

SolidWorks Secondary Development with Visual Basic 6 for an Automated Modular Fixture Assembly Approach

Uday H. Farhan

*Saff Mechanical Engineer
School of Engineering, Edith Cowan University
270 Joondalup drive, Joondalup,
Western Australia , 6027*

u.farhan@ecu.edu.au

Simona O'Brien

*MEng Sc MIEI, School of Engineering,
Edith Cowan University, 270 Joondalup drive,
Joondalup, Western Australia , 6027*

simona.obrein@ecu.edu.au

Majid T. Rad

*Senior lecturer, School of Engineering
Edith Cowan University
270 Joondalup drive, Joondalup
Western Australia , 6027*

m.rad@ecu.edu.au

Abstract

Modular fixtures (MFs) play an important role in terms of cost and production time reduction in manufacturing processes. In this paper, the authors illustrate an automated approach for MFs design and assembly. This approach is based on the secondary development of SolidWorks integrating with Visual Basic (VB) 6 programming language. SolidWorks API (Application programming interface) functions were applied in order to control SolidWorks commands and assembly operations. An ActiveX DLL project was created in VB 6 and a plug-in file in .dll format was generated. The outcomes were creating new menus in SolidWorks environment for selecting, inserting, and assembling MFs elements. The approach was applied for a side clamping procedure and for a semi-circular workpiece.

Keywords: Modular fixtures, SolidWorks API, Visual basic 6, Assembly simulation.

1. INTRODUCTION

As a result of the rapid development of the manufacturing systems, productivity has become one of the goals that need to be achieved as well as production time consuming [1]. Given this evidence, modular fixtures (MFs) have been considered as one of the important factors for achieving these goals [1]. This became from the flexibility of MFs elements that provide several workholding procedures for different workpieces [2] and this brought many benefits in flexible manufacturing systems (FMSs) that use computer numerical control (CNC) machines. The design and assembly process of MFs was the aim for many previous studies. Some of them focused on the assembly relationships of the fixture elements [3]. Others concentrated on information presentation of MFs [4]. For more improvement of MFs design, CAD technology has been utilised for automating the design and assembly for fixture elements. Information models were used as a tool for supporting fixture decision [5].

An approach for positioning the fixtures with rapid reconfiguration method was presented for an "intelligent fixture system" [6]. ICAD system and UG-2 modelling were used for developing MFs design system based on CAD software [7]. AutoCAD with 2D drawings was applied [8] while AutoCAD 3D modelling was employed for this purpose [9]. For more effective automated MFs systems, three factors should be considered by the designer, the database of the fixture elements, the proper design approach, and a powerful 3D CAD software for modelling MFs design and assembly process. The database of standard fixture elements was created in the previous work as well as introducing the appropriate methodology in order to develop a computer – assisted fixture design (CAFD) [1]. SolidWorks was used as CAD

software due to its excellent 3D capabilities for performing the modelling process for MFs. Based on the previous work; the use of secondary development of SolidWorks with VB 6 for automating fixture elements design and assembly is introduced in this paper. The developed approach is shown in Figure 1. This approach includes generating ActiveX DLL project in VB 6 and integrating with SolidWorks API for creating new menus in SolidWorks environment. A 3D model of the specific workpiece was designed in SolidWorks. The database for the standard fixture elements was established in the previous work [1]. The knowledge base for the assembly relationships was created by using If-then rules. This knowledge base is based on the mating features for assembly purpose in SolidWorks.

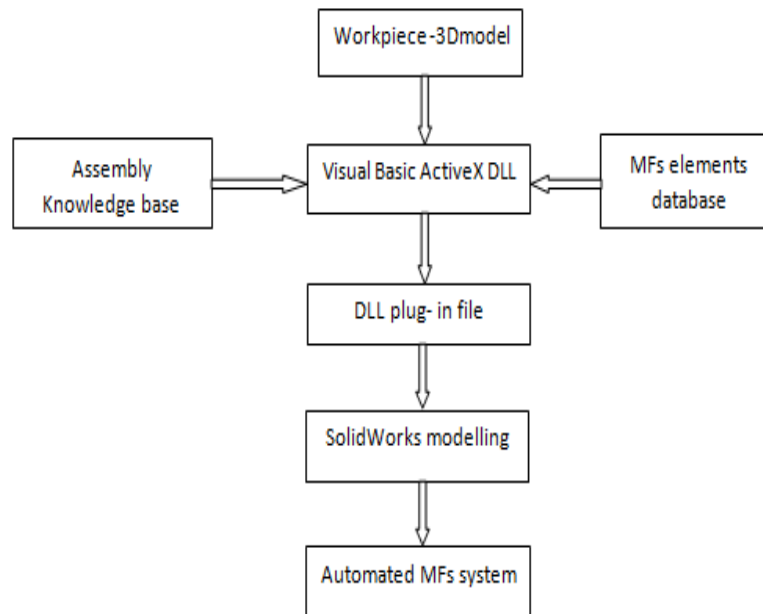


FIGURE 1: The flow chart of the developed approach.

2. WORKPIECE HOLDING PRINCIPLES

The produced workpiece should meet its specifications particularly for shape, dimensions and tolerances and to achieve this, it is important to locate and clamp the workpiece correctly on the machine [10]. Some other factors also influence the workpiece holding configuration including machining operation sequence, cost considerations, direction and strength of the cutting forces, and capabilities and orientation of the machine tools [10]. Moreover, fixture elements may be designed for a specific workpiece; and these are called dedicated fixtures. Conversely, fixtures can be selected and combined with a database of standard fixture elements; and these are called modular fixtures [10]. There is a specific function for each fixture element, and a completed fixture structure can be built from a number of elements, including the consideration of the types, classes and the functions that will lead to the appropriate machining operations.

