EtI ENGINEERING AND TECHNOLOGY IN INDIA
Volume 4 | Issue 2 | Oct., 2013 | 46-52

A CASE STUDY

Customization of CAD/CAM software: a case study of customization of UG/NX 4.0 for modeling couplings using knowledge fusion programming

SUNIL D. WAVALE AND KISHOR P. KOLHE

ABSTRACT : The CAD/CAM software available in the market are general purpose software. These software are not developed for particular user or a particular task. The process of molding of the software for particular application to suit the specific requirement of the customer is called as customization of software. Various industries use UG/NX to perform the task of solid modeling, assembly modeling and drafting of the various engineering products. Due to the wide variety of applications, couplings of various types are used frequently by many industries. The aim of this work is to customize UG/NX CAD/CAM software, to provide facilities that generate three dimensional part model of Oldham's coupling, and its assembly model using knowledge fusion programming.

KEY WORDS : Knowledge based application (KBA), Application programming interface (API), Knowledge fusion (KF), UG/NX 4.0, Open user interface (UI) styler

Article Chronicle : Received : 05.03.2013**; Accepted :** 09.09.2013

How to cite this Article : Wavale, Sunil D. and Kolhe, Kishor, P. (2013). Customization of CAD/CAM software: a case study of customization of UG/NX 4.0 for modeling couplings using knowledge fusion programming. *Engg. & Tech. in India*, **4**(2) : 46-52.

INTRODUCTION

Computer-aided design, Computer-aided manufacturing (CAD/CAM) software available in market are tremendously improving the productivity of designer by facilitating various features that reduce the product development time. However, it is observed that most of the manufacturing industries frequently design and manufacture similar type of components only or the assembly often consist of use of standard parts. The designer has to model the similar products repeatedly in both cases. This is not only time consuming but also it creates fatigue in the designer's work (Kurundkar, 2007). If designer is provided with custom programs for parts and assembly, he has to only input values for key parameters in the dialogue box and the part will automatically get generated.

MEMBERS OF RESEARCH FORUM

Address for correspondence :
SUNIL D. WAVALE, Department of Mechanical Engineering, Imperial College of Engineering and Research, Wagholi, PUNE (M.S.) INDIA
Email: sunildwavale@gmail.com

Coopted Authors :
KISHOR P. KOLHE, Department of Mechanical Engineering, Imperial College of Engineering and Research, Wagholi, PUNE (M.S.) INDIA
Email: kishor_kolhe@rediffmail.com,

Shaft couplings :

Shaft couplings are used in machinery for several purposes, but most common are as follows.

–To provide for the connection of shafts of units those are manufactured separately, like motor and generator, to provide mechanical flexibility, to reduce the transmission of shock load from one shaft to another, to introduce protection overload and to alter the characteristic of rotating unit.

So, due to wide varieties of applications, couplings of various types are being used frequently (Gawali *et al.*, 2005). Due to customization of CAD/CAM software, the efforts in modeling couplings each time are saved, and errors also get reduced due to ready and specialized programmes.

Knowledge based application (KBA) :

Knowledge based application broadly means to build up a system usually called as knowledge based system for solving problems in specific domain (Kurundkar, 2007). A knowledge based system (KBS), is normally in the form of an intelligent computer program, uses knowledge and inference procedure to solve the problems that are difficult enough to require significant human expertise for their solution. The knowledge of knowledge based system consists of facts and heuristics. According to the trend of advanced manufacturing technology and information technology, knowledge based enterprise will become mainstream model.

Design knowledge :

Design knowledge in product development integration system is a generalized concept (Kulkarni, 2009). Design information includes the form information such as engineering document, calculation expressions, CAD data, and informal information such as measurement and tolerance of design, scheme selection, market information, market forecast, gist of decision making, and so on. The result includes the knowledge re-created by product design, such as experience and rules. Market information knowledge includes consumer advices, individualization of demand, market direction, and so on. Design parameters include applied standard, technique parameters and technical requirement, etc. Engineering material knowledge includes handbook, catalog, standard rules include all what is created by field experts.

Knowledge fusion :

NX is now in a position to quickly introduce a significant, new range of smarter capabilities that no other CAD system can currently match (UG/NX help, 2006). It supports virtually all NX capabilities, including modeling of complex geometries features like extrusion, sweeps, and surface through meshes of curves. It also control NX digital simulation,NC programming, tool design and other functions to automate process that span the entire development process. Knowledge Fusion is a superior technology that permits NX to take advantage of engineering knowledge bases in conjunction with rules to deliver powerful applications while permitting knowledge based extension of NX by an end user.

