AMO - Advanced Modeling and Optimization, Volume 13, Number 1, 2011

Integration of Pro/Engineer with Visual C++ using Pro/Toolkit for design automation

K.C. Sabareeshan[1*], K. Willmert[1]

[1] Clarkson University, Mechanical & Aeronautical Engineering Department,
8 Clarkson Ave, Potsdam, NY, 13699
* Corresponding Author, Phone: 315-212-2216,e-mail: chitturk@clarkson.edu

Abstract

For years, Computer-Aided Design (CAD) has been a valuable tool in product development. Packages like Pro/Engineer, Unigraphics, CATIA, etc., are some of the most widely implemented CAD packages in industry. They enable creation of virtual three dimensional models of a product which can be used for analysis purposes to optimize the design, and then put into production. This research demonstrates automation of this feature creation operation on a CAD package, by integrating it with a programming language using an Application Program Interface (API). This results in considerable reduction in modeling time, compared to feature creation through the traditional method. However, the learning curve to develop this interface can be quite steep depending on how proficient the developer is with C/C++ programming. Also, more complex features would require sizable programs. The interface developed for this paper was for Pro/Engineer Wildfire 5.0, through C programming using the API Pro/Toolkit which comes standard in Pro/Engineer. The applications developed as a result, successfully demonstrated automation of feature creation on Pro/Engineer through a series of simple inputs from the user. Each application performs a different Pro/Engineer operation. Thus, it opens up a whole range of possibilities in the area of product development and has great potential.

1 Introduction

1.1 Parametrics

Parametric solid modeling is a key technology to define and manipulate solid models. The objective of parametric solid modeling is referred to as Parametrics. Parametrics was first introduced in Parametric Technology's Pro/Engineer® and it has now become a critical capability of most major CAD packages today (Hoffmann and Kim, 2001). In a parametric system, solid models are created and manipulated through parameterized and user-modifiable definitions. A parametric model is defined by its attributes which could be input and/or structural parameters. An input parameter is a variable or relationship that defines a key dimension whereas a structural parameter describes topological and hierarchical similarities between instances (Loukipoudis, 1996).

A large majority of mechanical parts and assemblies are not designed from scratch but are modifications of previously manufactured parts. A parametric model is hence a re-usable model. Most major CAD systems use two approaches for parameterization – Interactive and Programmatic.

An Interactive Approach involves using the native parametric capabilities within the CAD system to create flexible re-usable models. The model can be updated or modified from a common, shared parameter database. Once the model has been defined parametrically, the user allows the CAD program to regenerate or update the model with new values (Rohm, 2000).

A Programmatic Approach involves using a high level programming language to develop parametric models. This method eliminates the user interaction with the CAD package to create the model.

Most major CAD systems in industry have their own macro language or API's as defined in their proprietary native toolkit environment (Rohm, 2001). This approach allows the user to perform all of the functions that the CAD systems graphical user interface offers. This gives the user the benefit of repeatability, handling errorrs, requesting user input and accessing databases.

The programmatic approach has given engineers the ability to modify solid models quickly by changing a few pre-defined parameters and hence reduction in production time. However, despite these benefits, one of the biggest issues is the time consuming learning curve that engineers have to undergo.

Figure 1 illustrates how a programming language interacts with a CAD package through an API toolkit.

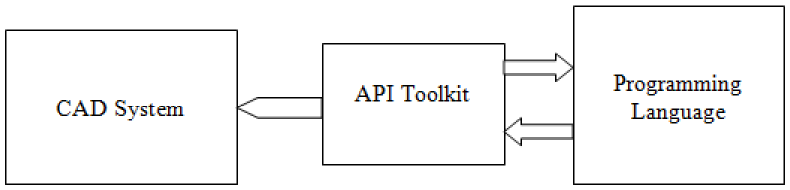

Figure 1: Programming language and CAD interface

1.2 Research Objective

The purpose of this research was to develop applications, using codes written in the C programming language, and integrate these into Pro/Engineer to perform various tasks within Pro/Engineer. Pro/Toolkit, the customization toolkit API for Pro/Engineer, was used to enable this integration.

1.3 Paper Outline

The paper is organized as follows, Section 2 explains how Pro/Toolkit works and talks about its capabilities. Section 3 discusses a custom application that was developed in this research. Finally, conclusions are made in Section 4.

