ON THE CREATION OF A UNIFIED MODELING LANGUAGE BASED COLLABORATIVE VIRTUAL ASSEMBLY/DISASSEMBLY SYSTEM

Jianzhong Mo, eMedia Center, College of Engineering, University of Wisconsin – Madison, WI 53713

Chi-Cheng Chu, Department of Mechanical Engineering, University of Wisconsin – Madison, WI 53713

B. S. Prabhu, Henry Samueli School of Engineering, UCLA, Mechanical and Aerospace Engineering, Los Angeles, CA 90095

Rajit Gadh, Henry Samueli School of Engineering, UCLA, Mechanical and Aerospace Engineering, Director, Internet-Collaborative Design Lab, 420 Westwood Plaza, Los Angeles, CA 90095

ABSTRACT

Design For Assembly/disassembly (DFA) allows systematic evaluations and analysis of product assembly and disassembly during the product design stage with the goal of reducing assembly costs, improving quality and shortening time to market. These evaluations include assemblability, disassemblability, part accessibility, and part layout [1]. With the growing availability and widespread use of high bandwidth network and Virtual Reality (VR) technologies, the issues involved with product assembly, disassembly may be addressed and resolved effectively in a collaborative virtual environment over the Internet. In this paper, an analysis on the framework and definition of a collaborative virtual assembly and disassembly system, called Motive3D [2, 3], is introduced and examined using Unified Modeling Language (UML) [4]. UML is an Object Management Group’s (OMG) standard for object-oriented analysis and software design method, which allows specifying, constructing, visualizing, and documenting a distributed software-intensive system. Carrying out systematic analysis on the Motive3D system using UML, we believe that it expedites system development and will be possible to expedite the product design cycle and enable the assembly and disassembly process over Internet, and thus reduce the assembly and disassembly process bottlenecks.

Key words: Design For Assembly (DFA), Collaborative Virtual Assembly/Disassembly, Unified Modeling Language (UML).

INTRODUCTION

Economic and technological forces are driving a worldwide transition from vertical to virtual corporations. Technologies such as Virtual Prototyping, Virtual Manufacturing, Virtual Design, Virtual Assembly and Virtual Maintenance have revealed their immense potential of becoming new paradigms for product realization. The Internet, Intranet and the Web-based technologies have also facilitated a collaborative environment for the different stages in the design cycle. Collaboration has enabled the sharing of knowledge and interaction between participants. Successful collaborative initiatives in the area of product design have been reported (Roucoule et al [5]). In a product cycle there are a number of other activities: manufacturing, inspection, assembly, disassembly, recycle, etc., and efforts are directed at incorporating these activities within the gamut of product realization.

Virtual Manufacturing is a new paradigm for collaborative manufacturing – design, manufacture, assembly, and disassembly on the Internet. Internet-based Collaborative Virtual Assembly/disassembly is a subset of Virtual Manufacturing. Virtual Assembly/disassembly attempts to mitigate the difficulties and other issues related to the methods of assembly and disassembling a product in the design stage rather than after manufacture (machining, forming, casting, etc.) in the case of assembly; and at the end of its life cycle in the case of disassembly. Based on the results of Virtual Assembly/disassembly analysis, design and handling faults can be eliminated, and designs can be improved for better assembling/disassembling. Internet-based Collaborative Virtual Assembly/disassembly exploits the Internet as a medium to post assembly and disassembly analysis results and to perform assembling and disassembling processes collaboratively in a virtual environment. Virtual Assembly/disassembly not only allows engineers to share the knowledge of assemblability and disassemblability of particular products, but also encourages
customers or suppliers to get involved in the manufacturing activities in a substantive way during the design phase of the product.

Virtual Assembly/disassembly can be adopted in the following areas to support Virtual Manufacturing:

1) Interactive Visualization: To provide greater immersive experience and flythrough capabilities for the visualization of a product or its designs, to animate/simulate product functionality, and to digitize complex assembly/disassembly sequences for components of a product.

2) Collaborative Assembly/disassembly: To provide geographically distant users information regarding the feasibility of assembling the components, the accessibility of components and the sequence and paths to assemble the components in an assembly.