2.1 Modular fixturing

Modular fixturing can be defined as a system for building several combinations of standard components that can serve a wide variety of workpiece. These fixture elements can be assembled and reused in order to generate different constructions of jigs and fixtures [2]. The building process of modular fixturing system depends on selecting the necessary fixture components to be assembled and this assembly process will be the base for building “more detailed systems”. As a result, fixture elements can be built for any kind of workpiece by using modular fixturing systems [2]. The benefits of modular fixturing result on reducing the design and assembly time by eliminating the use of the “dedicated fixtures” and their special

components [10]. Moreover, the database of the modular fixtures contains the necessary design's information with mating features for the standard elements and it is easy to be modified [10]. Modular fixturing systems were classified into three major kinds based on the construction basics: Subplate, T-slot, and dowel pin [2].

2.2 Modular fixturing building

Modular fixturing systems consist of a set of standard elements such as baseplates, locators, supporters, clamps, and all the other accessories (Figure 2). By assembling these elements together, suitable workholding systems for variety of workpiece can be achieved [2]. The assembling process starts with selecting the baseplate depending on the size of the workpiece. Then the locating elements are chosen and assembled to the baseplate. After that, the clamping elements are selected to suit the chosen workpiece. Finally, the other elements and accessories are added for completing the modular system [2]. After building the workholding modular system, the machined operations are started in order to produce the specific part or the workpiece. When the part production process is finished, the modular system is disassembled and the elements are sent to the store to be used for building other modular workholding systems [2].

Figure 3 shows an example of modular fixturing systems. The advantages of reuse the modular fixture elements lead to reducing time and costs. One hour of modular fixturing building process equals to about six hours of "conventional jig or fixture" building [2].

2.3 Visual Basic fundamentals

Visual Basic (VB) is a programming language developed by Microsoft. This language has been used widely among a high percentage of the developers as "primary development tool" compared to other programming languages [11]. VB is the advanced version of the BASIC language with "Integrated Development Environment (IDE)" which API accessing and graphical user interface [12].

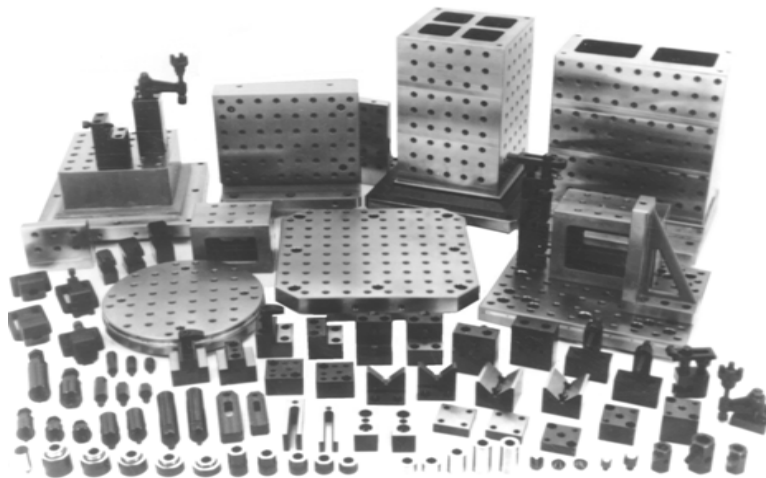


FIGURE 2: Modular fixture standard elements [2].

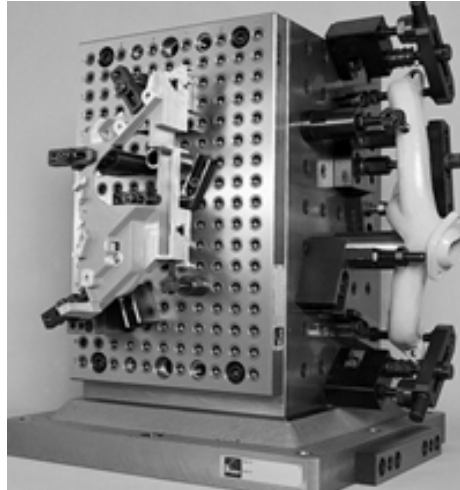


FIGURE 3: Modular workholding system [13].

Moreover, VB is considered as the engine for building the macros in all Microsoft software [11]. Therefore, VB has become an important tool for building different programs for many applications. There are different types of projects that can be generated in VB. For simple programming purposes, Standard EXE is used more commonly by the programmers. For more advanced programming functions, ActiveX projects are used. In this research, ActiveX DLL (ActiveX dynamic link libraries) has been created. This project allows the programmer to integrate VB with different windows applications. Also, this project controls the features and the operations of other applications by creating new menus and toolbars into the applications' environment. For database management purposes, VB is the engine of Microsoft Access for building the database and this gives the programmer the opportunity for controlling the database efficiently [11].

