

# A Novel Method of Using API to Generate Liaison Relationships from an Assembly

Arun Tom Mathew<sup>1\*</sup>, C. S. P. Rao<sup>2</sup>

<sup>1</sup>Research Scholar, Department of Mechanical Engineering, National Institute of Technology, Warangal, India; <sup>2</sup>Professor, Department of Mechanical Engineering, National Institute of Technology, Warangal, India.  
Email: aruntom123@gmail.com

Received November 17<sup>th</sup>, 2009; revised December 3<sup>rd</sup>, 2009; accepted December 20<sup>th</sup>, 2009

## ABSTRACT

*A mechanical assembly is a composition of interrelated parts. Assembly data base stores the geometric models of individual parts, the spatial positions and orientations of the parts in the assembly, and the relationships between parts. An assembly of parts can be represented by its liaison which has a description of its relationships between the various parts in the assembly. The problem is to not only make the information available but also use the relevant information for making decisions, especially determination of the assembly sequence plan. The method described in this paper extracts the feature based assembly information from CAD models of products and build up liaisons to facilitate assembly planning applications. The system works on the assumption that the designer explicitly defines joints and mating conditions. Further, a computer representation of mechanical assemblies in the form of liaisons is necessary in order to automate the generation of assembly plans. A novel method of extracting the assembly information and representing them in the form of liaisons is presented in this paper.*

**Keywords:** Assembly, Mate Entities, Liaisons, Solidworks API

## 1. Introduction

The deployment of product models for planning assembly processes has received significant attention over the years and considerable research is happening in the area of assembly planning over the years. But assembly planning still poses a challenge like the description of assembly data and information specifically. There is much interest in reducing the cost of assembly activities. Assembly costs account for 10–30% of total industrial product labor costs [1], and as much as 50% of product manufacturing cost [2,3]. One way of achieving this is to improve assembly planning which aims to identify and evaluate the different ways to identify and evaluate the different ways to construct a mechanical object from its component parts. Due to frequent changes in product design and manufacturing methods, it is desirable to automate and computerize the planning activity. Sequence generation plays an important role in designing and planning the product assembly process. The choice of the assembly sequence in which parts of assembly are put together can drastically affect the efficiency of the assembly process. Identifying part interdependencies in assemblies and planning the process of assembly are examples of complex decision making activities. Assemblies contain a very large amount of information and

complex relationships. An assembly planner is a system based on the geometric description of an assembly model identifies the parts that are involved in the construction of the assembly and generates the assembly plan. The model should provide a representation of parts and relationships such as contacts, degree of freedom among parts of assembly. Relational models represent geometric relations in the form of mating features between individual parts or subassemblies called liaisons. Of late, these parts or subassemblies are being designed using CAD programs, therefore the shape of each part and geometric information are already available in computer database. Since an effective description of assembly knowledge is very necessary, this information if extracted will be beneficial in identifying interdependencies between parts of the assembly and represent them in the form of a liaison diagram. In this paper, a novel approach of using the Automatic Programmable Interface (API) of the CAD software to extract the information which is then used to generate the liaison diagram which will be useful to generate assembly plans more efficiently.

The organization of the paper is as follows: Section 2 presents a literature review of assembly sequence generation; Section 3 describes the Generation of Assembly Relationships. Section 4 summarizes a different approach to the generation of assembly relationships using API

and building of a liaison diagram; Section 5 describes the system interface and the algorithm to extract relationships and generate liaisons; Section 6 gives an example of the approach and system and Section 7 gives concluding remarks.

## 2. Literature Survey

Over the years, a considerable progress has been made in the area of assembly planning specifically in the generation of assembly sequences. In general, assembly sequence planning consists of assembly modeling and assembly sequence generation. A lot of relevant information regarding the assembly could not be modeled and stored in the product while the assembly operation is done. The efficiency of assembly planning depends on the way the assembly information is modeled. Assembly information modeling is the base in this research, for generating assembly sequences. In the modeling of an assembly, the relation between the connected components must be established. The most commonly used method of assembly modeling is graph based called part mating graph [4] which represents the topological relationship between components of the assembly, where in the nodes represent the components and the arcs establish the relationship between the components. The mating conditions between two components provided by the designer are captured by a Virtual link mating graph [5]. Relational model graph includes parts, contacts and attachment relationships in a model [6]. De Fazio and Whitney called these mating graphs as liaison graphs [7]. Commercial CAD systems interpret assembly modeling as a means of providing functionality to the designer to easily position components with respect to each other [8]. Various detailed assembly representations have evolved including feature based [9], kinematics based [10] and geometry based [11]. Gottipolu and Ghosh [12,13] generated relationships by analyzing contact and mobility constraints. Laperriere and ElMaraghy [14] generated relationships using geometric and accessibility constraints. Generation of assembly relationships was also attempted using solid models by Chang [15]. Linn and Liu [16] described an algorithm developed to identify part liaison relationships presented in the commercial package, I-DEAS where the program processes geometric and topological data. Completely disassembling an assembly component based on the geometric contact relations results in the components explosion graph. Chen used the "contact above" concept to construct the Above Graph and then derive the Explosion Graph [17]. In the joint-based method [18], the assembly constraints are assigned on the components, but not on the geometric elements of the components. The method generates assembly models from kinematic joint constraints by extracting feasible joint mating features for each mating component, and then generates the assembly configuration for a set of joint constraints. Geometry-based representations capture