2 Pro/Toolkit

Pro/Toolkit provides functions that enable communication with Pro/Engineer through a program written in the C language with the help of Microsoft Visual Studio C++ (VC++). Each Pro/Toolkit library C function performs an action on a specific type of object. These objects are items in the Pro/Engineer database, such as features and surfaces, etc. Hence, Pro/Toolkit can be used to access most Pro/Engineer operations through these C functions.

2.1 How Pro/Toolkit works

The standard method by which a Pro/Toolkit application code is integrated into Pro/Engineer is called the “Synchronous mode” which uses either dynamically linked libraries (DLLs) or executables (EXEs). In the “DLL mode”, an object library file is created when the C code for the Pro/Toolkit application is complied and linked with the Pro/Toolkit library. This is designed to be linked into the Pro/Engineer executable when Pro/Engineer starts up. In the “EXE mode”, also known as “Spawn or Multi-process mode”, the code is compiled and linked to form a separate executable designed to be spawned by Pro/Engineer and is run as a child process of the Pro/Engineer session.

2.2 Pro/Toolkit capabilities

Pro/Toolkit can be used to develop applications that carry out all the feature creation operations that a user can carry out through Pro/Engineer. The capabilities of Pro/Toolkit are quite vast as it allows access to most Pro/Engineer operations, ranging from feature creation to assembly to developing production applications to Pro/Mechanica functions and Finite Element Modeling (FEM). Hence it can be used to develop various types of applications for Pro/Engineer depending on the need.

In the case of Pro/Mechanica, Pro/Toolkit can be used to develop applications that access properties of Mechanica items. It provides functions that evaluate a model's structural characteristics and thermal profile, and provide powerful tools for examining mechanism performance. The Finite Element Modeling (FEM) functions provided by Pro/Toolkit are designed to give access to data generated by the Pro/MESH module of Pro/ENGINEER.

3 A Pro/Toolkit Application

A Pro/Toolkit application serves as an interface between Pro/Engineer and the user. It essentially communicates to Pro/Engineer what the user wants to perform through a program and hence automates Pro/Engineer operations by having the user input values and parameters that define the feature. Figure 3 illustrates how a Pro/Toolkit application is started and how it works.

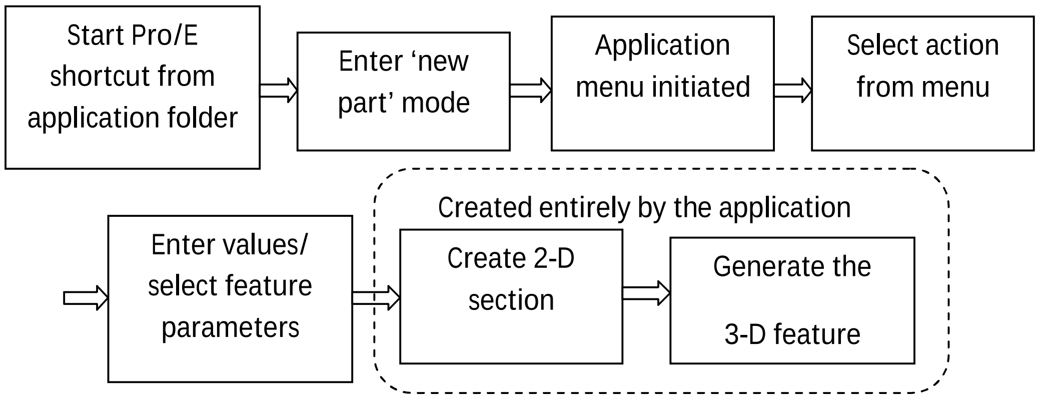
Start Pro/E shortcut from application folder
Enter 'new part' mode
Application menu initiated
Select action from menu
Enter values/ select feature parameters
Created entirely by the application
Create 2-D section
Generate the 3-D feature

Figure 3: Working of a Pro/Toolkit application

An application that automates feature creation in Pro/Engineer first creates a two-dimensional section based on values input by the user and then performs the three-dimensional feature operation on it. Depending on the program parameters that define the sketch, the two dimensional section could be of any shape.

An example application discussed in this paper creates an L-shaped block with a hole. It automates two feature operations - extrude, and hole creation. This application when started creates a custom menu in the Pro/Engineer session. It contains the menu item that starts the creation of the L-block. Upon selection, the application prompts the user to input the values for the appropriate dimensions (vertical length, base length, thickness, diameter of hole and its placement distances). Once these values are entered, the program first calls the function to create the two-dimensional L-shaped sketch and then performs the extrude operation on it. The application then instructs the user to select a surface on the L-block for the placement of the hole, and also asks for two more surfaces from which the hole is offset. Figure 4 shows the output of this application along with its parameters.