UI input :

The input can be taken directly from user or from database. The objective of the database is to collect and maintain data in a central storage so that it will be available for operations and decision making (Gawali *et al.*, 2005). The objectives of the database are accuracy, integrity, successful recovery from failure, privacy of data and good overall performance. Input errors can also be minimized by database connectivity.

Customizing unigraphics UG/NX 4.0 :

UG/NX provides an automation architecture that is called the common API.

Application programming interface (API) :

As graphics programming languages are generally interpreted, they are fairly slow. Also, because they are not widely used, the implementation of the languages may be less robust than that of C and C++ etc. Graphics languages are, therefore, generally limited to low performance tasks which involve graphics only, or to simple analysis with limited disk file input / output (I/O). If necessary to use system or library routines or functions, it is preferable to use a compiled language such as C++ for system customization (UG/NX help, 2006). The API normally comprises of a library of procedures that may be used for variety of functions. There are two types of API,

System dependent API :

The programming language that comes along with particular CAD/CAM package is called as a system dependent API e.g. AutoLISP for AutoCAD, GRIP for Unigraphics, Pro-Programming for Pro/Engineer, etc.

General purpose API :

The widely used programming languages that are common to every CAD/CAM software, are called general purpose API e.g. C, C++, Visual BASIC, Visual C++, etc.

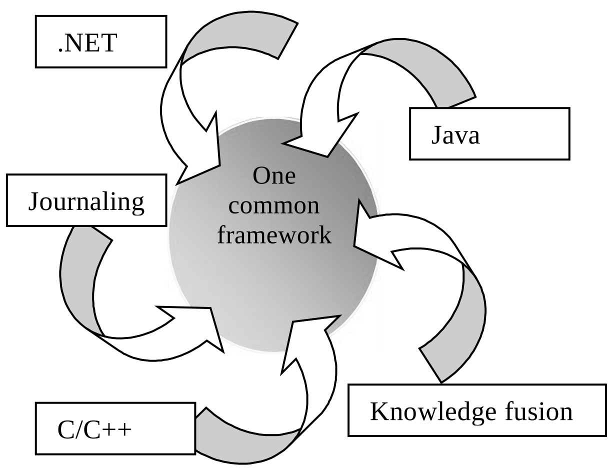


Fig. 1: UG/NX one common API

UG/NX one common API :

NX open for. NET API :

This interface provides programmatic access to NX core application functionality.

NX open for Java :

NX Open for Java is designed for users to take advantage of the benefits of the Java platform.

NX open C++ :

This new C++ library is compatible with the existing Open C and Open C++ APIs.

Open C:

The Open C API is a direct programming interface to NX that allows users to create custom applications.

Journaling :

The Journal utility is a rapid automation tool that records, edits, and replays interactive NX sessions.

NX open GRIP :

GRIP (GRaphics Interactive Programming) is an intermediate scripting language for automating CAD/CAM/CAE tasks.

Knowledge fusion :

This is described in the previous section on KBA.

Front end development :

Open User Interface Styler is a visual dialog box builder for NX users and third-party developers. Open User Interface Styler reduces development time and allows for rapid prototyping. It allows you to easily build NX dialogs according to preset standards, and provides compatibility with MenuScript. You can launch Open User Interface styler dialogs from a MenuScript menu bar. Open User Interface styler has the following features,

–The Object Browser allows you to browse and access each dialog item on your dialog.

–The Resource Editor allows Editor allows you to set or modify the attributes, callbacks, and selection options for any dialog item in your dialog.