2.4 SolidWorks API

Application programming interface (API) is a tool to write a code in a programming language within another application. As a result, a direct integration between different applications can be developed [14]. SolidWorks is one of the applications which support API with different programming languages such as C++, Visual Basic, and Visual Studio. SolidWorks API automates the design and the assembly operations by creating codes in a specific programming language [14] and it has been applied for different design methods. API is used to develop a web service material database with SolidWorks for simplifying the selecting of the materials by the designer [15]. Bo, Qin, and Fang developed a standard parts library by using Visual Basic code with SolidWorks API functions [16]. This system is based on creating Visual Basic forms for improving the design efficiency. Peng, Jing, and Xiaoyan applied Visual Basic Net in the SolidWorks second development for simulating "3D module of architectural process" [17]. They generated add-in VB project into Solidworks for automating the assembly process. SolidWorks API was employed in designing a model of "centrifugal fan impeller" [18]. This method is based on the geometric features to create the second development with SolidWorks and VB 6. Moreover, API with the user interface implementing knowledge based design features (KBE) application can help in customising the CAD system for certain tasks [19]. Reuse software was developed by applying SolidWorks API by Tian and Liu [20]. The system was built by using VB for secondary development of the CAD system for a standard part. Zhen and Yingyi introduced an assembly method based on SolidWorks [21]. They illustrated the steps of the assembly automation procedure by using Visual C++. They also explained how the information of the parts was stored in the database. The secondary development of SolidWorks was used by Yang for developing an intelligent system for assembly based on the parametric design [22]. Delphi programming language, SolidWorks API, and Access database were the techniques for creating this system.

3. CREATING THE ASSEMBLY KNOWLEDGE BASE

Based on the assembly methodology in the previous work [11], the assembly knowledge base was developed in IF-then rule structure to meet the requirements of the side clamping layout for the selected fixture elements in this study. Examples for these rules are:

Rule 1

If the locating method is V-blocks, then define the surfaces that are used to assemble them to the baseplate.

Rule 2

If the locating surface of the V-block is defined, then identify the location of the V-block on the baseplate. This depends on the sizes of the V-block and baseplate. The location can be defined by calculating the distance of the holes on the baseplate for the correct location.

According to the findings from the above rules, the mating features for assembling the V-block and the baseplate in the SolidWorks environment were defined:

Rule 3

If the surface is defined and the location is calculated, then use coincident mate to locate the V-block to the baseplate.

Rule 4

If the surface is defined and the location is calculated, then use concentric mate to assemble the V-block with the baseplate.

4. CREATING SOLIDWORKS ADD-IN AND USER INTERFACES

In this paper, add-in code was created in order to build menus in the SolidWorks environment for automating the inserting and assembly processes. The menus were built by using VB 6 integrating with SolidWorks API (Application Programming Interface). An ActiveX DLL project was developed in VB including SolidWorks libraries as references. In this case, two libraries were referenced; these are SldWorks Type Library and SolidWorks Exposed Type library. Adding these libraries to the ActiveX DLL project allows control SolidWorks commands and functions. The developed add-in code in this study is:

```
Dim bRet As Boolean 'boolean return
Dim lRet As Long 'long return
Dim axMenuID As String
Dim lngToolBarDocTypes As Long
Set axSldWorks = ThisSW
axCookie = Cookie
bRet = axSldWorks.SetAddinCallbackInfo(App.hInstance, Me, axCookie)
axMenuID = "Modular Fixture System"
lRet = axSldWorks.AddMenu(swDocASSEMBLY, axMenuID, 5)
axMenu1 = "Assembly@Side Clamping@" & axMenuID
axMenu2 = "Back stop@Side Clamping@" & axMenuID
axMenu3 = "Pivoting Clamp@Side Clamping@" & axMenuID
axMenu4 = "Riser block@Side Clamping@" & axMenuID
axMenu5 = "Workpiece@Side Clamping@" & axMenuID
axMenu6 = "V-block@Side Clamping@" & axMenuID
axMenu7 = "Baseplate@Side Clamping@" & axMenuID
axMenu8 = "Top Clamping@" & axMenuID
```

After writing the add-in code and adding the proper VB modules and forms to the ActiveX DLL project, the .dll file is created and copied to the SolidWorks directory. Then, this .dll file is opened in the SolidWorks environment to apply the function to the developed menus.

4.1 Assembly simulation by macros

For automating the fixture elements assembly process in SolidWorks, macros were created for simulation purposes. However, these macros can be applied only for the master SolidWorks document that they have been created in. The solution for this problem was to make global macros by modifying the recorded macros. This was completed by changing VB methods and classes of the recorded macros and adding the swConst modules and swAssembly or swPart class modules. This makes the macros available for any SolidWorks documents. However, these global macros are still not in the format for the developed ActiveX DLL project. The most important function for the add-in project is how to make the created menus calling the global macros which perform SolidWorks design and assembly. This was achieved by importing the global macros into the ActiveX DLL project as modules with .bas format and then writing a subroutine code for each macro and finally calling this subroutine by the specific menu's icon.

4.2 Implementation of the assembly simulation

The assembly simulation started by developing the macros for each fixture element and for the assembly steps. The macros for adding the fixture elements were created first. For more flexibility, a main menu called Modular Fixture System was built. Then, two sub menus were developed called Side Clamping and Top Clamping. The user can select which type of clamping system should be applied. Both sub menus is extended to other menus for the whole related fixture elements for this specific kind of fixturing (Figure 4). For the Side Clamping, the following elements were used:

- Baseplate;
- V-Block;
- Pivoting Clamp;
- Backstop;
- Riser block;
- Other accessories.

