past few years. The API provides the user with vast possibilities that can lead to productivity enhancement via the development of applications for engineering purposes. Such applications can include the automatic operation of various CAD features that can be embedded into a user interface and modern tools, based on programming, that provide innovative and parameter-driven design solutions. In addition, the applications can include interactions between software modules or a piece of software and a programmable machine. Subsequently, the main CAD-based tool for the automated generation of innovative design solutions is computational design. In computational design, geometry forms are shaped by values of parameters and the relationships between these forms are described by mathematical context. Therefore, a large number of applications are created and are based at the same time on mathematics and programming resources.

2. LITERATURE SURVEY

2.1 CAD based use of APIs

Viganò and Osorio-Gómez (2012) defined an approach to extract the liaison graph from a 3D CAD model and analyze a method to find at least a feasible assembly sequence at the early stage of the design process of a product and by means of the database of the PDM/PLM systems. The method could be useful to search the optimal sequence of assembling for a product, by comparing different sequences extracted in automatic mode from a 3D CAD model. Xia et al. (2015) proposed a software framework related to the unified representation architecture (URA) that makes computer aided design (CAD) and computer aided engineering (CAE) to be an organic entity and contains three components. The URA facilitates the incorporation by explicitly representing design and analysis information as design features, which maintains their associations through the history chain. Moreover, they proposed a unified mesh data (UMD) to unify the mesh of CAD model display and CAE analysis with the purpose of reducing the redundancy of mesh data. Chen et al. (2017) proposed a CAD-LCA (Life Cycle Assessment)

software integration approach based on two types of features: a) the Product Feature (PF) and b) the Operation Feature (OF), for life cycle representation. Authors developed a NX Unigraphics plug-in tool for PF extraction. Additionally, they proposed a feature-based LCA system prototype tool, which has the capabilities of processing product feature models, generating the life cycle process model based on product-to-operation feature mapping and performing life cycle inventory (LCI) and impact assessment. The work of Ding et al. (2016) focused on developing a fully automated system using robotic gas metal arc welding to additively manufacture metal components. Authors included a user-friendly interface so that operation of their system by non-experts to be feasible.

García-Hernández et al. (2016) developed a software application for measuring gears and described its implementation using general-purpose spreadsheet software. The communication between the spreadsheet and the coordinate measuring machine (CMM) software was established by ASCII files. Tapoglou (2019) presented a novel simulation model, embedded on a CAD environment that enables the accurate prediction of the non-deformed chip geometry, the form and dimensions of the chips produced during the cutting process as well as the characteristics of the gear gap. The work of Kyratsis et al. (2019) dealt with the development of an application, by using the API of a commercially used CAD system. This application provides the user with a simple and easy to use platform that can automate the design process of a high-profile product by considering many different design aspects. Chatziparasidis and Sapidis (2017) presented a solution method for knowledge-based engineering (KBE)-CAD transformation problem by using two product models: a) the schematic assembly model (SAM) and b) the intermediate assembly model (IAM). The SAM is designed to fully employ all sorts of information available in the KBE system, and incorporate that either in the list of 'SAM components' or in the related 'SAM connection rules', whereas the IAM translates this 'SAM model' into 3D part models and assembly features, in a manner that production of the final 3D mechanical-CAD model is automatic. Hong et al. (2014) investigated a dynamic assembly simplification approach in order to demonstrate and interact with virtual assembly process of complex product in real time. Authors proposed a new assembly features definition to assist the assembly features recognition, which is a main step of the dynamic assembly simplification. Oancea and Haba (2016) presented a new software tool, specialized in rotational parts, which allows the user to obtain the manufacturing sequences, cutting data for each process from the manufacturing sequence and finally to assist at the simulation of each process.

2.2 CAD based use of graphical algorithm editors

Computational design is the procedure of using programing to create and modify form, structure, and ornamentation. Furthermore, parametric modelling allows immediate generation of large number of design alternatives. Mitchell (1978) describe the term Generative Design Systems as devices that are capable of generating potential solutions for a given problem. Computational and Generative Design offers a number of benefits to CAD systems that can extend traditional CAD-based design techniques. According to Killian (2006) and Terzidis (2003) computational design methodologies allow automation of the design procedures and extension of the standard features of CAD applications, therefore overstepping their limitations. Applying these limitations, Krause (2003) dealt with the development of applications, by using the computational design methods to generate structures or objects. This means that, designers are able to program (textual languages) or develop programs (visual programming) that when executed, produce unique geometric models. Leitão and Santos (2011) categorize textual and visual programming languages in terms of representation method, and describes them, with examples of applications. Scripting language offers large number of assets in programming for CAD applications. Furthermore, textual programming languages allow control within a software (i.e. Python™ commands and structures joined with Rhino™ via RhinoScriptSyntax module). In addition, Green and Petre (1996) defined that visual programming languages allow users to create graphic-based programs rapidly, simple and more flexible by using program icon-like elements rather than by scripting code (i.e. Grasshopper™ graphical elements and routines join with Rhino™).

Grasshopper™ is described as a graphical algorithm editor and it enables developing parametric designs through visual programming. The numbers of designs that are generated from Grasshopper™ are digital objects in Rhino™. Especially, the representative compound geometries of 3D objects are NURBS surfaces. The RhinoScriptSyntax module contains functions that perform a variety of operations on Rhino™. The library of functions is about geometry, commands, document objects and application methods. Certainly, all these functions are returned as simple Python™ variables due to the ease of design processing. Kyratsis et al. (2019) dealt with the development of procedures that identifies the digital form design process in which computer simulation software tool is used to generate forms in 3D space. These parametric generative forms aim on designing innovative objects (Tzintzi et al., 2017) for 3D printing and laser cutting applications.