the surface mating constraints like fit, coplanar, etc to establish the relations of precedence and feasibility [19]. The connection-semantics based assembly tree hierarchy provides a way to consider both geometric information and non-geometric knowledge of the assembly and obtain the degree of freedom of the mating entities [20]. Product semantic information model which is made of semantic information is structured into a three level semantic abstract, from which the relevant information is retrieved [21]. During assembly operation the designer specifies the relative location and the orientation of the components and surface mating constraints in order to accomplish the desired joints. All information regarding relationships between parts should be captured and used during the assembly planning process. In this paper, a novel method to extract this information between the assembly, subassembly and components using the API of the CAD modeling package and building the liaison matrix of the assembly is proposed.

## 3. Generation of Assembly Relationships

SolidWorks, the commercial CAD system, is used as the main feature-based design environment. The benefit of using SolidWorks is, it includes an entire API with functions that can be called from Visual Basic. In addition, SolidWorks shares the same solid modeling engine as Unigraphics and several other CAD systems like the Pro/Engineer and Catia. In addition, these CAD systems account for large user and application bases.

The description of the relationships among the features of various parts is required for an assembly component. These features can be classified into assembly features and primal features. It is the primal features that participate in assembly constraints. The assembly module automatically determines which relationship is meant by the user based upon the features involved in the relationship and updates the degrees of freedom accordingly. The primary mating conditions are align, mate, mate entity, align offset, insert, orient etc. The align condition requires that the axial centre lines of two parts be collinear. The mate condition requires that the two mating faces lie in the same plane with their outward normal opposing each other. The offset condition requires that the two faces lie in parallel planes with their outward normal in the same direction. The relationships between a pair of parts are specified by the user in terms of their features and the mating conditions between them. The individual parts in an assembly are created before the assembly module is invoked. The assembly modeling module requires information about the relationships between the part features. The information specified for each mating condition includes the ID of the mating feature and the type of mating conditions.

To build a list of all the characteristics of an assembly the assembly format is developed to store all the charact-



Figure 1. Flow of information

eristics in an assembly as its signature. The method explores in depth the assembly tree and extracts assembly related information for each part.

a) The method retrieves the constraints used to specify the position of the part.

b) It identifies which entities are used to constrain the part or subassembly.

c) It identifies the parent features and part of each geometrical entity in use.

#### 4. Application Programming Interface (API)

Application programming Interface is an interface that defines the ways by which an application program may request services from libraries and/or operating systems. An API determines the terminology and calling conventions the programmer should employ to use the services. It may include specifications for routines, data structures, object classes and protocols used to communicate between the requesting software and the library.

An API may be Language-dependent, available only in a given programming language, using the syntax and elements of that language to make the API convenient to use in this context. It can also be Language-independent, written in a way that means it can be called from several programming languages. An API itself is largely abstract in that it specifies an interface and controls the behavior of the objects specified in that interface. The software that provides the functionality described by an API is said to be an execution of the API. An API is typically defined in terms of the programming language used to build the application. The API functions used in this paper are SolidWorks functions. The API functions are essential for developing the application software. The names of the mate features, the types, identities and the types of the mate surfaces, the mate clearances and the reference features etc are included in the mate information. There are three main SolidWorks document types namely Part Document, Assembly Document and Drawing Document. Each document type has its own object (PartDoc, DrawingDoc and AssemblyDoc) with its own set of related functions. For example, the AssemblyDoc::AddComponent4 method exists on the AssemblyDoc object because adding components is specific to assembly documents. The SolidWorks API also has functions that are common to all document types. For example, printing, saving, or determining the file name associated with a document would be common opera-

tions. To expose common document-level functions, the SolidWorks API uses the ModelDoc2 object. The ModelDoc2 object provides direct access to the PartDoc, DrawingDoc, and AssemblyDoc objects. As a general rule, the AssemblyDoc object provides access to functions that perform assembly operations; for example, adding new components, adding mate conditions, hiding and exploding components. The SolidWorks API also includes functions that are common to all document types; for example, determining the file name associated with a document is a common operation. To expose common document-level functions, the SolidWorks API uses the ModelDoc2 object. The structure of the assembly document is shown in Figure 2. The AssemblyDoc object is derived from the ModelDoc2 object. Therefore, an AssemblyDoc object has access to all of the functions on the ModelDoc2 object. The following objects related to the mate information like Mate, Mate Feature, Face and Surface. The Mate Object allows access to various assembly mate parameters. The MateEntity object enables access to mated objects and the assembly mate definition. The Feature Object allows access to the feature type, name, parameter data and to the next feature in the FeatureManager design tree. The mate information accessed through these objects in the FeatureManager design tree.