For each of these elements, a menu should be created. Then, by selecting the specific menu, a window is opened for more detail. For example, when clicking on the baseplate menu, a window of this element is opened to help the user select the proper baseplate (Figure 5). The window contains command buttons for the possible and available baseplates that could be used within the system. Each command button was highlighted by an icon of the specified baseplate and ID number was used for choosing the correct baseplate. The interface provides the flexibility for defining the location of the baseplate in X, Y, and Z directions in the SolidWorks environment. Then, the user marks the check box for the selected baseplate and then clicks on the command button.

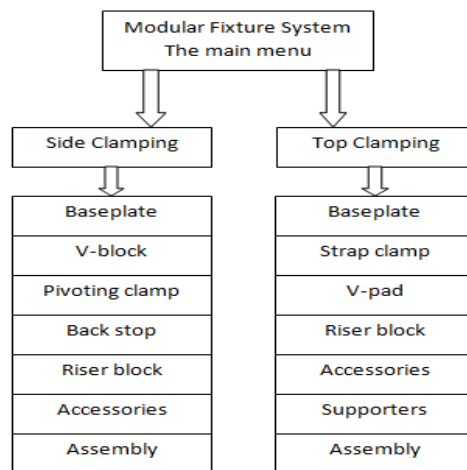


FIGURE 4: The developed main and extended menus.

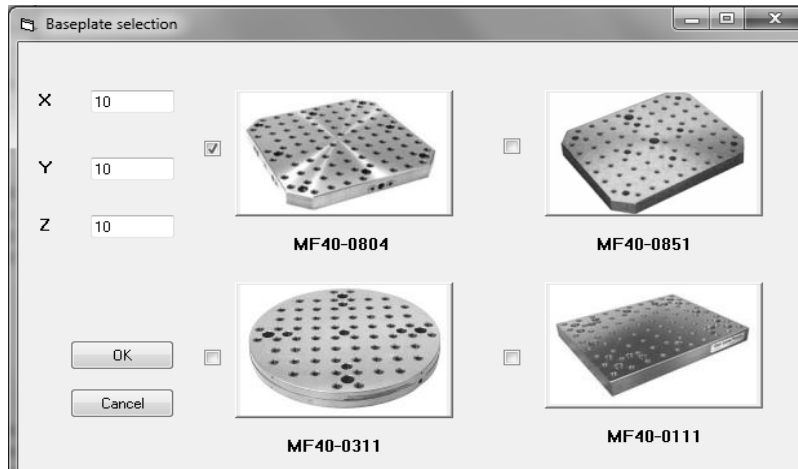


FIGURE 5: Four kinds are included in the baseplate selection interface.

In order to activate the command buttons in the baseplate selection interface, a macro was created to perform the inserting process of this element:

```
Dim swApp As Object
Dim Part As Object
Dim boolstatus As Boolean
Dim longstatus As Long, longwarnings As Long
Sub main()
Set swApp =Application.SldWorks
Set Part = swApp.ActiveDoc
boolstatus = Part.AddComponent("C:\Users\Desktop\baseplate -MF40-0804)
End Sub
```

This macro adds the baseplate MF40-0804 to the active SolidWorks application document named swApp. The macro should be now transferred to the global form by attaching swConst module and swAssembly or swPart class module. The swConst module contains the definitions for the SolidWorks API functions including the properties and methods. The swAssembly and swPart class modules define which type of SolidWorks documents that the specific macro will be applied to.

The next step is adding the macro to the ActiveX DLL project and converting it to the .bas format which makes the macro available for any SolidWorks assembly or part documents. The swConst module and swAssembly class module are also added to this project to improve availability for any assembly SolidWorks document. The previous process was followed for adding the rest of the macros for the elements to the project.

After completing selection and adding the elements to the SolidWorks environment, these elements need to be assembled. A menu called Assembly was created and added to the list of the extended menus as shown in Figure 3. By clicking this menu, an assembly interface is opened (Figure 6). This interface contains command buttons for executing the assembly simulation for the fixture elements.

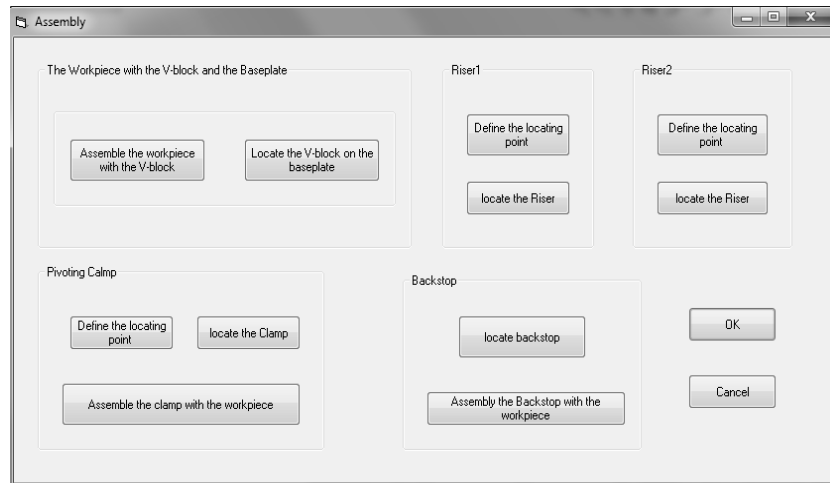


FIGURE 6: The user interface of fixture elements Assembly process.