The current paper presents a number of case studies that prove the value and the advantages that derive from the

implementation of the API and the programming resources of CAD systems. Specifically, a programming approach is being proposed related to the automatic creation of the engineering documents for the manufacturing of standard mechanical systems and to the automatic feature creation.
Additionally, the computational design point of view is introduced through illustrative case studies. Designs based on RhinoScriptSyntax Module (Python™ and Rhino™) and on Grasshopper™ application offer a variety of examples and features as displayed in the present work.

3. DEVELOPMENT OF CASE STUDIES

3.1 Automating engineering documentations

Engineering documents may include drawings, process plans and bill of materials (BOM). These documents enable the manufacturing process of a product or a system. The following case study presents the opportunity to create a tool for automatic creation of such documents with the aid of SolidWorks™ API (Fradinho et al., 2015). Figure 1 illustrates the workflow for the automated creation of the necessary documents. Firstly, the already designed assembly of the mechanical system opens in silent mode, so that the user may not be disrupted. Moreover, all the components that comprise the assembly open in silent mode too. Next, the default information and the custom properties that the assembly contains are extracted and then exported to a spreadsheet (this action requires that Excel™ or similar software is already installed). The type of information and the number of properties that can be extracted depend on the selection of the user. Typical properties are the part number, the description, the material, the weight, the quantity and the cost of a part. User may choose which of the properties and attributes of the model wants to include in the BOM and even add new ones. Subsequently, a new drawing document opens. All the pre-defined from the user options, such as the paper size, the drawing standards and the memorandum, are applied to the new document. At the same time, the extracted data (e.g. the BOM) are placed on the document at a pre-defined from the user position and with a specified style. Finally, the standard 3-view drawing of the first part in the assembly is placed on the new document. With the use of a "For" loop, the same procedure is performed for the rest of the assembly's components. Hence, depending on the number of the components that the assembly contains, an equal number of sheets are created. On top of that, a standard 3-view drawing of the assembly and a sectioned view drawing are also created (Figure 1).
To open the assembly of the designed product and its components the "OpenDoc6" method was implemented. The usage of this method requires the name and the type of the object to be opened. The mass related properties of the assembly and each of the parts can be extracted with the "CreateMassProperty" method, whereas the custom properties such as the material, weight, cost, etc. can be linked to a BOM and at the same time saved to a file with the aid of "InsertBomTable3" and "SaveAsText2" methods respectively. The creation and the setup of the new drawing document are achieved with the "NewDocument" and the "SetupSheet6" methods.

