International Journal of Scientific Engineering and Technology Research

ISSN 2319-8885
Vol.04,Issue.08,
April-2015,
Pages:1421-1426

www.ijsetr.com

Software Development (Customization) in UG/NX 4.0 for Coupling Module using Knowledge Fusion Programming

WAVALE S.D[1], GAWAI U.S[2], KATKAM V.D[3]

[1]Assistant Professor, ICOER, Pune, India, E-mail: sunildwavale@gmail.com.
[2]Assistant Professor, ICOER, Pune, India, E-mail: ujwalgawai14@gmail.com.
[3]Assistant Professor, ICOER, Pune, India, E-mail: vineethkatkam@gamil.com.

Abstract: The aim of this work is to develop customized software, which generates three dimensional coupling part models, assembly models and to develop easy user interface in UG/NX4.0 using knowledge Fusion programming. This will save modeling time and speed up the work of designer.

Keywords: Knowledge Based Application (kBA), Application Programming Interface (API), Knowledge Fusion (KF), UG/NX 4.0, Open User Interface (UI) Styler.

I. INTRODUCTION

The CAD software that is available in market is tremendously improving the productivity of designer by facilitating various features that are reducing the product development time. However, it has been observed that most of the manufacturing industries produce similar type of component only or the assembly consist of maximum use of standard parts. Then designer has to model the similar products repeatedly in both cases. This is not only time consuming but also it produce fatigue to the designer. If designer creates automation of standard parts wants specific part he has to only put values in the dialogue box, the part will automatically get generated.

Shaft Coupling: Shaft Couplings are used in machinery for several purposes, but most common are as follows.

- To provide for the connection of shafts of units that are manufactured separately such as motor and generator and provide disconnection for repairs and alternatives.
- To provide misalignment of shaft or to introduce mechanical flexibility.
- To reduce the transmission of shock load from one shaft to another.
- To introduce protection overload.
- To alter the characteristic of rotating unit.
- So, due to wide varieties of applications coupling of various types being used frequently. Due to customization the efforts in modeling coupling each time is saved, and manual errors are also get reduced due to correct programming.

II.CUSTOMIZING UNIGRAPHICS NX4.0

NX provides an automation architecture that serves as the foundation for all NX APIs as well as a new journaling utility. Called the Common API, this foundation combines the power of journaling and automation with the freedom of a language–neutral platform.

NX Open Common Application Programming Interface:

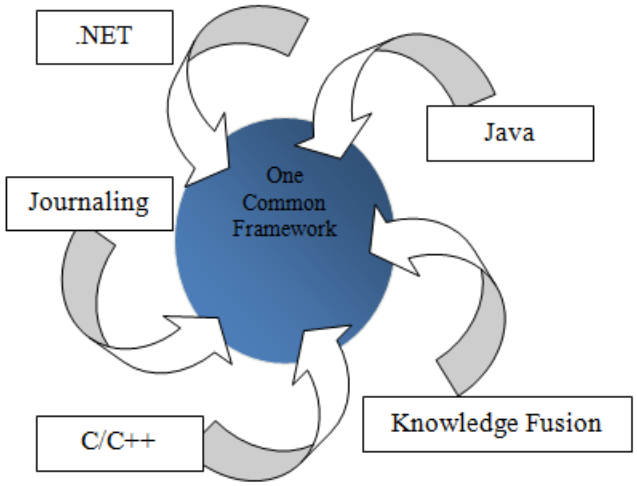

Fig.1.

A. NX Open For .NET API

A New API for use with Microsoft's .NET framework. Built on the Common API, this interface provides programmatic access to NX core application functionality, making it possible to create advanced automation programs using any of the .NET–compliant languages, including Visual Basic.NET and C#. Because the API is built on the .NET framework, users can take full advantage of all the benefits provided by that framework.

B. NX Open for Java

A new API derived from the Common API and built for use with the Java platform. NX Open for Java is designed for

users who want to write advanced NX Open automation programs while taking advantage of the benefits of the Java platform.

C. NX Open C++

A new C++ object oriented library derived from the Common API. This new C++ library is 100% compatible with the existing Open C and Open C++ APIs, and is the product through which users can enhance their existing Open C/C++ programs with calls to new NX functionality.