The simulation of the assembly process begins by taking the V-block and the workpiece as an example. These two elements were added to SolidWorks and then a macro for assembling them was recorded. A tangent mate feature was used between the same-cylindrical face of the workpiece and the two V shape faces of the V-block as shown in Figure 7. The code of assembly is:

```

Dim swApp As Object
Dim Part As Object
Dim boolstatus As Boolean
Dim longstatus As Long, longwarnings As Long
Sub main()
Set swApp = _
Application.SldWorks
Set Part = swApp.ActiveDoc
boolstatus = Part.Extension.SelectByID2("", "FACE", 0.1467759532569, 0.2282465512145,
0.2052684525268, True, 1, Nothing, 0)
boolstatus = Part.Extension.SelectByID2("", "FACE", 0.04935040746813, 0.05664959253187,
0.08677583155765, True, 1, Nothing, 0)
Dim swApp As Object
Dim Part As Object
Dim boolstatus As Boolean
Dim longstatus As Long, longwarnings As Long
Sub main()
Set swApp = _
Application.SldWorks
Set Part = swApp.ActiveDoc
boolstatus = Part.Extension.SelectByID2("", "FACE", 0.1467759532569, 0.2282465512145,
0.2052684525268, True, 1, Nothing, 0)
boolstatus = Part.Extension.SelectByID2("", "FACE", 0.04935040746813, 0.05664959253187,
0.08677583155765, True, 1, Nothing, 0)
Dim myMate As Object
0.08290013687201, 0.1404353580714, True, 1, Nothing, 0)
Set myMate = Part.AddMate3(4, 1, False, 0.08962381358722, 0, 0, 0.001, 0.001, 0.5235987755983,
0.5235987755983, 0.5235987755983, False, longstatus)
Part.ClearSelection2 True
Part.EditRebuild3
boolstatus = Part.Extension.SelectByID2("", "FACE", 0.08743441472234, 0.1639087578332,
0.1920150657936, True, 1, Nothing, 0)
boolstatus = Part.Extension.SelectByID2("", "FACE", 0.146900136872,
Set myMate = Part.AddMate3(4, 1, False, 0.09399852070358, 0, 0, 0.001, 0.001, 0.5235987755983,
0.5235987755983, 0.5235987755983, False, longstatus)
Part.ClearSelection2 True
Part.EditRebuild3
End Sub
    
```


The same procedure was followed for transferring this code to the global form and then to the .bas form. The assembly simulation was divided into several steps in this study. For compiling the command buttons in the user interfaces, two approaches were followed. The first approach is calling the macro for each element by using the subroutine:

```
Public Sub cmdBaseplate_Click ()  
Baseplate.main  
End Sub
```

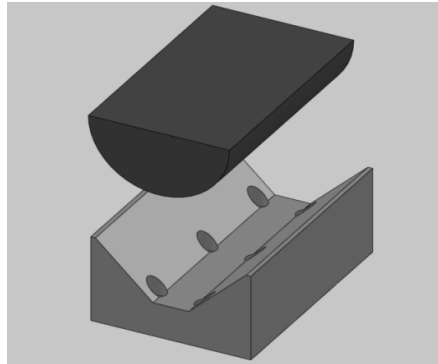


FIGURE 7: The example of the chosen V-block and the workpiece.

This subroutine calls the macro Baseplate by clicking the command button to add this element to the SolidWorks environment. The second approach is writing a code in order to compile the command button:

```
Public Sub cmdBaseplate_Click ()  
Dim swApp As Object  
Dim Part As Object  
Dim boolstatus As Boolean  
Dim longstatus As Long, longwarnings As Long  
Sub main()  
Set swApp = Application.SldWorks  
Set Part = swApp.ActiveDoc  
boolstatus=Part.AddComponent("C:\Users\Desktop\Baseplate.SLDPRT",0.1690002083701,  
0.2903048910654, -0.02491659965801)  
End Sub
```

Both approaches are correct and compatible. To compile these approaches, it is important to attach the macros in .bas format to the master directory of the ActiveX DLL project. The developed menus in the SolidWorks environment are shown in Figure 8.

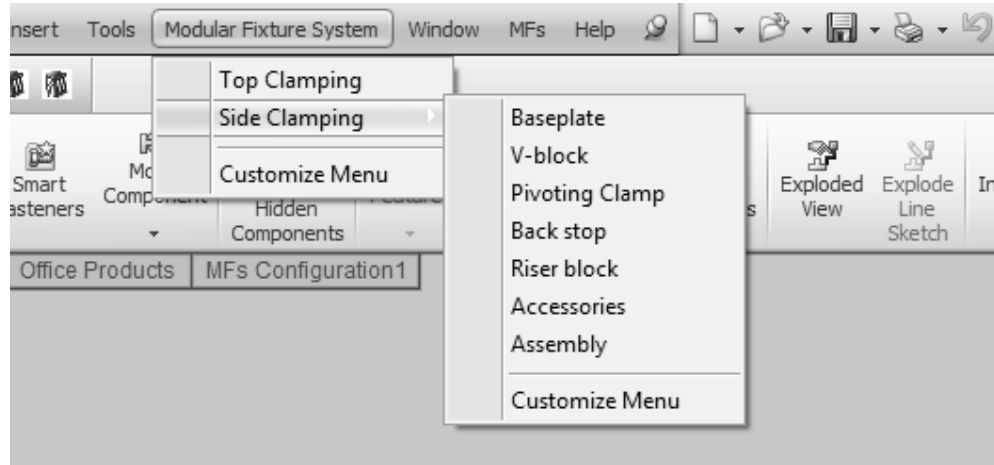


FIGURE 8: The main menu and the extended menus developed for the system.