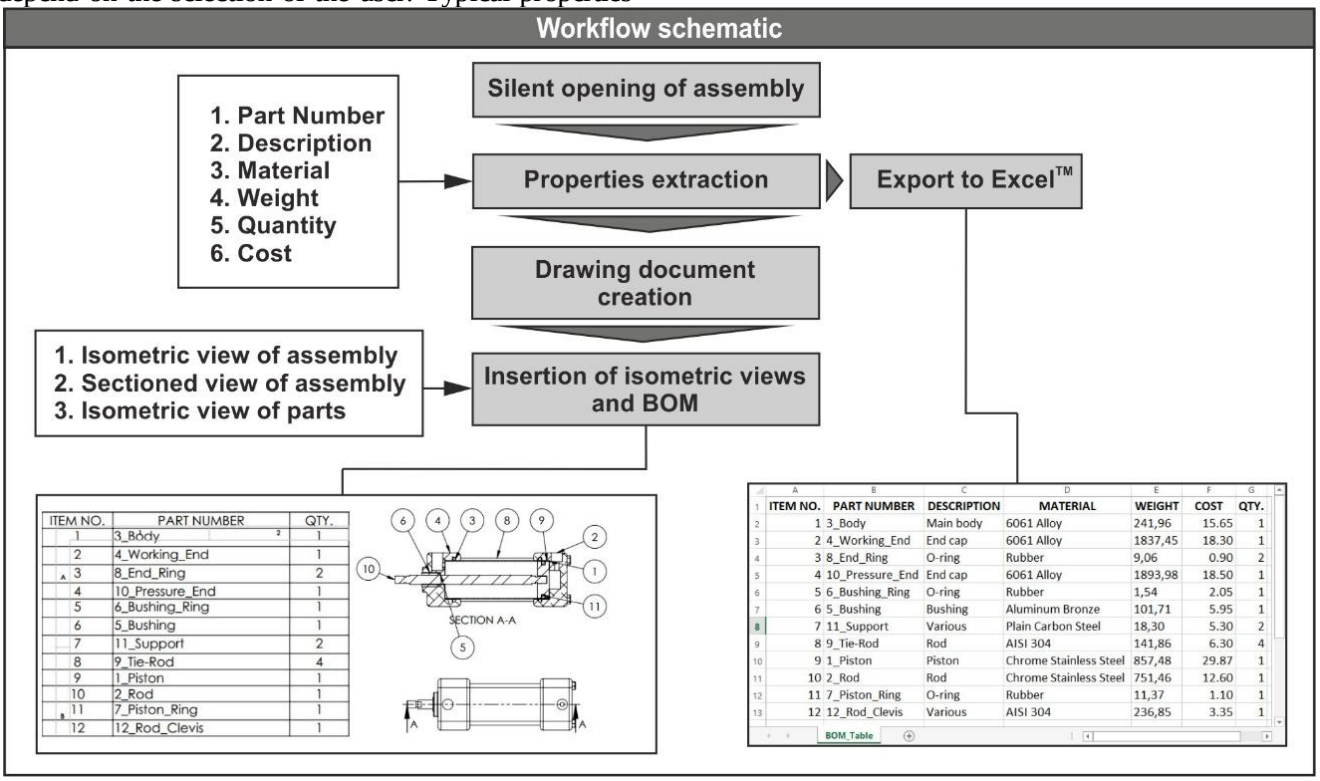

Fig. 1. The workflow of the automatic document creation tool

User must define a template for the proper use of the first method, whereas for the latter must specify a number of options such as the sheet size, the scale, the name of the sheet and the margins. Each new sheet of the drawing document is created with the "NewSheet4" method. The insertion of the defined views of the model along with the annotations (dimensions, symbols, etc.) is realized with the "CreateDrawViewFromModelView3" and the "InsertModelAnnotations3" methods accordingly. The placement of the model views is pre-defined by configuring three variables (Double) that represent the three coordinates X, Y, Z of the placement location. The insert of the annotations is performed based on the chosen source of dimensions (e.g. a designed part) and the type of annotations the user wishes to include in the drawing. Finally, the "CreateAutoBalloonOptions" method is required to create an array, where the preferences for the balloons are stored. These preferences are applied to the balloons, which are inserted to the model of the drawing with the "AutoBalloon5".

Table 1 contains the most important SolidWorks™ API methods that were used in order to develop the aforementioned tool with the aid of Visual Basic™ for Applications (VBA) programming language. It is possible that the syntax of the methods may differ slightly depending on the SolidWorks™ version.

It is concluded, that the presented tool can lead to short developing times of a product and may increase the productivity of design engineers, since they are relieved of time-consuming routine tasks and are free to focus on more creative assignments.

Table 1. Basic API methods for the automatic document creation

Method	**Usage**
OpenDoc6	Opens an existing document
CreateMassProperty	Obtains mass property information about one or more solid bodies in the document
InsertBomTable3	Inserts a BOM table in a part or assembly document
SaveAsText2	Saves a table to a text data file
NewDocument	Creates a new document based on a specified template
SetupSheet6	Sets up the specified drawing sheet
NewSheet4	Creates a new drawing sheet in the current drawing document
CreateDrawViewFromModelView3	Creates a drawing view on the current drawing sheet using the specified model view
InsertModelAnnotations3	Inserts model annotations into the current drawing document in the currently selected drawing view
CreateAutoBalloonOptions	Creates an object that stores auto balloon options
AutoBalloon5	Automatically inserts BOM balloons in selected drawing views

3.2 Automating topology selection

Almost every design task performed with a CAD system requires the selection of certain topology objects such as a vertex, an edge or a surface. In this case study, a tool with a user interface (Figure 2, Kyratsis et al., 2011) is being presented for automatic creation of standard design features based on topology selection. The user interface makes possible the input of values and the selection of different options, thus is the mean for the interconnection between the user and the software (Kyratsis et al., 2018; Tzotzis et al., 2017).

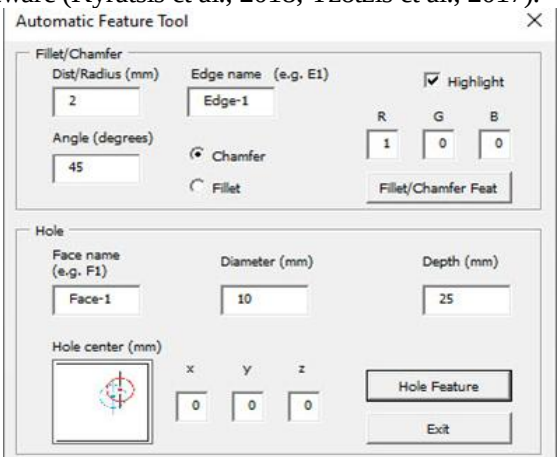

Fig. 2. The tool's interface

First step during the operation of this tool is the connection to an open model document (e.g. a part document) so that access to the solid bodies of the model can be achieved. Next, with the implementation of a "For" loop, the tool "searches" for the specified topology entity (face or edge) based on the user's preference. Upon finding all the model's faces or edges, the default ID of each one of them are changed according to the pre-defined prefix plus an ascending number as suffix. For example, if a part model contains twelve faces then they will be renamed to Face-1, Face-2, Face-3, …, Face-11 and Face-12 respectively. The renamed faces along with their new IDs are then stored to an array, which can be accessed by the tool's code. Finally, the topology object, which corresponds to the defined name is selected and the appropriate operation (hole, fillet or chamfer creation) is performed depending on the user's selection (Figure 2).