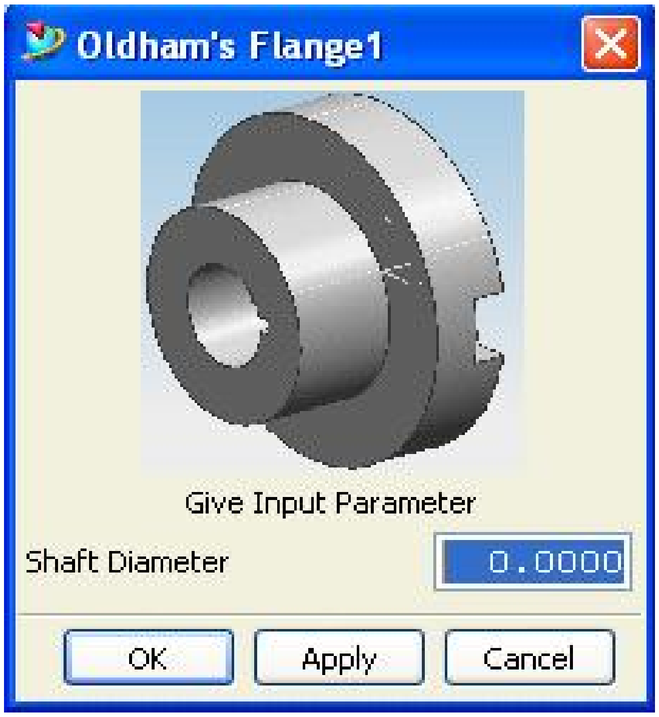
–A detachable toolbar, located on the main window, provides you with the ability to quickly access and execute commands.

–Programmatic Interface: Open C and C++ API functions provide a programmatic interface to the Open User Interface Styler.

User interfaces are developed for modeling various parts and assembly in the current application. Each of the UI includes a bit map image of the part/assembly being modeled and a few dialog box controls like labels, edit box for real data, push buttons, etc. The following UI are developed.

User interface for the first flange :

Shaft diameter is the key parameter which is required to be input by the user. The other parameters are not required to be input.


Oldham's Flange1

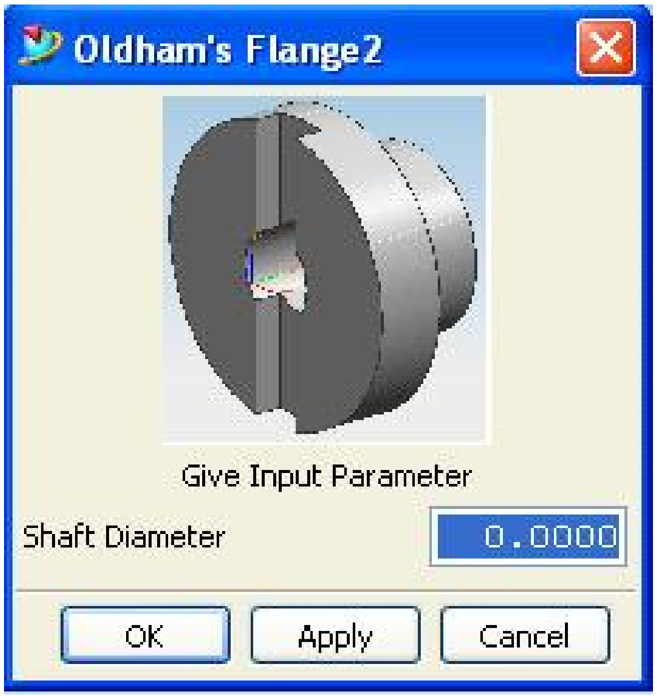
Give Input Parameter

Shaft Diameter 0.0000

Fig. 2: **User interface for Oldham's flange1**

UI for the second flange :

This UI is almost similar to the earlier one except for the orientation of the flange. This can be seen from the programme in Annexure 1 and 2.


Oldham's Flange2

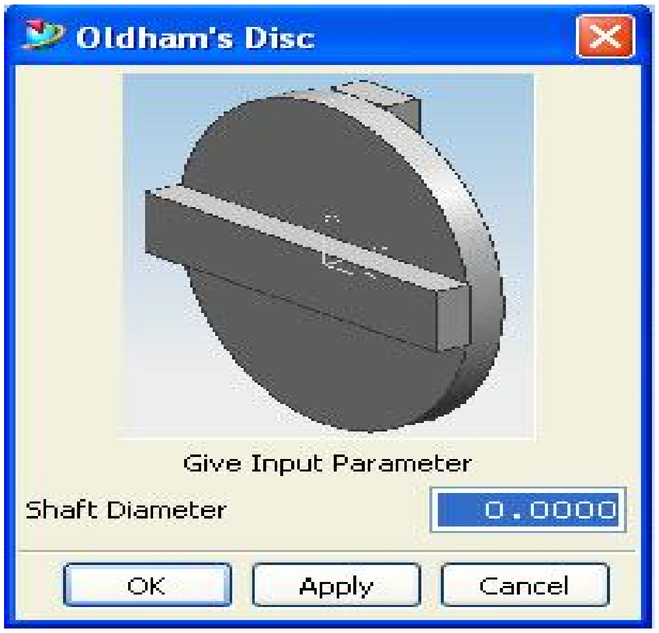
Give Input Parameter

Shaft Diameter 0.0000

Fig. 3: **User interface for Oldham's flange 2**

UI for the disc :