#### 5. Mate Information

Most CAD systems represent the assembly using constr-

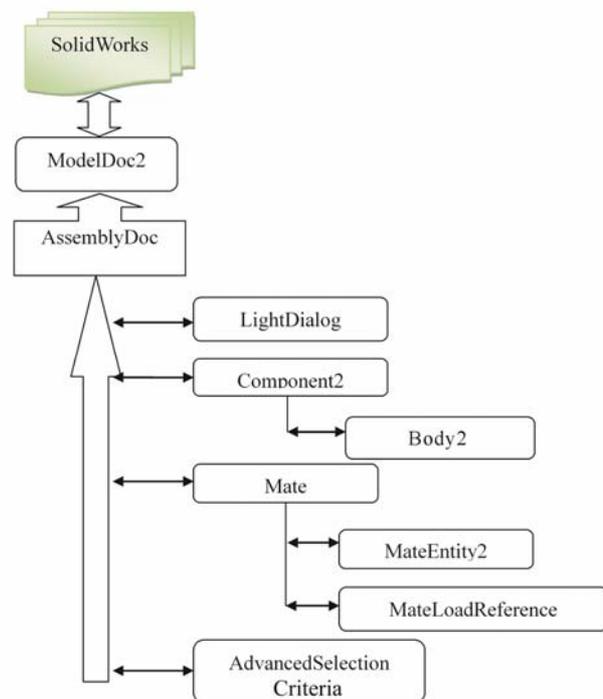


Figure 2. Detailed structure of SolidWorks assembly document

relationships between parts. A mate feature tree is obtained after the parts are assembled together using SolidWorks. With the help of SolidWorks API functions the mate constraints information is extracted by traversing the mate feature tree object. The information generated is represented in an object oriented way to generate assembly strategies especially assembly sequence plans.

A mate surface is a geometric surface of a part model that has mate relationships with other part models. A part model is composed of many surfaces, but only a few surfaces have mate relationships with other parts. Each surface is mated with the other using the mate entities (Table 1) generating a constraint relation. The constraint list includes all the geometrical constraints defined in the CAD model; each constraint in the list includes the constraint type, such as concentric, against and collinear, and mate tolerance, etc and its corresponding code as displayed in the Table 2.

The interface consists of a provision to load Solidworks.exe and create part models and assemble them. The part models are saved as .sldprt file and assembly models are saved as .sldasm file. A provision is provided to assemble the part models directly if they already exist.

### 5.1 Assembly Mate Extraction (AME)

#### Algorithm

An algorithm is developed and executed using Visual Basic for Applications in order to access feature tree of

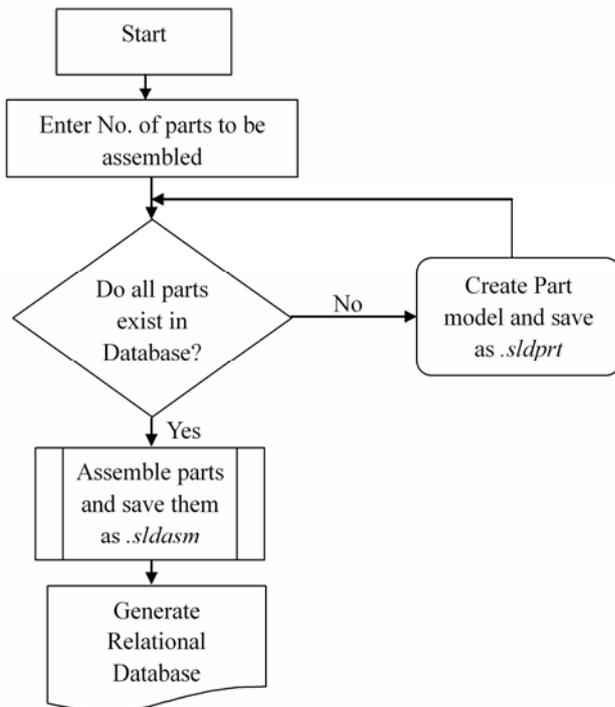


Figure 3. Process flow to generate relational database

the Solidworks assembly module.