5. RESULTS AND DISCUSSION

Simulation by macros was employed for developing the user interface for the related fixture elements. Figure 9 shows the selection of the side clamps and the back stops. The interface of V-blocks selection showed in Figure 10 while Figure 11 illustrates the riser blocks interface. Other user interfaces were developed to enhance the system's flexibility. These user interfaces were developed in a different approach because there are more details should be considered for some elements such as the necessary accessories for completing the assembly process for the fixture elements (Figure 12).

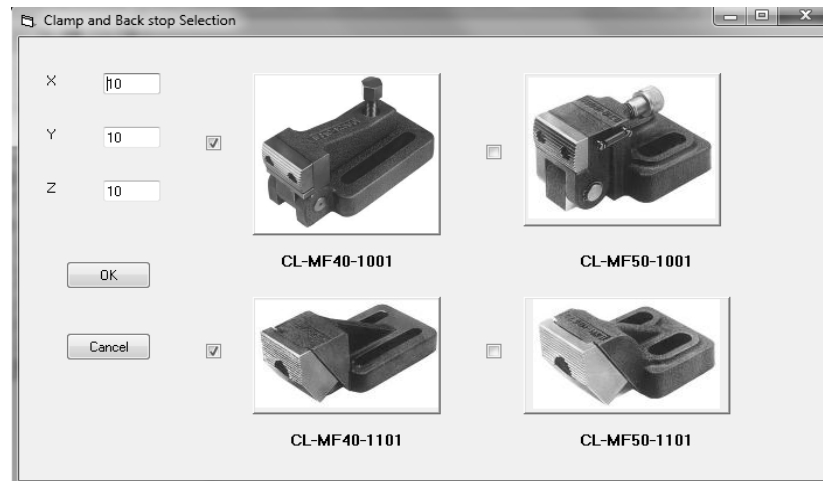


FIGURE 9: The selection window for the side clamps and the back stops.

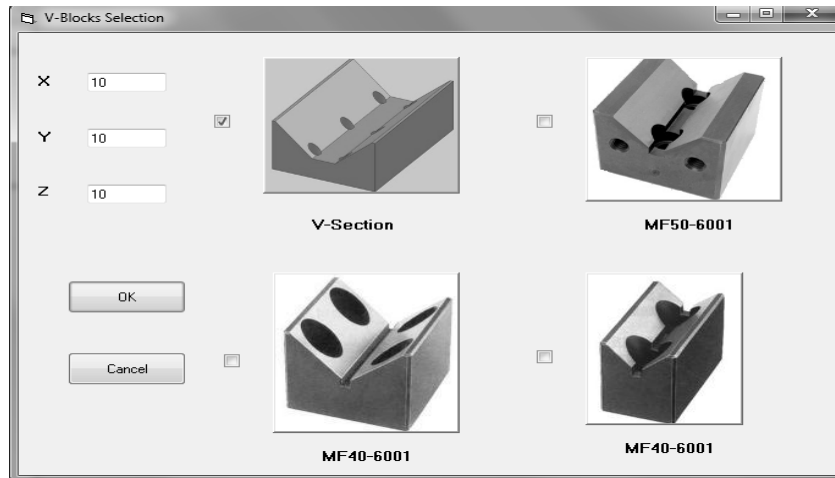


FIGURE 10: The selection window for the V-blocks in the system.

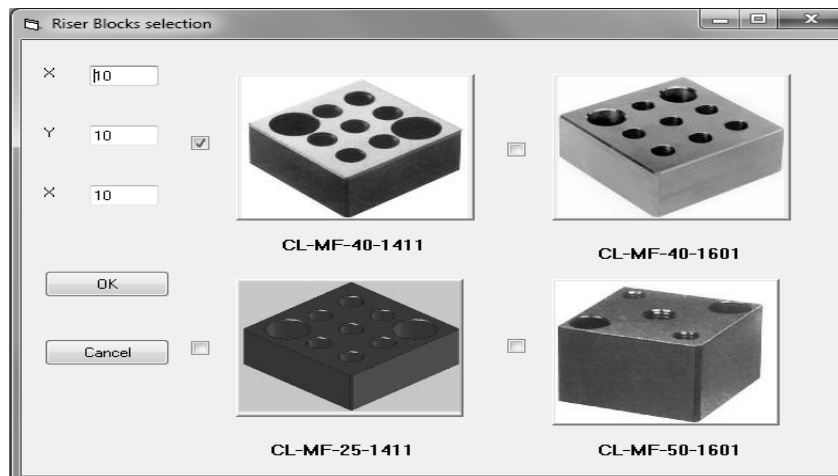


FIGURE 11: The selection window for the riser blocks in the system.