D. Open C

The Open C API (a.k.a. User Function) is a direct programming interface to NX that allows users to create custom applications using the popular C programming language. It is used by NX developers, customers, and alliance partners to produce unique applications to augment NX or to act as completely separate utilities. Open C also provides a fully extensible data model, allowing customers to define new types of objects that can be treated just like standard NX objects and stored persistently in NX part files.

E. Open C++

Open C++ is an object–oriented interface to NX. Written in C++, this API takes full advantage of object–oriented features including inheritance, encapsulation and polymorphism. Open C++ provides complete access to its class hierarchy, allowing customers to override methods, derive their own classes, and create entirely new, persistent objects in NX. Open C++ is fully compatible with the existing Open C API.

F. Journaling

The Journal utility is a rapid automation tool that records, edits, and replays interactive NX sessions. Built from the Common API and based on the programming language Visual Basic .NET, it produces a scripted file from an interactive session of NX which can be run at a later time to replay the session. These sessions can be edited and enhanced with simple programming constructs and user interface components to produce a rapidly–generated customized program. All enhancements are provided through the Common API. As a consequence, these enhancements are available to all libraries derived from the Common API. By providing a single automation framework for all of our automation interfaces, our automation strategy is consolidated into one common, consistent and stable approach for providing customization tools for our users. In addition to the Common API toolkits provided above, UGS provides the following automation tools.

G. NX Open GRIP

GRIP (Graphics Interactive Programming) is an intermediate scripting language for automating CAD/CAM/ CAE tasks. Users can create applications to automate Numerical Control (NC) operations, create geometric and drafting objects, control system parameters, perform file management functions and modify existing geometry. GRIP is a legacy API which is maintained but not actively enhanced.

H. NX Open User Interface Styler

User Interface Styler is a visual user interface builder that makes it possible to interactively design portable NX–style dialogs. Used both internally by UGS developers and externally by users and third–party developers, User Interface Styler provides the application module, dialog builder, objects, libraries and documentation necessary to interactively create production ready dialogs. The User Interface Styler is compatible with MenuScript and can be associated with an action in a MenuScript ".men" file. Thus, User Interface Styler dialogs can be launched from a MenuScript menubar. Information about the User Interface Styler product can be found in the Tools section of the online Help.

III. MENU SCRIPT

MenuScript is a tool that allows end users and third–party developers to use ASCII files to edit NX menus, and create custom menus for their own applications in an integrated, seamless manner. Menu files support custom tailoring of the main menu bar and the Quick View Popup menu. Customers can create specialized menus and user interface dialogs, exposing and augmenting only the NX functions required in the custom workflow process. Information about the MenuScript product can be found in the Tools section of the online Help.

IV. KNOWLEDGE FUSION

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 as shown in Figs.2 to 10.

A. Front End Development

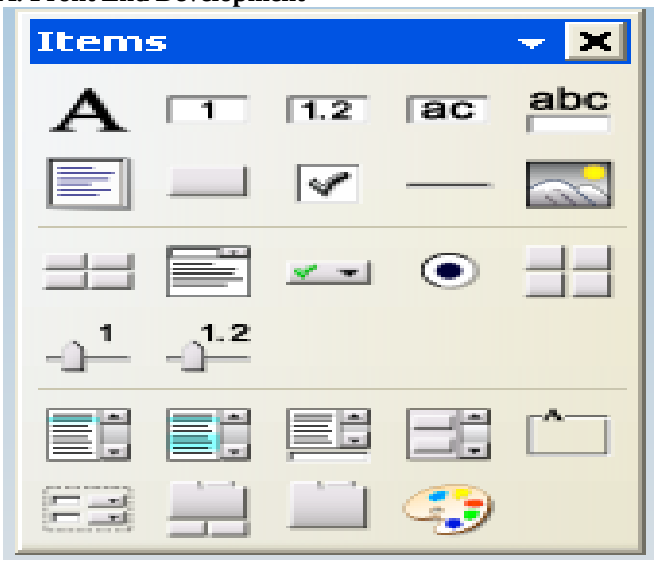

Fig.2. Open UI Styler Toolbar.