The relationships between parts in the assembly are extracted using the Assembly Mate Extraction Algorithm and stored in a database. In addition to the mate type and mate entities types, the algorithm returns the following array of doubles: [ *pointX*, *pointY*, *pointZ*, *vectorI*, *vectorJ*, *vectorK*, *halfangle*, *radius* ]

where

- *pointX* is the X location of this mate entity in the assembly model space
- *pointY* is the Y location of this mate entity in the assembly model space
- *pointZ* is the Z location of this mate entity in the assembly model space
- *vectorI* is the i component of the assembly mate vector
- *vectorJ* is the j component of the assembly mate vector
- *vectorK* is the k component of the assembly mate vector
- *halfangle* is the value for the half angle
- *radius* is the value for the radius

Table 1. List of mate entity types

MATE TYPES	CODE
swMateUnsupported	0
swMatePoint	1
swMateLine	2
swMatePlane	3
swMateCylinder	4
swMateCone	5
swMateSphere	6
swMateCircle	7

Table 2. List of mate types

MATE TYPES	CODE
swMateCOINCIDENT	0
swMateCONCENTRIC	1
swMatePERPENDICULAR	2
swMatePARALLEL	3
swMateTANGENT	4
swMateDISTANCE	5
swMateANGLE	6
swMateUNKNOWN	7
swMateSYMMETRIC	8
swMateCAMFOLLOWER	9
swMateGEAR	10