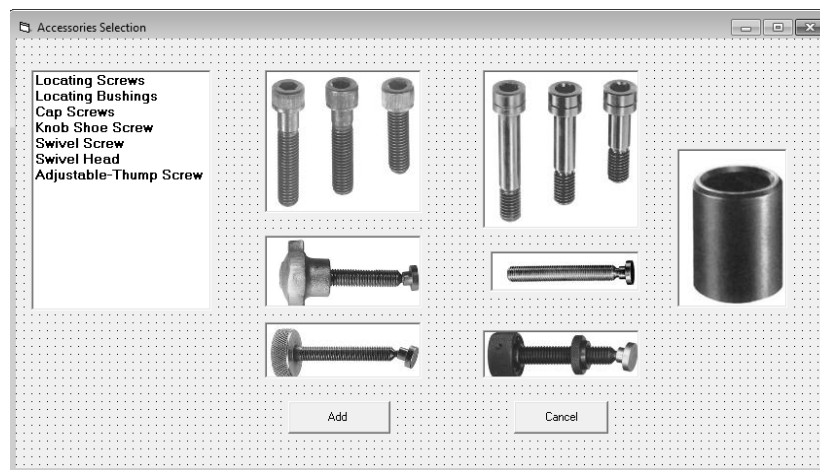


FIGURE 12: The selection window for the accessories to complete the fixturing system.

The assembly process begins after adding the components in the SolidWorks environment as shown in Figure 13. Then, macros were developed for assembling two parts appropriately as explained previously for the workpiece and the V-block. The created macros for the system were used in order to assemble:

- The workpiece with the V-block;
- The V-block with the baseplate;
- The riser block with the baseplate (two macros);
- The pivoting clamp with the riser block;
- The pivoting clamp with the workpiece;
- The back stop with the riser block;
- The backstop with the workpiece.

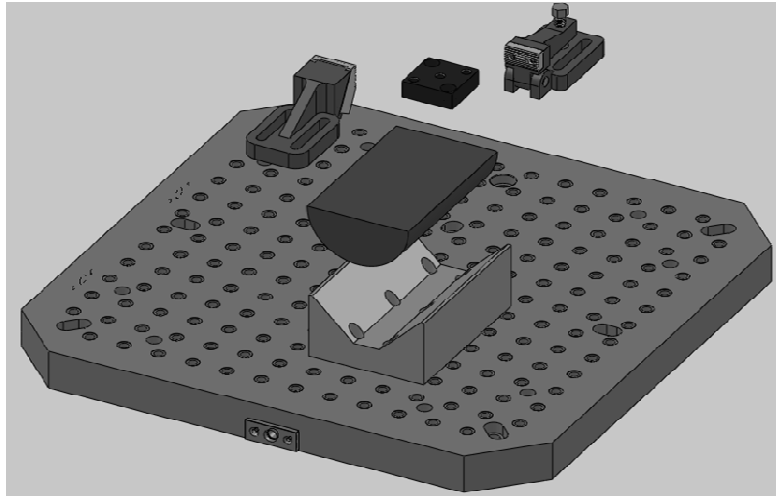


FIGURE 13: Modular fixture elements added in SolidWorks environment from the menus.

Therefore, eight steps were used for completing the assembly simulation. The process was divided into this way because errors could occur if the whole process was one step. This is related to the problems in SolidWorks when applying the functions of rotating and reviewing on the components. The positions and the directions of the elements were predefined in the VB codes. By completing the eight assembly steps, the final MFs layout is shown in Figure 14.

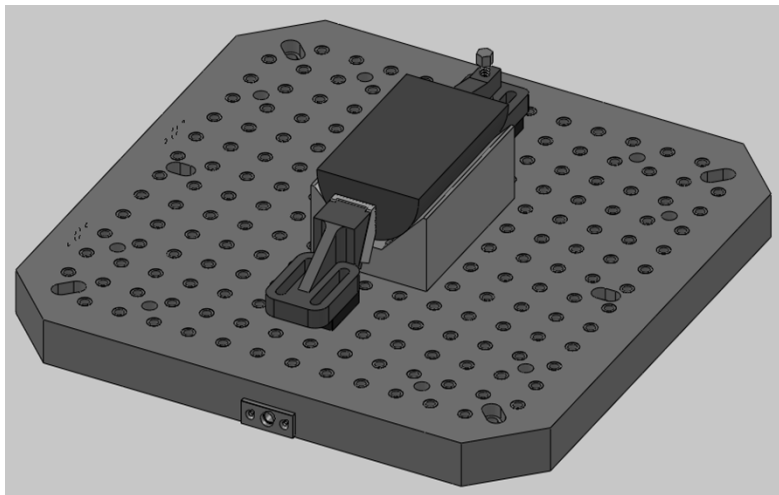


FIGURE 14: The complete layout of modular fixture elements generated after assembly.

Interference detection was running after finishing the assembly to ensure that there is no collision between the fixture components. This process was completed by SolidWorks and if interference is found, the options of mating features such as choosing the correct edges or faces should be modified to achieve the efficient assembly functions.

A layout of MFs was generated for side clamping in this paper. The layout is for a semicircular workpiece and it simulates the assembly process for the fixture elements that should be completed for this kind of workpieces for machining by CNC. This simulation provides the opportunity to overcome the errors that could happen during this process and, therefore, the appropriate MFs layout can be achieved.

Use of SolidWorks in this work provided MFs assembly with a strong support compared to previous studies. This is because the modelling environment and the 3D assembly mating features which make SolidWorks as a powerful CAD software for providing user graphic interface to assemble MFs.

As programming languages such as C++ and AutoLISP were integrated with CAD software to automate MFs assembly, VB 6 was employed in this paper due to its capabilities for building the macros in all Microsoft software and, therefore, accesses the functions of SolidWorks API for developing the graphical user interfaces and new menus. Moreover, VB 6 is considered as a flexible programming language that can be integrated with different applications. Therefore, the secondary development approach presented in this paper enhances and simplifies the automation process of MFs.