3) Virtual Maintenance: To allow the determination of the sequence and cost in disassembling/assembling components for product maintenance collaboratively via Internet. In turn, the designers may make design changes to facilitate ease-of-assembly/disassembly for maintenance.

4) Virtual Recycling: To allow the determination of the maximal profitable disassembly sequence for separating components of different materials. Maximizing the recycling profit results in greater incentives for companies to recycle a product. In addition, to allow companies to determine the cost involved when the product is disassembled for recycling.

In this paper, a Collaborative Virtual Assembly/disassembly System, called Motive3D, is developed using Unified Modeling Language (UML). UML diagrams such as Use Case Diagram, Class Diagram, Sequence Diagram, Activity Diagram and Physical Diagram are used to identify and analyze the definitions and functionality within the Motive3D system. Details of these diagram analysis results are presented in the following sections.

RELATED RESEARCH

The research to date on Assembly/disassembly planning has focused mainly on assembly/disassembly sequences and path generation. A disassembly sequence is defined as the order of separating a component from product. For complete disassembly sequence generation, Homem de Mello et al [6] introduced AND/OR graph which organizes conditions and precedence relationships between components. Baldwin [7] introduced an assembly sequence diagram that presents the capability or incapability to disassemble a component. Yokota and Brough [8] introduced the method of precedence relations, which describes disassembly precedence among components. Woo and Dutta [9], Mattikalli and Khosla [10] and Wolter et al [11] introduced geometry constraints and contact geometry, which quantify the ease of disassembly of components for disassembly sequences. Wilson et al [12] introduced a non-directional blocking graph, which describes interfering parts from the blocking nature of components. Previous research at I-CARVE Lab of University of Wisconsin - Madison (Lee and Gadh [13], Shyamsunder et al [14], Srinivasan and Gadh [15, 16, 17]) has established sound foundations of assembly/disassembly sequencing, destructive disassembly, selective disassembly, disassembly evaluation, and assembly modeling. But not much effort has been put on the research of Collaborative Virtual Assembly/disassembly.

The Unified Modeling Language (UML) is an Object Management Group’s (OMG) standard object-oriented analysis and design method. UML unifies the perspectives among many different kinds of systems, development phases (requirements analysis, design and implementation), and internal concepts. It is a notation, mainly diagrammatic, to express system design and analysis [18]. The development of UML began in late 1994 when Grady Booch and Jim Rumbaugh of Rational Software Corporation began their work on unifying Object Modeling Technique (OMT) [19, 20] methods. In 1995, this unification effort merged with Object-Oriented Software Engineering (OOSE) method when Ivar Jacobson [21] and his Objectory company joined Rational Software Corporation. OMG is currently developing UML2.0. In this paper, UML diagrams such as Use Case Diagram, Class Diagram, Sequence Diagram, Activity Diagram and Physical Diagram are used to design and analyze Collaborative Virtual Assembly/disassembly system. The software implementation, called Motive3D, is developed based upon this Object-Oriented approach.

COLLABORATIVE VIRTUAL ASSEMBLY / DISASSEMBLY

Virtual Assembly/disassembly deals with following aspect of a product design: 1) Functional Analyze: to analyze product and component’s functional relations to avoid functional redundancy and deficiency, thus, number of parts and product structure is reduced and simplified. 2) Connection Analyze: to analyze the ease of connecting processes, e.g., simplify connecting mechanisms, avoid fasteners as much as possible, and eliminate connection adjustments. 3) Operation Analyze: to analyze the ease of handling, orientation, positioning, design optimization through path and sequence analysis, and assembly/disassembly instructions.

For the convenience of addressing the above three analysis activities in a Collaborative Virtual Assembly/disassembly System, Assembly Relation Model (ARM) [22] is defined. In the following paragraph, the definition of ARM is briefly described.