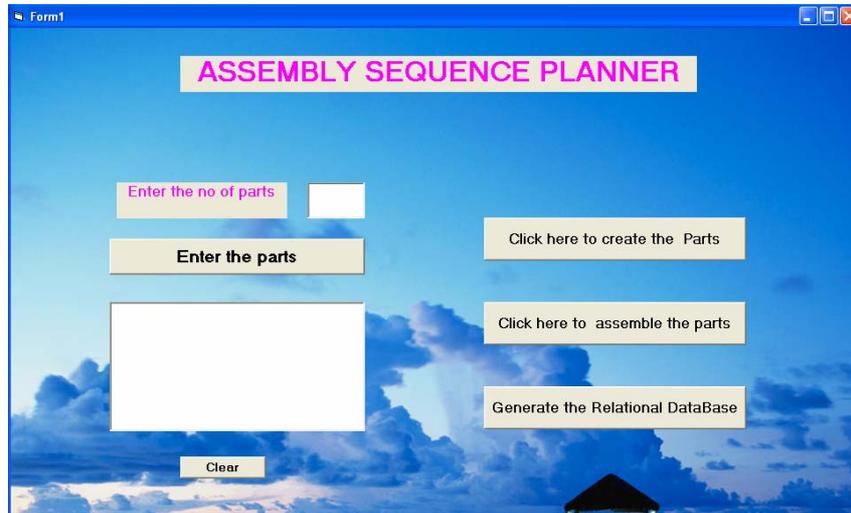


Figure 4. User interface between SolidWorks and Visual Basic

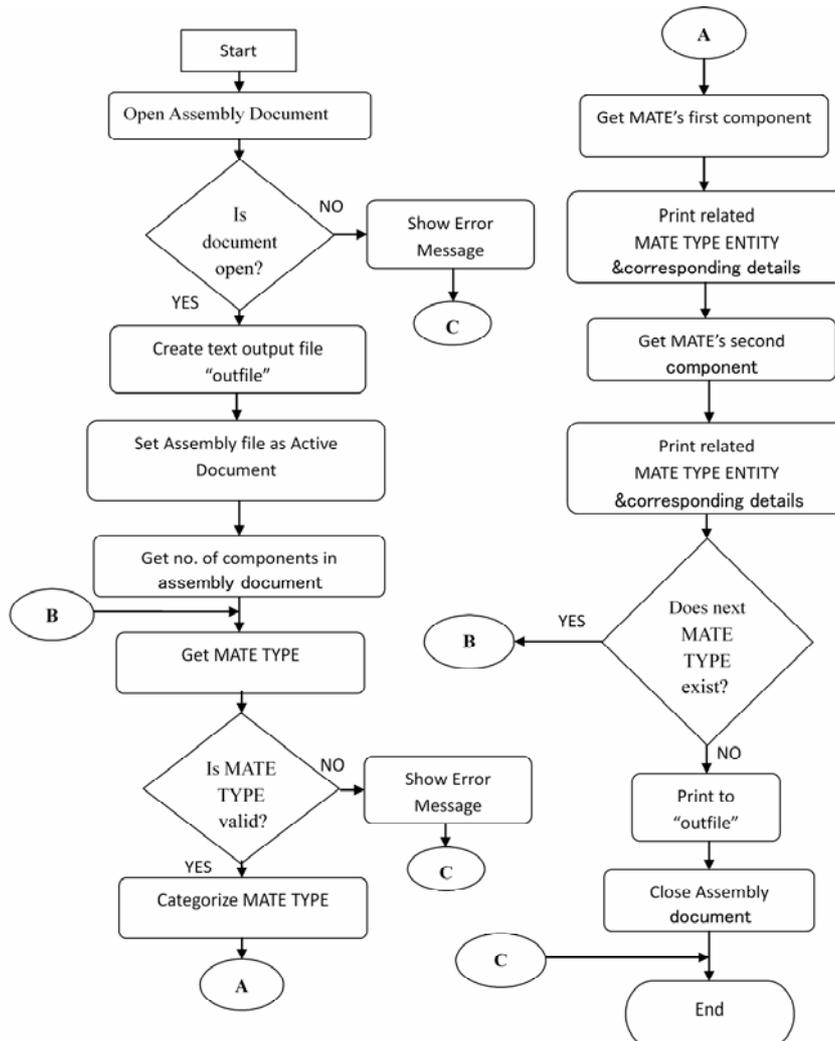


Figure 5. Flowchart for AME algorithm

To define the mate entity, the following information shown in Table 3 is returned based on the mate type. All coordinate information is given in terms of the assembly coordinate system where the mate resides.

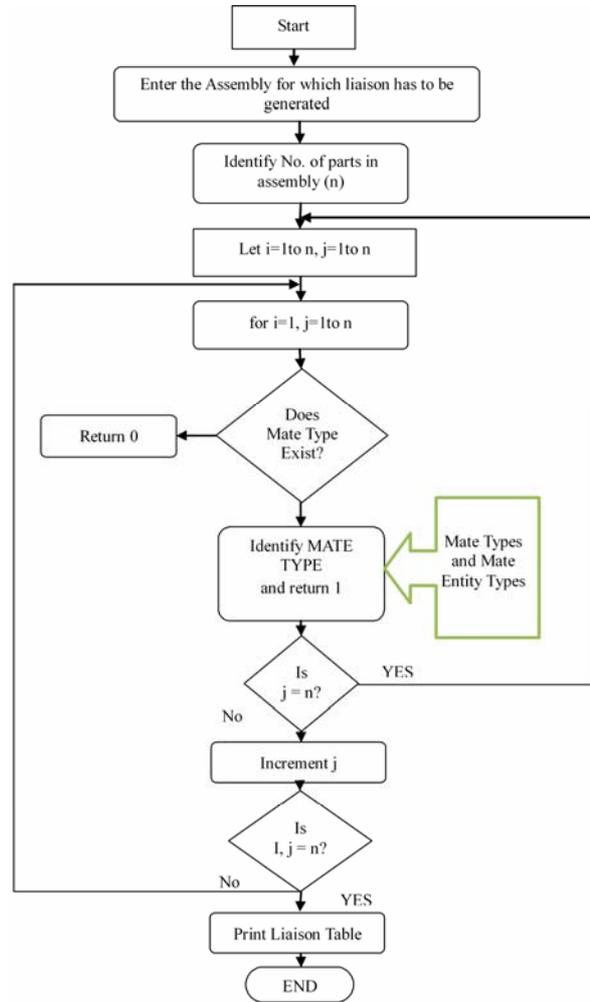
The database contains the relations (or the Mategroups in Solidworks) between various parts of the assembly. There can be more than one relation between parts such as the axis of the two parts is aligned to each other and one face of the first part is coincident to the other. In addition to the basic relations, the database also contains the type of relation or mate whether it is a point contact, line contact or a plane contact as in the case. It can also be inferred whether the component is a cylindrical component or a prismatic component. If the program returns a value for the radius then the component is a cylindrical component else it is a prismatic component. The mass properties such as the weight, volume of individual components taking part in the process of assembly is also considered. The above information is necessary to assist in the selection of the base component for the assembly and to analyze the assembly plan with respect the Design for Assembly.

### 5.2 Symbolic Representation of Liaisons

Liaison graph can portray different logical and physical contact relations among the parts of the assembly. Liaison graph is a two-tuples  $G = (P,L)$ , where  $P$  is a set of nodes that represent parts, and  $L$  a set of edges that represent any of certain user defined relations between parts called liaisons. Given the liaisons graph, a decomposition method is used to systematically generate the assembly plans. The graph representation is difficult to process using a computer, but can easily handle the information in a matrix form. The table of liaisons or the liaison matrix is used to represent the contact information between

**Table 3. Information is returned based on the mate type**

Mate Type	Information Returned
swMatePoint	<i>pointX, pointY, pointZ</i>
swMateLine	<i>pointX, pointY, pointZ, vectorI, vectorJ, vectorK</i> where the point is a point on the line and the vector represents the line direction.
swMatePlane	<i>pointX, pointY, pointZ, vectorI, vectorJ, vectorK</i> where the point is a point on the plane and the vector represents the plane normal.
swMateCylinder	<i>pointX, pointY, pointZ, vectorI, vectorJ, vectorK, halfangle</i> where the point is a point on the cylinder axis and the vector represents the cylinder axis.
swMateCone	<i>pointX, pointY, pointZ, vectorI, vectorJ, vectorK, halfangle, radius</i> where the point is a point on the cone axis and the vector represents the cone axis.