6. CONCLUSION

SolidWorks API was employed for automating the assembly process for MFs elements. The simulation of this process was completed by using macros functions in SolidWorks. For this purpose, an ActiveX DLL project was created in VB 6 and a plug-in file in .dll format was generated for developing new menus into SolidWorks environment. From these menus, user interfaces were expanded to be opened by clicking each of these menus allowing the user for selecting and inserting the right elements. The approach of applying SolidWorks API was explained in details for a semi-circular workpiece and can be applied for similar cases. The system was tested by using Interference detection tool included in SolidWorks to be sure that no collision between MFs elements. The developed approach results in saving time and cost for designing and assembling MFs elements.

The future work of this research can be directed towards extending the knowledge base for the assembly process including other fixturing methods and fixture elements to meet the requirements for building several MFs configurations. In addition, a comprehensive automated process can be achieved by extending the VB codes and macros. This leads to defining and modifying the positions and the directions of the fixture elements. Therefore, the errors can be overcome in order to make the process more efficient and in one step rather than many steps.

7. REFERENCES

1. U.H. Farhan and M.T. Rad. "Design of modular fixtures using a 3D-modelling approach". In: 19th International Congress on Modelling and Simulation, 2011, pp.405-411.
2. E.G. Hoffman. "*Jig and fixture design*". New York: Delmar Learning Drafting Series, 2004.
3. G. Peng, X. He, H. Yu, et al. "Precise manipulation approach to facilitate interactive modular fixture assembly design in a virtual environment". *assembly automation*, 28, 2008.
4. F. Mervyn, A.S. Kumar and A.Y.C. Nee. "Fixture design information support for integrated design and manufacturing". *international journal of production research*, 44, 2006.
5. N. Bugtai and R.I.M. Young. "Information models in an integrated fixture decision support tool". *Journal of Materials Processing Technology*, 76, 29-35, 1998.
6. M. Ryll, T.N. Papastathis and S. Ratchev. "Towards an intelligent fixturing system with rapid reconfiguration and part positioning". *Journal of Materials Processing Technology*, 201, 198-203, 2008.
7. J.R. Dai, A.Y.C. Nee, J.Y.H. Fuh, et al. "An approach to automating modular fixture design and assembly". *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 211, 509-521, 1997.
8. B.S. Babu, P.M. Valli, A.V.V. Kumar, et al. "Automatic modular fixture generation in computer aided process planning systems". *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 1147-1152, 2005.
9. X. Kong, Yangyi, J. Zhou, et al. "Research and development of the software on computer-aided fixtures designing". In: 10th International Conference on Computer-Aided Industrial Design & Conceptual Design CAID & CD, 2009, pp.1233-1236.
10. T.C. Chang, R.A. Wysk and H.P. Wang. "*Computer aided manufacturing*". New Jersey: Pearson Education Inc, 2006.
11. M.C. Kerman and R.L. Brown. "*Computer programming fundamentals with applications in visual basic 6.0*". Reading, Mass: Addison-Wesley, 2000.
12. H.M. Deitel, P.J. Deitel and T.R. Nieto. "*Visual Basic 6 how to program*". Upper Saddle River, N.J: Prentice Hall, 1999.
13. "KIPP WORKHOLDING SYSTEMS". Internet:
<http://www.kipp.com/App/WebObjects/XSeMIPSKIIPP.woa/cms/page/locale.enGB/pid.1097.1105.1227/Workholding-systems.html>.
14. S.P. Prince, R.G. Ryan and T. Mincer. "Common API : Using Visual Basic to Communicate between Engineering Design and Analytical Software Tools". In: ASEE Annual Conferenc 2005,
15. M.C. Doo, J.K. Hyung, C.L. Jae, et al. "Web-Based Material Database for Material Selection and its Application Programming Interface (API) for CAD". *Key Engineering Materials*, 345-346, 1593-1596, 2007.

16. S. Bo, Q. Guangtai and F. Yadong. "Research of standard parts library construction for SolidWorks by Visual Basic". In: Electronic and Mechanical Engineering and Information Technology (EMEIT) International Conference, 2011, pp.2651-2654.
17. Y. Peng, J. Xie and X. Wang. "Research and Realization on Architectural 3D Model of Architectural Process Simulation System". In: Second International Workshop on computer science and engineering, 2009, pp.543-547.
18. X. Ning and Q. Jiang. "A digital design method of geometric model for centrifugal fan impeller based on SolidWorks and VB". In: Electronic and Mechanical Engineering and Information Technology (EMEIT), 2011 International Conference on, 2011, pp.4023-4026.
19. S. Danjou, N. Lupa and P. Koehler. "Approach for Automated Product Modeling Using Knowledge-Based Design Features". *Computer-Aided Design and Applications*, 5, 622-629, 2008.
20. J. Tian, S. Liu and H. Fu. "CAD System Design on Standard Part Based on Software Reuse". In: Fourth International Symposium on Knowledge Acquisition and Modeling (KAM), 2011, pp.229-232.
21. M. Zhen and J. Yingyi. "Automatic assembly for combined mold components based on SolidWorks". In: International Conference on Electronics, Communications and Control (ICECC), 2011, pp.166-169.
22. Y. Yang. "The parametric design and intelligent assembly system based on the secondary development of solidworks". In: 2nd International Conference on Computer Engineering and Technology (ICCET), 2010, pp.602-605.