ARM is defined as follow:

\[ ARM = <C, F, G, J, R> \]

Where ARM is Assembly Relation Model, a set of following five elements:

- \( C \) is a finite set of parts/components:
  \[ C = \{C_1, C_2, ..., C_m\} \]
- \( m \) is the total number of components of an assembly.
- \( F \) is a finite set of functions:
  \[ F = \{f(C_1), f(C_2), ..., f(C_m)\} \]
- \( f(C_i) \) represents the total functions of component \( C_i \).
- Component functions are divided into two categories: 1) essential functions: the functions necessary to accomplish product’s performance, and 2) assistant functions: the functions to achieve essential functions.
  \( G \) is a finite set of component’s geometric model in Boundary Representation (B-rep) [23] format:
  \[ G = \{g(C_1), g(C_2), ..., g(C_m)\} \]
  \( g(C_i) \) represents the geometric model of component \( C_i \) in B-rep format, a component can be represented by \( BODY, LUMP, SHELL, FACE, LOOP, COEDGE, EDGE, \) and
**CURVE (L)**

is a finite set of connecting mechanisms:

\[ J = \{ J_1, J_2, \ldots, J_n \} \]

\( J_i \) represents one connecting mechanism between components. \( n \) is the total number of connecting mechanisms. Examples of typical connecting mechanisms are thread connecting, pin connecting, rivet connecting, key connecting, interference fit, and welding.

**RELATION (R)**

is a finite set of relations to represent all relations among components’ geometric definitions, such as planar relations, collinear relation, perpendicular relation, and parallel relation. \( R \) is a set of connection relations, which represents the relations among connecting mechanisms and components.

Collaborative Virtual assembly/disassembly exploits the Internet as a medium to post Assembly/disassembly analysis results and to perform assembling/disassembling processes collaboratively in a virtual environment. There are three requirements for Collaborative Virtual Assembly/disassembly systems: 1) To automatically/collaboratively generate an assembly/disassembly sequence and an efficient path by deploying assembly/disassembly instructions/knowledge or heuristics accessible by Internet. 2) To provide capability to interactively visualize assembly/disassembly sequences and paths accessible over the Internet. 3) To evaluate and post assembly/disassembly cost/benefits based on the different sequences and paths accessible over the Internet. Based on these requirements, UML diagrams are introduced to examine the design of the software implementation of current research, Motive3D. UML allows specifying, constructing, visualizing, and documenting a distributed software-intensive system. Carrying out systematic analysis on the Motive3D system using UML will expedite the product design cycle and enable the assembly and disassembly process over Internet. In the following sections, it is assumed that ARM has been established and serves as the information source for all assembly/disassembly analysis. To simplify while maintaining the integrity of case study, only Operation Analyzing of Motive3D system will be discussed in this paper.

**Requirements Analysis in Use Case Diagram**

A use case is a set of scenarios tied together by a common use goal. A use case is defined as a kind of classifier which represents a coherent unit of functionality provided by a system, a subsystem, or a class. Use case is manifested by the sequences of messages exchanged among the system (subsystem, class) and one or more outside interactors (called actors) together with actions performed by the system (subsystem, class). Use case diagram is a diagram for visualizing actors and use cases together with their relationships. An actor defines a coherent set of roles that users of an entity can play when interacting with the entity. Use case and use case diagram are used to model system requirements. In addition to the links among actors and use cases, several kinds of relationships can be introduced between use cases. The relationships are associations between the actors and the use cases, generalizations between the actors. Typical relationships among the use cases are generalizations, extends, and includes. The use case diagram for Operation Analyzing of Motive3D system will be discussed in the following paragraphs.

To describe the system behavior from user’s standpoint, we generated the use case diagram for Operation Analyzing as shown in Fig 1. There are seven general use cases (denoted as ellipses): (1) Assembly Instruction, (2) Path Generation, (3) Sequence Generation, (4) Check Interference, (5) Check Interference For Polygon Geometry, (6) Check Interference For Native Geometry, and (7) Cost Evaluation. Use case “Assembly Instruction” depends on use cases “Path Generation” and “Sequence Generation”, both of which again include use case “Check Interference”. Include relationship occurs when a serial of behavior is similar across more than one use case. Use cases “Check Interference For Polygon Geometry” and “Check Interference For Native Geometry” are detailed use cases for “Check Interference”. These use case have generalization relationships: “Check Interference For Polygon Geometry” and “Check Interference For Native Geometry”. In use case diagram, an actor is a role that a user plays with respect to the system. In Fig 1., there are two actors (denoted as stickman symbols): User and Native Geometry Kernel. Actor User performs four use cases while actor Native Geometry Kernel performs only one use case. Actor Native Geometry Kernel represents a software module that checks interference and returns the operation results via assembly and part geometric relations of ARM.