The shaft diameter key parameter is used to compute the other parameters of disc. This is possible due to setting-up of inter-part relations.



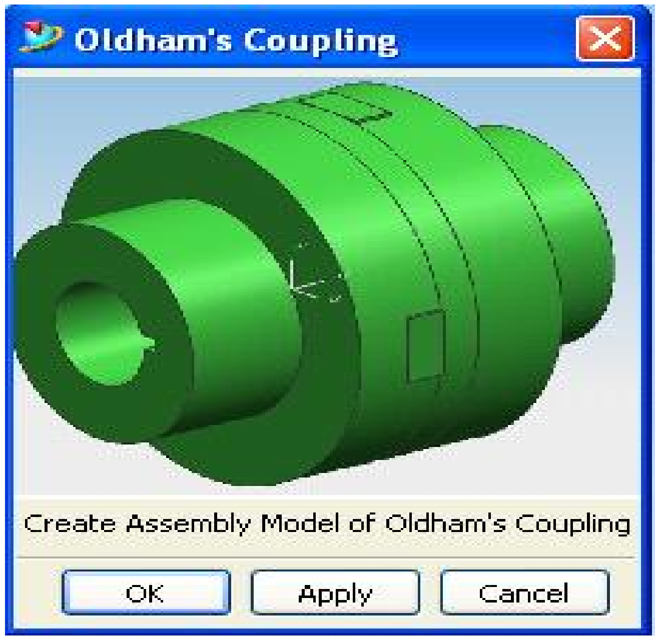
Give Input Parameter

Shaft Diameter 0.0000

Fig. 4: **User interface for Oldham's disc**

UI for the Oldham's coupling assembly :

Use of the "Ok" or "Apply" push button in this dialog box initiates the automated assembly of the parts of the coupling assembly.



Create Assembly Model of Oldham's Coupling

Fig. 5: **User interface for assembly of Oldham's coupling**

Back end development :

Knowledge fusion (KF) is a fully integrated knowledge based engineering (KBE) tool that permits knowledge based extension of NX by the end user. Compared to traditional KBE technologies, the tight integration of knowledge fusion into NX digital product development system provides a significant advantage in the industry. It permits the creation of powerful applications that take advantage of engineering knowledge. It supports the capture and reuse of design intent and user intelligence to increase design speed and productivity, while intelligently controlling change propagation.

Back-end development includes :

–Identifying the parametric relations amongst the various parameters of a part.

–Storing the knowledge about the parametric relations in the KF programme.

–Identification of inter-part relations and use KF to prepare the assembly code.

Fig. 6 shows a sketch of the flange. Symbolic parametric relations used for modeling of the flange are given below,

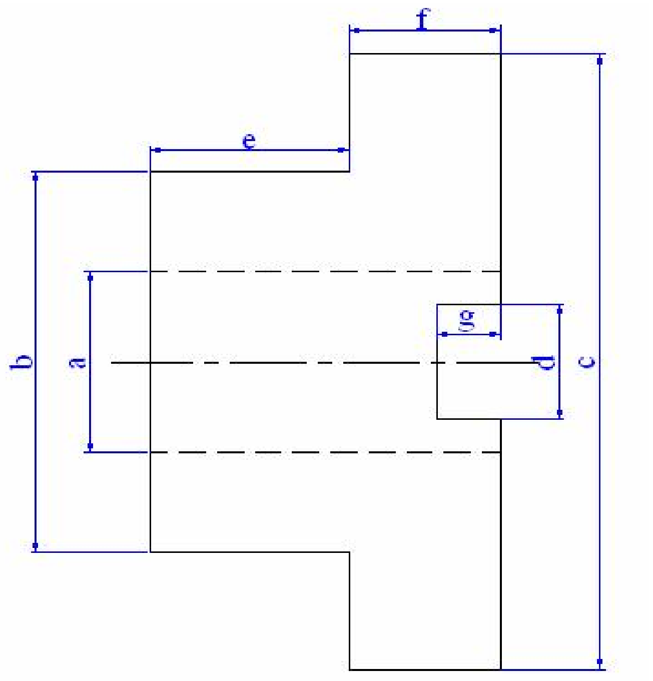


Fig. 6: **Flange of Oldham's coupling**

–Shaft Diameter (a) = ex_diameter:

–Outer diameter of hub (b) = (ex_diameter: * 4)/1.9047

–Rim (c) = (ex_diameter:*4)/1.1764

–Rectangular recess (d) = (ex_diameter: *1.92)/3

–Length of hub (e) = (ex_diameter: *2.28)/3

–Width of Rim (f) = (ex_diameter: * 2.28)/3

–Width of recess (g) = ((ex_diameter: *1.92)/3)/2

Shaft diameter is the key/driving parameter. Other ones are derived from the various relations. Every company may have their own design standards and accordingly, the parametric relations may be unique based on the respective designs.

Samples of such parametric relations and KF program code for the part modeling and assembly are given in Annexure 1-4.

The various solid models developed using KF are same as shown on respective user interfaces (dialogue boxes). See figure 2, 3, 4, 5.

Conclusion :

A user friendly customized software module for generating solid models of parts and assembly models for Oldham's couplings is demonstrated. Similar work is done for a few other types of couplings and can be extended for the remaining types.

This customized software automates the iterative process of modeling of various types of couplings. The time spent by the designer in routine repetitive work is drastically saved and he can use this time in creative aspects of the design process.

ANNEXURE-1 :

Program to Generate Oldham's Flange1

```
    #! UGNX/KF 2.0
    DefClass: oldhamsflange1 (ug_base_part);
    (Canonical Number Parameter Modifiable)
    ex_diameter: UF_STYLER_create_dialog;
    (Canonical Number Parameter Modifiable) ex_height:;
    (Canonical Point Parameter Modifiable) ex_origin:
   Point (0, 0, 0);
    (Child) shaftdia: {
    Class,       ug_cylinder,
    Diameter,    ex_diameter:;
    Height,      (ex_diameter:*2.28)/3.0;
    Origin,      point (0, 0, 0);
    x_axis,      vector (1, 0, 0);
    z_axis,      vector (0, 0, 1);
    Operation,   subtract;
```

Direction, vector (0, -1, 0) :

```
    Target,      {cylinder1 :};
    };
    (Child) cylinder1: {
    Class,       ug_cylinder,
    Diameter,    (ex_diameter:*4)/1.1764;
    Height,      (ex_diameter:*2.28)/3.0;
    Origin,      ex_origin:,
    Direction,   vector (0, -1, 0);
    };
    (Child) cylinder2 :{
    Class,       ug_cylinder,
    Diameter,    (ex_diameter:*4.0)/1.9047;
    Height,      ex_diameter:;
    Origin,      point (0, -(ex_diameter:*2.28)/3.0,0);
    x_axis,      vector (1, 0, 0);
    z_axis,      vector (0, 0, 1);
    Operation,   unite;
    Target,      {cylinder1:, shaftdia:};
```

Direction, vector (0, -1, 0) :

```
    };
    (Child) cylinder3 :{
    Class,       ug_cylinder,
    Diameter,    ex_diameter:;
    Height,      ex_diameter:;
    Origin,      point (0,-(ex_diameter:*2.28)/3.0,0);
    x_axis,      vector (1, 0, 0);
    z_axis,      vector (0, 0, 1);
    Operation,   subtract;
    Target,      {cylinder2 :};
```

Direction, vector (0, -1, 0) :

```
    };
    (Child) key: {
    Class,       ug_block,
    Length,      ex_diameter:/7;
    Width,       ex_diameter:;
    Height,      ex_diameter:/7;
    Origin,      point ((ex_diameter:/2.1),-(ex_diameter :+(
   ex_diameter:*2.28)/3.0),0);
    x_axis,      vector (1, 0, 0);
    z_axis,      vector (0, 0, 1);
    Operation,   subtract;
    Target,
    {Shaftdia:, cylinder1:,cylinder2:,cylinder3:};
```