**Figure 6. Flowchart for LTG algorithm**

the components. If one or more relation is found to exist between 2 pairs of components of the assembly, then that pair will have a value one else zero.

*i.e.*, if  $L_{i,j} = 1$ , then relation exist between component ‘i’ and ‘j’

if  $L_{i,j} = 0$ , then no relation exist between component ‘i’ and ‘j’

The Matrix Cell  $ij$  contains the value of  $L$  as 0 or 1.

It assumed that between two parts there is only one edge representing a liaison that includes all contacts such as collinear, coincident etc. There is no liaison joining a vertex to itself because it is impossible to assemble a component to itself. A component belongs to the product it has at least a liaison with another component of the assembly. If a component  $p_i$  can be assembled to  $p_j$ , the reverse is also true. An algorithm named Liaison Table Generator (LTG) Algorithm is developed to generate the liaison matrix based on the relationships existing among parts and is described in Figure 6.

## 6. Implementation with an Example

The clamping fixture (Figure 7) taken for the present study has 7 components namely the base, plunger, link, knob, lever, clamp-end and pin. SolidWorks software was used to model the assembly. The individual components were created as separate geometric models in the part models in the part mode and saved as “.sldprt” files. Next, the assembly modeling mode is invoked and the Base is taken as the support component. After specifying the assembly constraints, the assembly was built by adding the remaining components to the casting. All the components are assembled using the mate attributes like the Coincident, Parallel, Perpendicular, Tangent, Concentric, Distance and Angle. The completed assembly model is then saved as a “.sldasm” file.

The feature tree of the clamping fixture modeled and assembled is generated in the assembly module after adding all the mating conditions is shown in Figure 8. This feature tree appears on the left side of the Assembly module. The mate relationship information stored in this tree is extracted using the AME algorithm and stored in a notepad file as shown in Figure 9. The AME algorithm is implemented in a computer program coded in Visual Basic for Applications (VBA) in the Microsoft Windows platform. VBA is chosen, essentially for availability and portability reasons in addition to the ability to interface with SolidWorks. The machine used to run the program was a notebook with a Pentium Centrino Duo 1.6 GHz processor, with 1 GB DDR RAM.

Once the notepad file is generated, it acts as a database, from which the liaisons between the various components of the assembly can be generated. The LTG algorithm is applied to the clamping fixture assembly and the liaison table with relations is generated. The LTG algorithm is implemented JAVA and resultant liaison table is shown in Figure 10. The output of the program gives the number of parts in the assembly, mating relationships that exist between various parts of the assembly and finally displays the liaison diagram or table showing the relations.

## 7. Conclusions

The liaison information contains relationships between parts is the basic information needed for assembly sequence planning. Although many modeling packages provide information about the solid models, the information regarding to the relationships between parts in the assembly is not explicitly available. In this paper, a novel method of determining the relationship liaison diagram is proposed. The method first extracts the assembly information, processes it and then generates liaisons. The mating feature matrix of two contacting parts in the assembly is established. The technique is fully automated, simplifies the process of extracting geometrical constraints for any given assembly considering the relationships between components of a CAD model using the

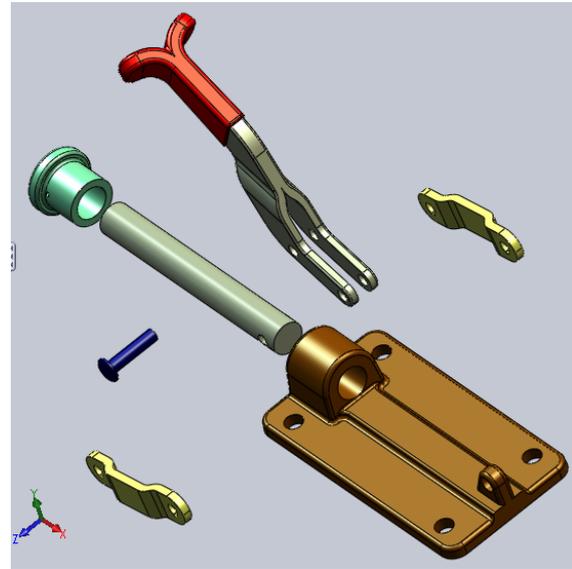


Figure 7. Example of clamping fixture (exploded view)