Open User Interface Styler is a visual dialog box builer for NX users and third-party developers. Open User Interface Styler reduces development time and allows for rapid prototyping because of the visual builder and automatic file generation, allows you to easily build NX dialogs according to preset standards, and provides compatibility with MenuScript; you can launch Open User Interface styler

dialogs form a MenuScript menubar. Open User Interface styler has the following features. Open User Inetface Styler is a visual dialog builder that allows you to visually select and add dialog items to your dialog.

- 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 sttributes, 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 qickly access and execute commands.
- **Programmatic Interface:** Open C and C++ API functions provide a programmatic interface to the Open User Interface Styler. For more information, see the open C and C++ API Reference manual or the UF_STYLER.H header file.

Fig.3. User Interface for Oldhams Flange1.

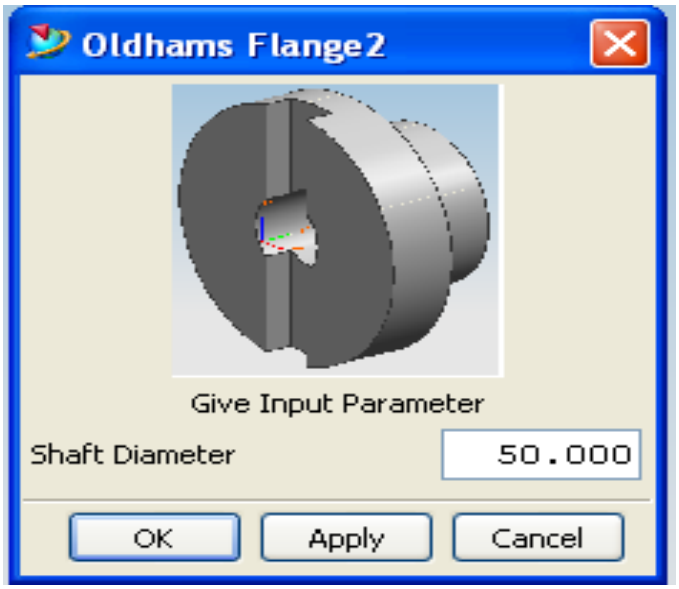

Fig.4. User Interface for Oldhams Flange2.

Fig.5.User Interface for Oldhams Disc.

Fig.6. User interface for assembly of Oldhams Copling.

Various User Interfaces Developed Using Open UI Styler.

B. Back End Development

Knowledge Fusion 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. Knowledge fusion 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 desing speed and productivity.while intelligently controlling change propogation.

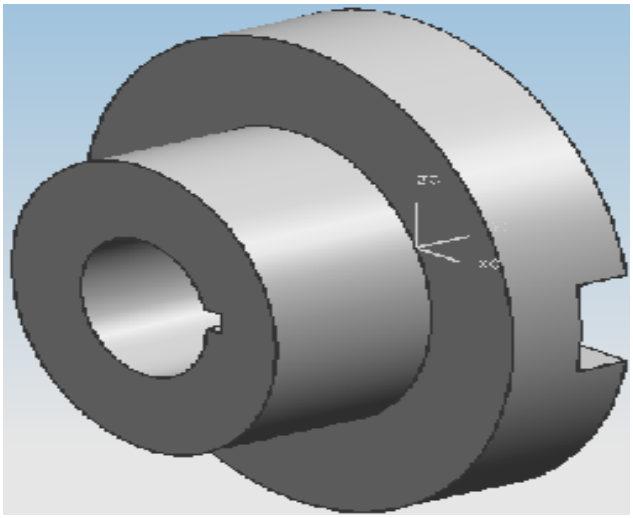

Fig.7. An Oldhams Flange1 Part is Modeled Using KF programming.(see ANNEXURE-1).

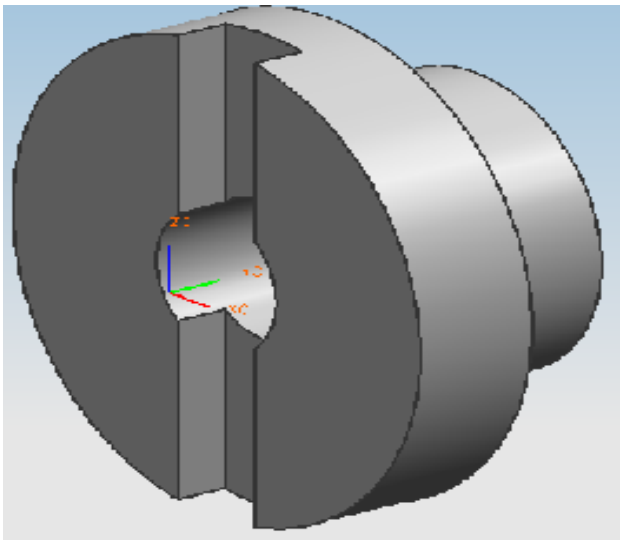

Fig.8. An Oldhams Flange2 is Part Modeled Using KF Programming. (see ANNEXURE-2).