Optionally, the selected face or edge can be highlighted with the use of colour; this is particularly helpful, when working on a large assembly with many components. The hole feature corresponds to the selected face, whereas the fillet/chamfer feature corresponds to the selected edge. Figure 3 illustrates

the steps followed for automatically creating standard CAD based features on a selected topology object.
The "ActiveDoc" API method is required to make feasible the connection between the model document, that is currently open with the API objects. In case there is no document active, the user will be prompted to open one. Prior to the naming procedure, the tool must gain access to the solid bodies that are present in the model. This can be achieved with the "GetBodies2" method. For the retrieval of the faces and the edges, the "GetFaces" and the "GetEdges" methods are used accordingly. A "For" loop is utilized in order to retrieve each of the faces and edges in the solid bodies and create an array. With the same loop, the naming procedure of the topology objects takes place. The used method is the "SetEntityName" and the name is formed with a user specified prefix and an arithmetic suffix that is generated by a counter. Subsequently, the selection procedure is realized with the "GetEntityByName" and the "Select4" methods. Depending on the name of the topology object that the user wishes to process, the retrieval of the specified object is done with the "GetEntityByName" method, whereas the selection is performed with the "Select4" method. The term selection corresponds to the fact, that the selected object is marked as it would, when the user moves the pointer to that object and clicks. Finally, the methods "InsertFeatureChamfer" and "FeatureFillet3" are responsible for the creation of a chamfer or a fillet feature respectively, on the selected topology object. The usage of these methods requires the set of a number of parameters, such as the radius and the type of fillet, the width, the angle and the type of chamfer.

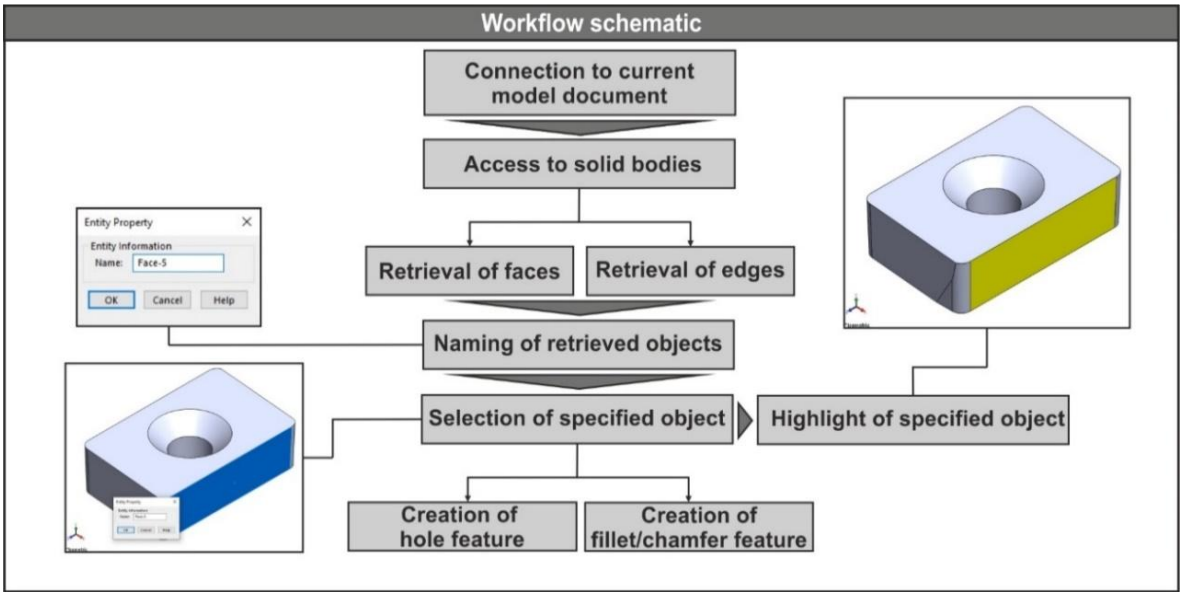

Fig. 3. The workflow of the automatic feature creation tool

The method that gets the options (RGB values) for the highlight function is the "GetMaterialPropertyValues2", whereas the method that applies the colour effect on the selected object is the "SetMaterialPropertyValues2".
Most of the SolidWokrs™ API methods that were used in the presented tool and their usage are included in Table 2. This case study demonstrates the possibilities that derive from the employment of the API and the CAD based programming resources. Concluding, the presented tool can aid design engineers to generate product designs in a faster and a more efficient way.

Table 2. Basic API methods for the automatic feature creation

Method	**Usage**
ActiveDoc	Connects to the currently active document
GetBodies2	Gain access to the bodies in the currently active part
GetFaces	Gets all the faces on the body
GetEdges	Gets the edges for the selected body
SetEntityName	Sets the name of the entity
GetEntityByName	Gets an entity (face, edge, vertex) by name
Select4	Selects an entity and marks it
GetMaterialPropertyValues2	Gets the material property values for the selected entity
SetMaterialPropertyValues2	Sets the material property values for the selected entity
InsertFeatureChamfer	Inserts a chamfer
FeatureFillet3	Creates the specified fillet feature for selected edges or faces
FeatureCut4	Creates a cut extrude feature

3.3 Computation design and textual programming

Every textual programming language follows three important principles: a) primitive elements, b) combination mechanisms, and c) abstraction mechanisms. Furthermore, textual programming languages are described using a linear sequence of characters (Findler et al., 2002). The contribution between textual programming languages and CAD pieces of software, offers users a great number of benefits, when creating compound geometries and structures.