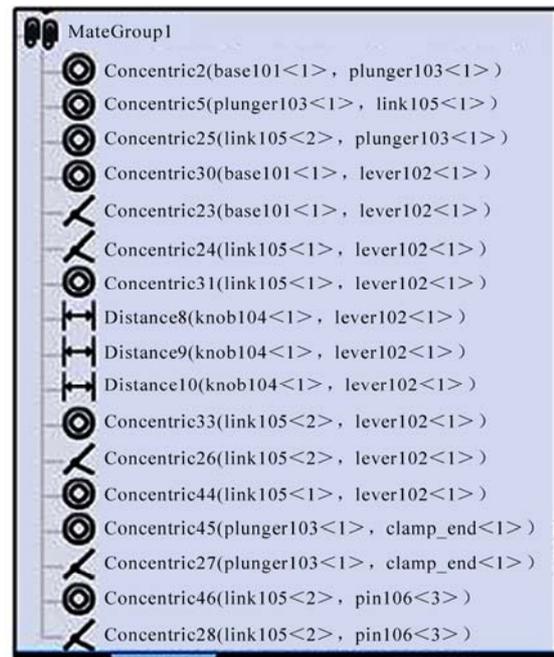


Figure 8. Feature tree of the assembly created in SolidWorks

Automated Programmable Interface of the CAD software. The main constraint in this approach is that the designer should model the assembly components and define the mating conditions while assembling the product. Once the database containing the relationships is extracted, it can be used to generate the liaison diagram and liaison tree. Taking the precedence relations into consideration, feasible assembly sequences of a given assembly can be generated.

```
" File=D:\SolidWorks Working Directory\clamping fixture \clamping_fixture.sldasm "
MateGroup1
"
" Concentric2 "
Type =1 "
AlignFlag =1 "
CanBeFLipped=False "
"
Component =base101-1 "
MateEntType =4 "
(x,y,z) =(-6.09316913443831E-02, 2.60376887513812E-02, -1.62423420159542E-16) "
(i,j,k) =(1,0,0) "
Radius 1 =0.00635 "
Radius 2 =0 "
"
" Concentric2 "
Type =1 "
AlignFlag =1 "
CanBeFLipped=False "
"
Component =plunger103-1 "
MateEntType =4 "
(x,y,z) =(-2.06503960267056E-02, 2.060376887513812E-02, -1.62423420159542E-16) "
(i,j,k) =(-1,0,0) "
Radius 1 =0.00635 "
Radius 2 =0 "
"
-----
" Concentrics "
"
Concentrics "
Type =1 "
AlignFlag =1 "
CanBeFLipped=False "
"
Component =plunger103-1 "
MateEntType =4 "
(x,y,z) =(-2.70003960267056E-02, 2.60376887513812E-02, -6.350000000000E-03) "
(i,j,k) =(0, 0,1) "
Radius 1 =2.38125000000001E-03 "
Radius 2 =0 "
"
Concentrics "
Type =1 "
AlignFlag =1 "
CanBeFLipped=False "
"
Component =link105-1 "
MateEntType =4 "
(x,y,z) =(-2.70003960267056E-02, 2.60376887513812E-02, -6.350000000000E-03) "
(i,j,k) =(1.15754264840233E-16, -5.18983244566464E-16, -1) "
Radius 1 =0.00238125 "
```

Figure 9. Snapshot of the relational database generated

```
C:\WINDOWS\system32\cmd.exe
C:\ScrewJack>java -jar screwJack.jar "C:\ScrewJack\clamping_fixture.txt"
"pin106-3" "plunger103-1" "link105-1" "knob104-1" "lever102-1" "link105-2" "clam
p_end-1" "base101-1"
Concentric2 = < base101-1, plunger103-1>
Concentric5 = < plunger103-1, link105-1>
Concentric25 = < link105-2, plunger103-1>
Concentric30 = < base101-1, lever102-1>
Coincident23 = < base101-1, lever102-1>
Coincident24 = < link105-1, lever102-1>
Concentric31 = < link105-1, lever102-1>
Distance8 = < knob104-1, lever102-1>
Distance9 = < knob104-1, lever102-1>
Distance10 = < knob104-1, lever102-1>
Concentric33 = < link105-2, lever102-1>
Coincident26 = < link105-2, lever102-1>
Concentric44 = < link105-1, lever102-1>
Concentric45 = < plunger103-1, clamp_end-1>
Coincident27 = < plunger103-1, clamp_end-1>
Concentric46 = < link105-2, pin106-3>
Coincident28 = < link105-2, pin106-3>

=====
0 0 0 0 0 1 0 0
0 0 1 0 0 1 1 1
0 1 0 0 1 0 0 0
0 0 0 0 1 0 0 0
0 0 1 1 0 1 0 1
1 1 0 0 1 0 0 0
0 1 0 0 0 0 0 0
0 1 0 0 1 0 0 0
=====
```