Direction, vector (0, -1, 0) :

```
    };
    (Child) ex_block: {
    Class,       ug_block,
    Length,      (ex_diameter:*4)/1.1764;
    Width,       ((ex_diameter:*1.92/3)/2);
    Height,      (ex_diameter:*1.92/3);
    Origin,      point((-(ex_diameter:*4)/1.1764)/2,-
   ((ex_diameter:*1.92/3)/2),-((ex_diameter:*1.92/3)/2));
    y_axis,      vector (0, 1, 0);
    z_axis,      vector (0, 0, 1);
    Operation,   subtract;
    Target,{cylinder1:,  shaftdia:,cylinder2:,cylinder3:};
```

Direction, vector (0, -1, 0) :

```
    };
```

ANNEXURE-2

Program to Generate Oldham's Flange2 :

```
    #! UGNX/KF 2.0
    DefClass: oldhamsflange2 (ug_base_part);
    (Canonical Number Parameter Modifiable)
   ex_diameter:UF_STYLER_create_dialog;
    (Canonical Number Parameter Modifiable)ex_height:;
    (Canonical Point Parameter Modifiable)
    ex_origin:    Point (0, 0, 0);
    (Child) shaftdia: {
    Class,        ug_cylinder,
    Diameter,     ex_diameter:;
    Height,       (ex_diameter:*2.28)/3.0;
    Origin,       point (0,(ex_diameter:/3.125),0);
    x_axis,       vector (1, 0, 0);
    z_axis,       vector (0, 0, 1);
    Operation,    subtract;
```

Direction, vector (0, 1, 0) :

```
    Target; {cylinder1 :};
    };
    (Child) cylinder1: {
    Class,        ug_cylinder,
    Diameter,     (ex_diameter:*4)/1.1764;
    Height,       (ex_diameter:*2.28)/3.0;
    Origin,       point (0,((ex_diameter:*1.92/3)/2),0);
```

Direction, vector (0, 1, 0) :

```
    };
    (Child) cylinder2 :{
    Class,        ug_cylinder,
    Diameter,     (ex_diameter:*4.0)/1.9047;
    Height,       ex_diameter:;
    Origin,       point (0, ( ( e x _ diameter : *1.92/3) /
  2)+(ex_diameter:*2.28)/3.0,0);
    x_axis,       vector (1, 0, 0);
    z_axis,       vector (0, 0, 1);
    Operation,    unite;
    Target,       {cylinder1:, shaftdia:};
```

Direction, Vector (0, 1, 0) :

```
    };
    (Child) cylinder3 :{
    Class,        ug_cylinder,
    Diameter,     ex_diameter:;
    Height,       ex_diameter:;
    Origin,       point(0,((ex_diameter:*1.92/3)/
  2)+(ex_diameter:*2.28)/3.0,0);
    x_axis,       vector (1, 0, 0);
    z_axis,       vector (0, 0, 1);
    Operation,    subtract;
    Target,       {cylinder2 :};
```

Direction, vector (0, 1, 0) :

```
    };
    (Child) key: {
    Class,        ug_block,
    Length,       ex_diameter:/7;
    Width,        ex_diameter:;
    Height,       ex_diameter:/7;
    Origin,       point ((ex_diameter:/2.1),
   ((ex_diameter:*1.92/3)/2) + (ex_diameter:*2.28)/3.0, 0);
    x_axis,       vector (1, 0, 0);
    z_axis,       vector (0, 0, 1);
    Operation,    subtract;
    Target,  {shaftdia:,cylinder1:,cylinder2:,cylinder3:};
```

Direction, vector (0, 1, 0) :

```
    };
    (Child) ex_block: {
    Class,        ug_block,
    Length,       (ex_diameter:*1.92/3);
    Width,        ((ex_diameter:*1.92/3)/2);
    Height,       (ex_diameter:*4)/1.1764;
    Origin,       point (((ex_diameter:*1.92/3)/2),
   (ex_diameter:*1.92/3)/2, (-(ex_diameter:*4)/1.1764)/2);
    x_axis,       vector (1, 0, 0);
    y_axis,       vector (0, 1, 0);
    Operation,    subtract;
    Target,       {cylinder1:, shaftdia:,cylinder2:,cylinder3:};
```