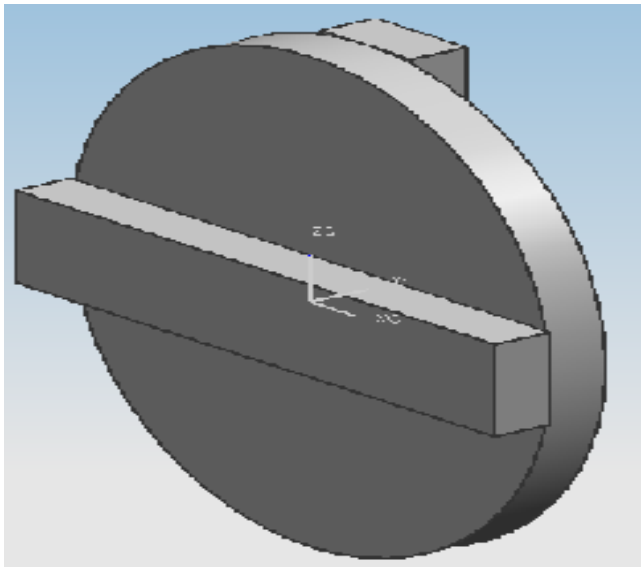

Fig.9. An Oldhams Disc Part is Modeled using KF Programming. (see ANNEXURE-3).

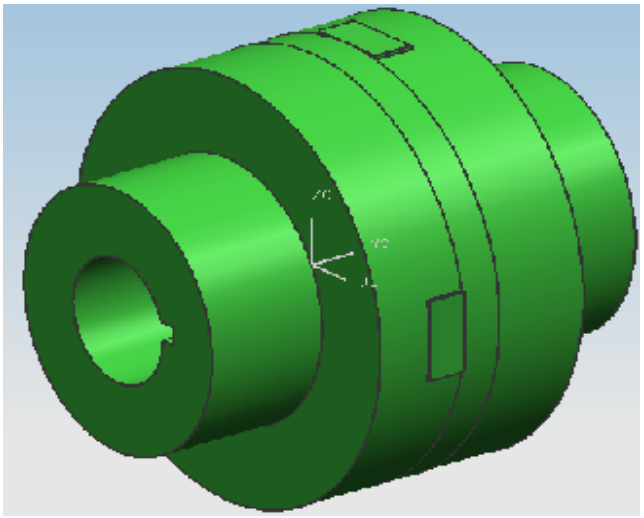

Fig.10. An Assembly of Oldhams Coupling is modeled Using KF programming. (see ANNEXURE-4).

V. CONCLUSION

A method for generating solid part modeling, solid assembly modeling for Coupling module is developed. This customized software automates the iterative process of coupling modeling. the time spent by the designerin routine repetitive work is drastically saved and he can use this time in creative aspect of the design process.

VI. REFERENCES

[1] S.S. Jayachandran & S. Narayanan "Program To Draw Threads Using Computer-Aided Design" Industrial Engineering Jouranal (india)vol XXVNO.4 April 1996 pp1-4.
[2] Ian Sommerville, "Software Engineering" Pearson Education 6th Edition.
[3] Bhatt N.D., Panchal V.M.,"Machine Drawing" Charotar Publishing House,India 41st edition 2006.
[4] UG Documentation help;Unigraphics solution Inc,2004.
[5] Kurundkar R.C.; "A knowledge Driven Automation For Selection Procedure & Computer Aided Design of bearing", M.Tech dissertation,sggsie&t.Nanded.2006-07.

ANNEXURE-1

PROGRAM TO GENERATE OLDHAMS 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 OLDHAMS 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,((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,    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_diam
eter:*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 OLDHAMS 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 OLDHAMSCOUPLING

```
#! 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";
};
```