The next case study presents an application for automatic creation of such unique 3D geometries with the aid of Python™ (textual programming language) and Rhino™ (CAD software). Figure 4 illustrates the workflow for the automated creation of 3D geometries. First, the user creates two different types of objects by using Rhino™ tools: a free form surface and two planes (A) and (B). At first, all dimensions of these elements are up to the users' choice. The designed surface is the one that will be cut (by the two planes) on the corresponding flat layers. Afterwards, the user is able to insert specific numerical values for the parameters required by the command line. The combination of parameters that the user chooses, results in building the final form of the 3D structure. The parameters used are:

- Step (number of cuts from Plane A and Plane B on the surface),
- Count (number of final surface layers)
- Extrude (layers' thickness) and
- Offset.

The proposed procedure was developed in Python™ programming language by importing *rhinoscriptsyntax module* to the Rhino™ software. Some of the commands used for the script are presented in Table 3.

Figure 5 depicts the second stage of the complete procedure and presents four alternative designs for the same surface. This emphasises the development of a complete family of geometries.

In more details:

- When the selected values are: Step=5, Count=20, Extrude=2 and Offset=0.2, the result of the produced structure (A) is a dense grid.
- Next, the user inserts new values, thus a new structure is created. The main characteristic is that the new structure (B) is thinner than the first one. The values for the second geometry are: Step=8, Count=20, Extrude=1 and Offset=1.
- Structure (C) is the thinnest structure from the four examples, because of the new set of values selected (Step=10, Count=20, Extrude=4 and Offset=0.2).
- Finally, structure (D) is characterised by the most high-density grid of all alternatives. The values for the last example are: Step=2, Count=80, Extrude=1 and Offset=0.1.

Likewise, the user can insert new values and the proposed application can build alternative styles of the basic structure. The suggested example is based on laser CNC cutting techniques – in order to create wooden structures for architectural and product design purposes. According to Kolarevich (2001), CNC cutting or 2D fabrication, is one of the most used fabrication techniques. Laser-cutters use a high-intensity focused beam of infrared light in combination with a jet of highly pressurized gas (carbon dioxide) to melt or burn the material that is being cut. The alternative types of structures that are proposed from current application are different kind of prototypes with an aim to explore aesthetics and technical characteristics from the designers' point of view. It is concluded that the presented application can lead to a holistic tool that allow architects, mechanical engineers, product designers and artists to create different types of structures by using appropriate parameters and basic geometries.

Table 3. Basic script commands used for the computational design application

Command	**Usage**
GetObject	Inserts Plane A, Plane B, Surface
GetReal	Inserts values for Step
GetInteger	Inserts values for Count
ExtrudeCurveStraight	Inserts values for Extrude
OffsetSurface	Inserts values for Offset
MoveObject	Moves Plane A and Plane B
IntersectBreps	Creates segments

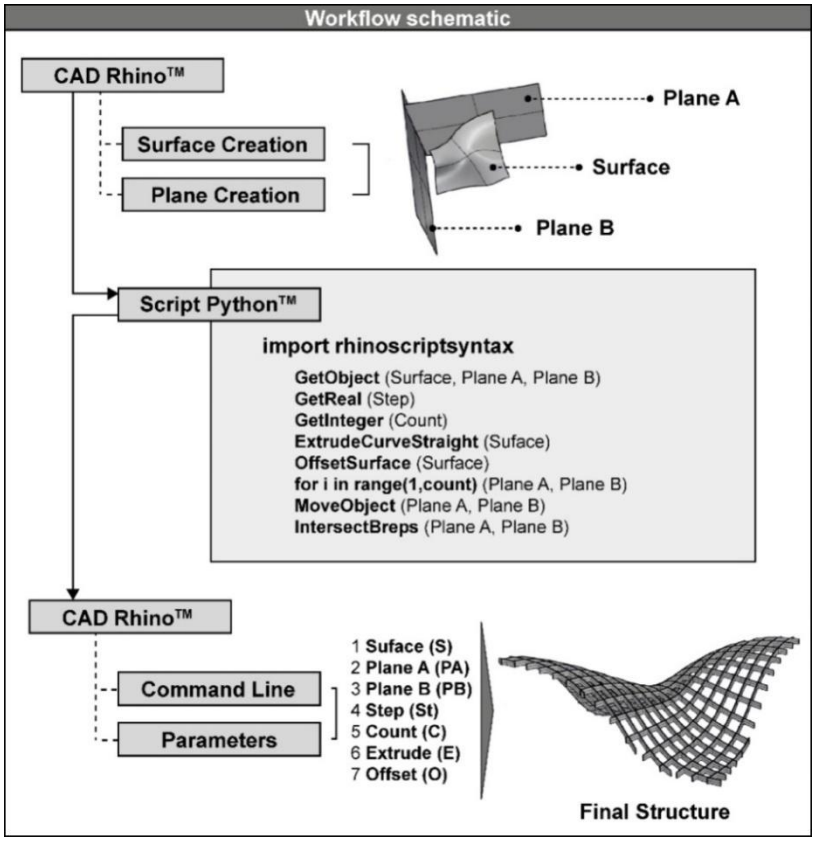

Fig. 4. The workflow of CAD application based on computational design and textual programming