Direction, vector (0, 1, 0) ;

```
     };
```

ANNEXURE-3

Program to Generate Oldham's Disc :

```
    #! UGNX/KF 2.0
    DefClass: disc1 (ug_base_part);
    (Canonical Number Parameter Modifiable)
   ex_diameter:UF_STYLER_create_dialog;
    (Canonical Number Parameter Modifiable) ex_height:
    (Canonical Point Parameter Modifiable) ex_origin:
    Point (0, 0, 0);
    (Child) cylinder1: {
    Class,        ug_cylinder,
    Diameter,     (ex_diameter:*4)/1.1764;
    Height,       (ex_diameter:/3.125);
    Origin,       ex_origin:;
    Direction,    vector (0, 1, 0);
    };
```

(Child) block1: {

```
    Class,        ug_block,
    Length,       (ex_diameter:*4)/1.1764;
    Width,        ((ex_diameter:*1.92/3)/2);
```

```
      Height,      (ex_diameter:*1.92/3);
      Origin,      point((-(ex_diameter:*4)/1.1764)/2,-
           ((ex_diameter:*1.92/3)/2),-((ex_diameter:*1.92/3)/
2));
      y_axis,      vector (0, 1, 0);
      z_axis,      vector (0, 0, 1);
      Operation,   unite;
      Target,      {cylinder1 :};
      Direction,   vector (0, -1, 0);
      };
```

(Child) block2: {

```
      Class,       ug_block,
      Length,      (ex_diameter:*1.92/3);
      Width,       ((ex_diameter:*1.92/3)/2);
      Height,      (ex_diameter:*4)/1.1764;
      Origin,      point(-((ex_diameter:*1.92/3)/
    2),(ex_diameter:/3.125),-((ex_diameter:*4)/1.1764/2));
      y_axis,      vector (0, 1, 0);
      z_axis,      vector (0, 0, 1);
      Operation,   unite;
      Target,      {cylinder1 :};
      Direction,   vector (0, -1, 0);
      };
```

ANNEXURE-4

Program to Generate Assembly of Oldham's coupling **:**

```
      #! UGNX/KF 2.0
      DefClass: oldhamscoupling (ug_base_part);
      (Canonical Number Parameter
Modifiable)ex_diameter:;
      (Canonical Number Parameter Modifiable)ex_height:
      (Canonical Point Parameter Modifiable)ex_origin:
       Point (0, 0, 0);
      (String) FName: "a.prt";
      (String) FName: "b.prt";
      (String) FName: "c.prt";
      (String) CName: "oldhamscoupling";
      (String) flangepartname: "oldhamsflange1";
      (String) flangepartname: "oldhamsflange2";
      (String) flangepartname: "disc1";
      (Child) oldhamsflange1: {
      Class, ug_component;
```

File_Name, "a.prt";

```
      Reference_Set_Name,"MODEL";
      Component_Name, "oldhamsflange1";
      Origin, Point (0, 0, 0);
      X_axis, Vector (1, 0, 0);
      Y_axis, Vector (0, 1, 0);
      };
      (Child) oldhamsflange2: {
      Class, ug_component;
```

File_Name, "b.prt";

```
      Reference_Set_Name,"MODEL";
      Component_Name, "oldhamsflange2";
      };
      (Child) disc1: {
      Class, ug_component;
```

File_Name, "c.prt";

```
      Reference_Set_Name,"MODEL";
      Component_Name, "disc1";
      };
```

REFERENCES

Bhatt, N.D. and Panchal, V.M. (2006). *Machine drawing.* Charotar Publishing House.

Gawali, S., Mundhe and Vinkare, S.K. (2005). Customization of CATIA for coupling design Visual BASIC API; B.E. (Prod. Engg.) Project Report, SGGSIE&T, Nanded.

Jayachandran, S.S. and Narayanan, S. (1996). Programme to draw threads using computer aided design; *Industrial Engg. J. India*, **25** (4): 1-4.

Khan, A.N. Shiraz (2007). Customization of CATIA for geometric modeling of fly wheels using VB API; M.Tech. dissertation, SGGSIE&T, Nanded.

Kulkarni, S.R. (2009). Customization in UG/NX4.0 for gear modeling; M.Tech. dissertation, SGGSIE&T, Nanded (M.S.) INDIA.

Kurundkar, R.C.(2007). A knowledge driven automation for selection procedure and computer aided design of bearing ; M. Tech. dissertation, SGGSIE&T, Nanded (M.S.) INDIA.

UG/NX4 Documentation help (2006).