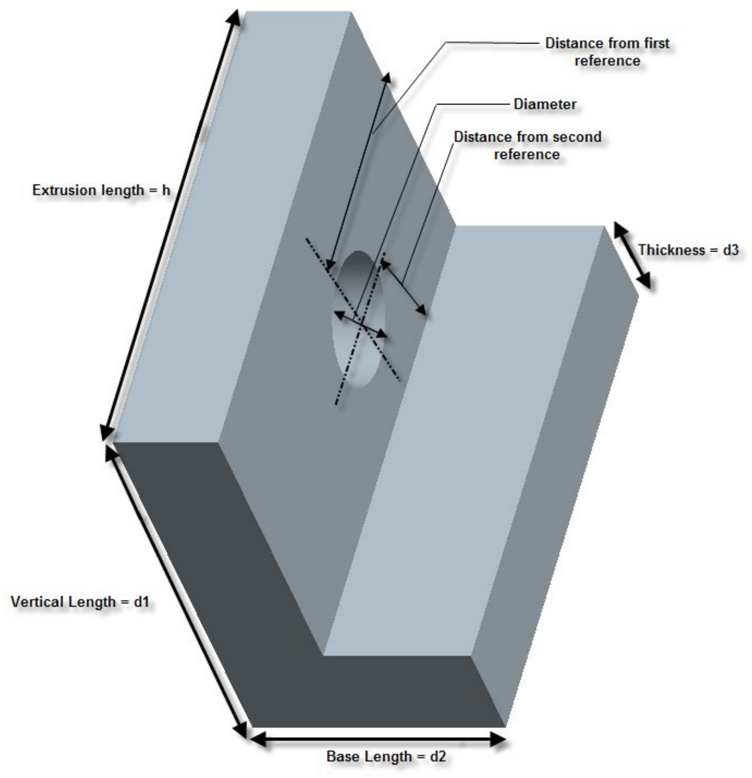

Figure 4: An L-Block with a hole

The code consists of various functions that communicate with each other to carry out the desired actions. For this application there are four main functions. The first one creates the menu; the next function performs the desired action upon selection of a menu item. This second function contains another sub-function that generates the two dimensional section based on the dimensions input.

The following steps are carried out by the code :

1. First, the user is prompted to input values for the dimension of the L-block, the hole and placement parameters for the hole using the functions ProMessageDisplay and ProMessageDoubleRead.

2. The creation of the element tree is then initiated. This tree holds all the information necessary for Pro/Engineer to create a feature, in this case the extrude and hole features.
3. The feature type and form are then defined, thus letting Pro/Engineer know what kind of extrusion and hole are desired.
4. Then, based on the dimensions for the L-block input by the user, a two dimensional sketch is generated by calling the sub-function.
5. Upon creation of this sketch, the extrude operation is performed on it and the three dimensional L-block is displayed to the user.
6. Then the user is prompted to select a surface for the placement of the hole and surfaces from which it is offset.
7. Once these parameters are selected, the hole is created.

The entire code can be found in the M.S thesis entitled ‘Integration of Pro/Engineer with Microsoft Visual C++ using Pro/Toolkit’ at Clarkson University, Potsdam, NY. Other examples are also given in this thesis.

4 Conclusions

The application for feature creation is not only simple to understand and implement, but also reduces the time required to perform the operation to a great extent compared to the traditional method. But the learning curve to develop an application can be steep depending on the developer’s level of programming proficiency. This technique of developing an interface using Pro/Toolkit also provides new openings to explore capabilities in the area of design automation. A possible area would be to link a Pro/Toolkit application with other analysis packages using an external program to create an optimization routine.

References

Hoffmann C., and Kim K., (2001) Towards Valid Parametric CAD Models. Computer-Aided Design, Vol. 33, pp. 81-90.

Loukipoudis, E. N., (1996) Object management in a programming-by-examples, parametric computer-aided system. The Visual Computer, Volume 12, pp. 296-306.

Pro/Engineer Wildfire 5.0 Pro/Toolkit Users guide (2010). Pro/Engineer Wildfire 5.0 Documentation. (Comes with the software package).

Rohm, T., (2001) Graphical Creation of CAD Parametric Application Programs. M.S. Thesis, Brigham Young University.

Sabareeshan, K. C., (2010) Integration of Pro/Engineer with Microsoft Visual C++ using Pro/Toolkit. M.S Thesis, Clarkson University.