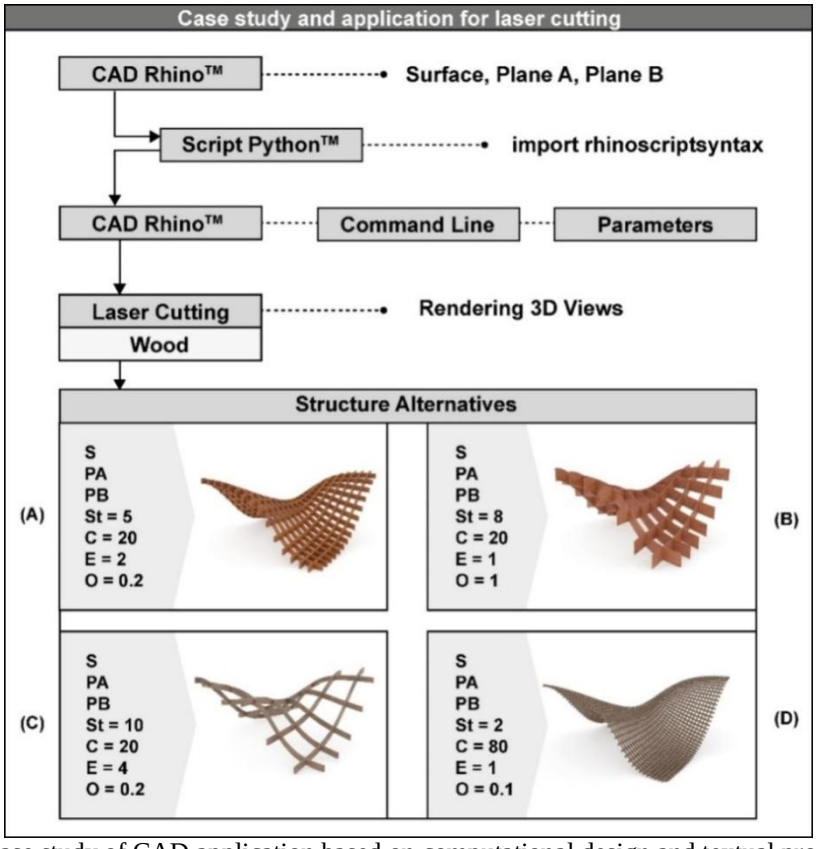

Fig. 5. A case study of CAD application based on computational design and textual programming

3.4 Computation design and visual programming

Myers (1990) focused on a term of visual programming language (VPL) that allows the description of programs in a bidimensional representation consisting of iconic elements that can be interactively manipulated by the user according to some spacial grammar. The proposed framework for exploration of design alternatives can be seen in

Figure 6. A parametric model is developed using Grasshopper™, which is a plugin for Rhino™. The most popular tool for the Rhino™ design community, in order to create models that are based on computational design principles, is Grasshopper™. Computational design is the action of using a visual programing language with an aim to create and modify form, structure, and ornamentation. Some of the benefits offered are: the precision, the automation, the generativity, the randomness and the parameterization achieved.

The following case study presents an application for automatic creation of such unique 3D patterns with the aid of Grasshopper™ (visual programming language) and Rhino™ (CAD software). Figure 6 illustrates the workflow for the automated creation of 3D patterns for 3D printing applications. At the first stage of the presented workflow, designer is able to create a series of Construct Points, using the Grasshopper™ visual elements. Two kinds of parameters define the command "Series": the Step (N) and the Count (C). The second stage includes all these commands that are necessary to count all points and the distances between them. Then these pieces of data are saved into a list. A command that allows sorting a list from the minimum to maximum number (or the opposite) is used for the construction of the "construct points" pattern. Following that, the user chooses a shape that will be the main geometry reproduced to the pattern morphology. The proposed shape for this case study is polygon. The final stage includes all commands from Grasshopper™ that give to polygons a 3D entity. Especially, the "Extrude" command, "Cup Holes" and "ReMap" command are used. The final result is a 3D pattern consisting of polygons.

As previously mentioned, at the second stage a list of X and Y coordinates of all of Construct Points was created. The specific list sorts points to the pattern morphology by the MIN/MAX parameter. Figure 7 presents that the user can change the parameters of X and Y coordinates, and as a result to transform the final pattern of 3D polygons. Two differentiated styles of pattern (A) and (B) are proposed. The parameters used for patterns (A) and (B) are: X-Coordinate=30, Y-Coordinate=10 and X-Coordinate=70, Y-Coordinate=74, respectively.

Furthermore, Figure 7 illustrates a proposal for an application. Especially, the suggested example is based on the principle that 3D printing technologies will be used, in order to create plastic structures for architectural and product design purposes. It is concluded that the presented application can lead to another holistic tool, based on visual programming language (VPL), that allow the user to create different types of patterns by using only visual elements and parameters in a graphic interface. The advantage of the proposed procedure is the automatic creation of alternative patterns, thus building a family of geometries.