Figure 10. Liaison table generated

## REFERENCES

- [1] J. L. Nevins and D. E. Whitney, "Concurrent design of product and processes," McGraw-Hill, New York, 1989.
- [2] U. Rembold, C. Blume, and R. Dillmann, "Computer-integrated manufacturing technology and systems," Marcel Dekker, New York, 1985.
- [3] S. S. F. Smith, "Using multiple genetic operators to reduce premature convergence in genetic assembly planning," *Computers in Industry*, Vol. 54, Iss. 1, pp. 35–49, May 2004.
- [4] C. M. Eastman, "The design of assembly," SAE Technical Paper Series 0148-7191/81/0223-0197, 1981.
- [5] H. Ko and K. Lee, "Automatic assembly procedure generation from mating conditions," *Computer Aided Design*, Vol. 19, pp. 3–10, 1987.
- [6] L. S. Homen De Mello, and A. C. Sanderson, "Representations of mechanical assembly sequences," *IEEE Transactions on Robotics and Automation*, Vol. 7, No. 2, pp. 211–227, 1991.
- [7] D. F. Baldwin, T. E. Abell, M. C. M. Lui, T. L. D. Fazio, and D. E. Whitney, "An integrated computer aid for generating and evaluating assembly sequences for mechanical products," *IEEE International Conference on Robotics and Automation*, Vol. 7, No. 1, pp. 78–94, 1991.
- [8] K. W. Lyons, V. N. Rajan, and R. Sreerangam, "Representations and methodologies for Assembly Modeling," NIST Int. Rep., Gaithersburg, MD, 1996.
- [9] J. J. Shah and M. T. Rogers, "Assembly modeling as an extension of feature based design," *Recent Trends in Engineering Design*, Vol. 3, No. 3 & 4, pp. 218–237, 1993.
- [10] C. Mascle, "Features modeling in assembly sequence and resource planning," In *Proceedings IEEE International Symposium on Assembly and Task Planning*, Pittsburgh, PA, pp. 232–237, 1995.
- [11] R. Anantha, G. A. Kramer, and R.H. Crawford, "Assembly modeling by geometric constraint satisfaction," *Computer Aided Design*, Vol. 28, No. 9, pp. 707–722, 1996.
- [12] R. B. Gottipolu and K. Ghosh, "An Integrated approach to the generation of assembly sequences," *International Journal of Computer Applications in Technology*, Vol. 8, No. 3–4, pp. 125–138, 1995.
- [13] R. B. Gottipolu and K. Ghosh, "A simplified and efficient representation for evaluation and selection of assembly sequences," *Computers in Industry*, Vol. 50, pp. 251–264, 2003.
- [14] L. Laperriere and H. A. ElMaraghy, "Assembly sequences planning for simultaneous engineering applications," *International Journal of Advanced Manufacturing Technology*, Vol. 9, pp. 231–244, 1994.
- [15] A. C. Lin and T. C. Chang, "3D MAPS: Three dimensional mechanical assembly planning system," *Journal of Manufacturing Systems*, Vol. 12, No. 6, pp. 437–456, 1993.
- [16] R. J. Linn and H. Liu, "An automatic assembly liaison extraction method and assembly liaison model," *IIE Transactions*, Vol. 31, pp. 353–363, 1996.
- [17] R. S. Chen, K. Y. Lu, and P. H. Tai, "Optimizing assembly planning through a three-stage integrated approach," *International Journal of Production Economics*, Vol. 88, pp. 243–256, 2004.
- [18] J. S. Kim, K. S. Kim, J. Y. Lee, and J. H. Jeong, "Generation of assembly models from kinematic constraints," *International Journal of Advanced Manufacturing Technology*, Vol. 26, pp. 131–137, 2005.
- [19] R. Sudarsan, Y. H. Han, S. Fofou, S. C. Feng, U. Roy, F. Wang, R. D. Sriram, and K. Lyons, "A model for capturing product assembly information," *Journal of Computing and Information Science in Engineering*, Vol. 6, No. 1, pp. 11–21, 2006.
- [20] T. Dong, R. Tong, Ling, and J. Dong, "A Knowledge based approach to assembly sequence planning," *International Journal of Advanced Manufacturing Technology*, Vol. 32, pp. 1232–1244, 2007.
- [21] H. Wang, D. Xiang, G. Duan, and L. Zhang, "Assembly planning based on semantic modelling approach," *Computers in Industry*, Vol. 58, pp. 227–239, 2007.