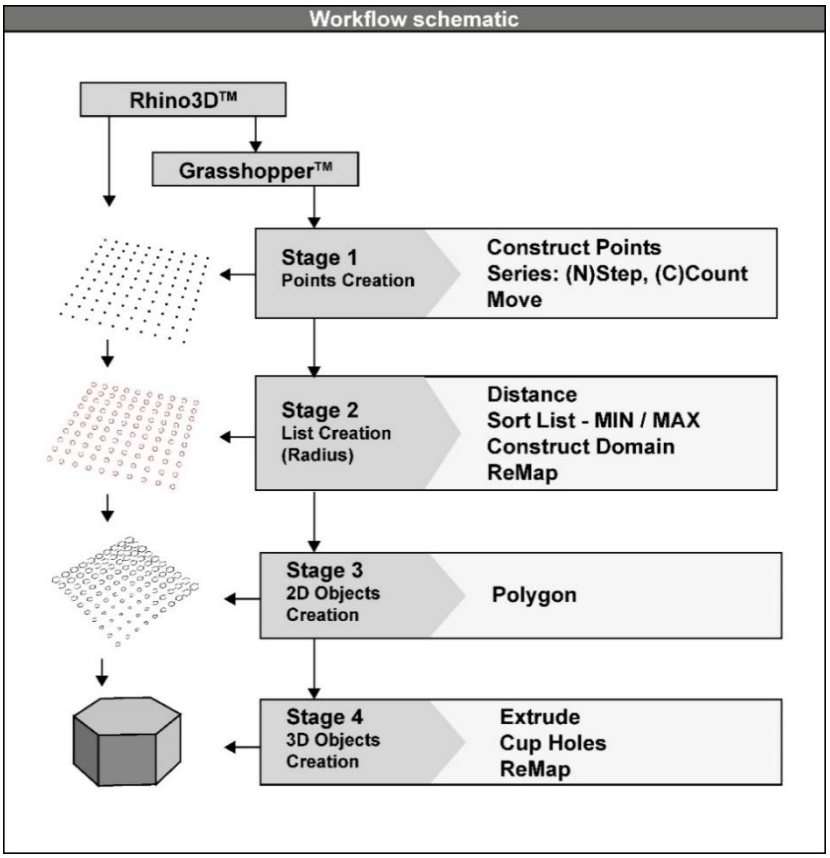

Fig. 6. The workflow of CAD application based on computational design and visual programming

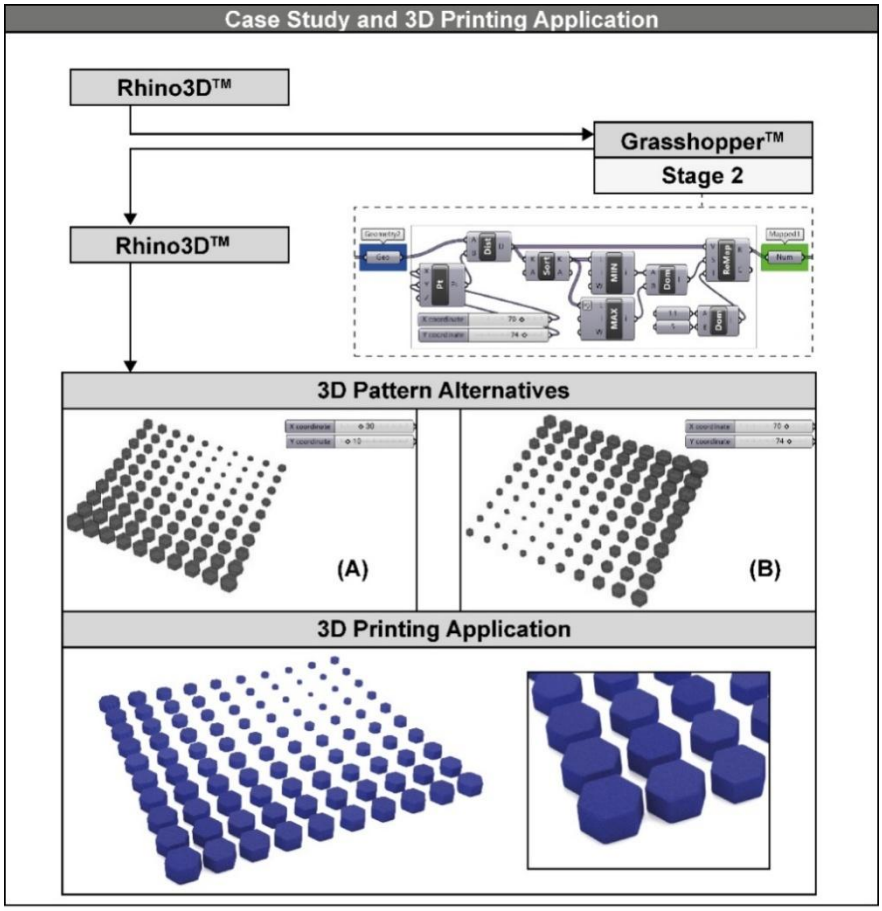

Fig. 7. A case study of CAD application based on computational design and visual programming

4. CONCLUSIONS

In the present work, an effort was made to present and discuss the possibilities that emerge through the CAD based programming and highlight the advantages that derive from the development of programs related to the automation of multiple design applications. Specifically, the implementation of the API and the programming resources of SolidWorks™ and Rhino™ CAD systems are being demonstrated via four case studies. The first two case studies deal with the automatic creation of engineering documents and the development of an interactive tool that can support common routine design tasks. On the other hand, the last two case studies are related to the automated procedure of creating complicated and unique 3D geometries and to the development of 3D patterns for 3D printing applications.

Upon finalizing the aforementioned case studies, the following conclusions can be deducted: a) the programming of CAD systems can be used to develop simple supportive macros and even complete applications, b) it is possible to develop application tools that embed a user-friendly interface for better visualization of various procedures and for establishing an interconnectivity between different pieces of software, c) most of the standard design tasks can be automated, hence reduce the development time of products and systems d) complex geometries and patterns can be simplified for use with additive manufacturing or 2D CNC laser cut-and-engraving operations.

Furthermore, such programming techniques can help design engineers to increase their productivity and allocate work time in a more efficient manner. Additionally, the value of these techniques increases, when the developed programs and tools provide assistance to processes where a great number of parameters is involved (e.g. simplification of complex surfaces with multiple planes and vertices for rapid and reliable 3D printing).

5. REFERENCES

1. Chatziparasidis I. and Sapidis N., (2017). *Framework to automate mechanical-system design using multiple product-models and assembly feature technology*, Int. J. Prod. Lifecycle Manag., **10**(2), 124-150

2. Chen Z., Tao J., and Yu S., (2017). *A Feature-based CAD-LCA Software Integration Approach for Eco-design*, Procedia CIRP, **61**, 721–726

3. Ding D., Shen C., Pan Z., Cuiuri D., Li H., Larkin N. and Van Duin S., (2016). *Towards an automated robotic arc-welding-based additive manufacturing system from CAD to finished part*, Comput. Des., **73**, 66–75

4. Findler R. B., Clements J., Flanagan C., Flatt M.,

Krishnamurthi S., Steckler P., and Felleisen M., (2002). *DrScheme: A programming environment for Scheme*, J. Funct. Program., **12**(2), 159–182
5. García-Hernández C., Gella-Marín R., Huertas-Talón J. L., and Berges-Muro L., (2016). *Algorithm for measuring gears implemented with general-purpose spreadsheet software*, Measurement, **85**, 1–12
6. Green T. R. G., Petre M., (1996). *Usability Analysis of Visual Programming Environments: a Cognitive Dimensions Framework*, J. Vis. Lang. and Comp., **7**(2), 131–174
7. Hong X., Yuan L., Jian-Feng Y., and Hui C., (2014). *Dynamic assembly simplification for virtual assembly process of complex product*, Assem. Autom., **34**(1), 1–15
8. Killian A., (2006). *Design innovation through constraint modeling*, Int. J. Arch. Comp., **4**(1), 87–105
9. Kolarevich, B., (2001). *Digital Fabrication: Manufacturing Architecture in Information Age, Reinventing the Discourse - How Digital Tools Help Bridge and Transform Research*, Education and Practice in Architecture, Proceedings of the Twenty First Annual Conference of the Association for Computer-Aided Design in Architecture, pp. 268 – 278,
10. Krause J., (2003). *Reflections: The Creative Process of Generative Design in Architecture*, Proceedings of the 6th international conference on generative art, 136-149
11. Kyratsis P., Tapoglou N., Bilalis N., Antoniadis A., (2011). *Thrust force prediction of twist drill tools using a 3D CAD system application programming interface*, Int. J. Mach.Mach., **10**(1/2), 18-33.
12. Kyratsis P., Tzotzis A., Tzetzis D., and Sapidis N., (2018). *Pneumatic cylinder design using cad-based programming*, Acad. J. Manuf. Eng., **16**(2), 107–113
13. Kyratsis P., Gabis E., Tzotzis A., Tzetzis D., and Kakoulis K., (2019) *CAD based product design: A case study*, Int. J. Mod. Manuf. Technol., **11**(3) 88–93
14. Kyratsis P., Manavis, A., Gianniotis, P. and Ghiculescu, D., (2019). *A non-Conventional methodology for interior product design using conceptual principles and parametric tools*, Nonconventional Technologies Review, **23**(4), 16-21
15. Leitão A. and Santos L. (2011). *Programming Languages for Generative Design: Visual or Textual?*, 29th eCAADe Conference Proceedings, 549-557
16. Mitchell W. J., (1978). *The theoretical foundation of computer-aided architectural design*, Environ.Plan., **2**(2), 127–150
17. Myers, B.A., (1990). *Taxonomies of Visual Programming and Program Visualization*, Vis. Lang. and Comp., **1**(1), 97 – 123
18. Oancea G. and 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**, 75-78
19. Tapoglou N., (2019). *Calculation of non-deformed chip and gear geometry in power skiving using a CAD-based simulation*, Int. J. Adv. Manuf. Technol., **100**(5–8), 1779–1785
20. Terzidis, K., (2003). *Expressive form: A conceptual approach to computational design*, (New York, USA: Spon Press)
21. Tzintzi V., Manavis A., Efkolidis N., Dimopoulos C., Kakoulis K. and Kyratsis P., (2017). *Conceptual design of jewellery: a space-based aesthetics approach*, in MATEC Web of Conferences **112** p. 07025 doi: 10.1051/matecconf/201711207025
22. Tzotzis A., García-Hernández C., Huertas-Talón J. L., Tzetzis D., and Kyratsis P., (2017). *Engineering applications using CAD based application programming interface*, in MATEC Web of Conferences, **94**, p. 7
23. Viganò R. and Osorio-Gómez G., (2012). *Assembly planning with automated retrieval of assembly sequences from CAD model information*, Assem. Autom., **32**(4), 347–360
24. J Fradinho, D Nedelcu, A Gabriel-Santos, A Gonçalves-Coelho, A Mourão, (2015). *Some trends and proposals for the inclusion of sustainability in the design of manufacturing process*, IOP conference series: Materials science and engineering **95**(1), 012142
25. Xia Z., Wang Q., Wang Y. and Yu C., (2015). *A CAD/CAE incorporate software framework using a unified representation architecture*, Adv. Eng. Softw., **87**, 68-85

Received: March 13, 2020 / Accepted: June 15, 2020 / Paper available online: June 20, 2020