**Class Diagram**

In UML, a class diagram is a collection of static declarative model elements, such as classes, interfaces, and their relationships, connected as a graph to each other and to their dependencies.
A class is the descriptor for a set of objects with similar structure, behavior, or relationships. A summary of UML class diagram notation is shown in Fig. 2. As shown in Fig. 2, relationships include sub-class relation, associate relation, and part-of relation (composition relation and aggregation relation). In a UML class diagram, a stereotype may be used to represent a usage distinction. The name of the stereotype is quoted within matched guillemets.

![Fig. 2 Summery of UML Class Diagram Notation](image)

Class diagrams are used to describe the types of objects in the system and the static relationship among them. From the specification view of Motive3D Virtual Assembly/disassembly system, the class diagram is shown in Fig. 3. The classes (notated as rectangles) are grouped into five categories or packages: interface classes, controller classes, entity classes, service classes, and modeling kernel.

Interface classes are user-interface objects. Those classes are designed to interact with users such as menu activation and mouse movement. Class Canvas3D is the interface class to render an assembly or product in 3D. Controller classes are non-user objects and are responsible for handling system events. A controller class defines the method for the system operation. Among controller classes of Virtual Assembly/disassembly system, class Universe is designed to handle system events about operations on geometry model, such as construction and modification of Scene, Path, and Sequence. Class Motive3D handle visualization events (pan, rotate and zoom geometry models) delegated by class Universe. Class Sensor handles virtual reality device (such as 3D mouse, digital glove and motion trackers) events delegated by class Universe. Sensors are attached to Entity Node.

Service classes are utility objects. GeometryHandlers provide method for faceting, interference checking, path-sweep volume generation and part clearance calculation. For assembly parts modeled as native geometry such as the geometry formats defined in ACIS or Parasolid, GeometryHandlers call APIs of ACIS or Parasolid kernel.

Entity classes are objects which represent domain concepts. When system exits, entity classes will be saved into hard disk. Among entity classes, Class Scene is used to describe structure of an assembly and relations among assembly parts. A Scene is aggregated with a number of Nodes. Group Node, Geometry Node and Transform Node are subclasses of Node. Geometry Node represents assembly part and Group Node while Transform Node represents relations among assembly parts. Path and Sequence are two important objects in Virtual Assembly/disassembly system and are defined as follow:

Path State (PS) is set of assembly/disassembly states:

\[ PS = s_1, s_2, \ldots, s_n \]  

where \( s_i \) is the assembly/disassembly state of component \( C_i \) in \( A \) and \( n \) is the number of states, \( A \) is assembly. Disassembly state \( s = [p, q] \), \( p \) is the position element of \( s \), and \( q \) is the
orientation element of \( s \) represented in quaternion or matrix format.

*Path (P)* is a transition from one state to another:
\[
P = T(s_i, s_j),
\]
where \( s_i, s_j \in PS, \) \( T \) is the interpolation method (linear or cubic interpolation).

An example of path generated by Motive3D is shown in Fig. 4. In this example, there are 5 path states and 4 paths to assemble the part Oil_pan onto a motor. Those 5 paths include 1 rotation and 3 translations.

*Sequence (S)* is a list of paths: \( S = \{P_1, P_2, \ldots, P_i, \ldots, P_j, \ldots, P_n\} \), where \( P_i \) is the disassembly path of \( C_i \in A \), \( P_j \) is the disassembly path of \( C_j \in A \), \( i \neq j, i, j < n \), \( n \) is the number of paths, and \( P_i \) and \( P_{i+1} \) have the relation of \( P_i \propto P_{i+1} \) or \( P_i \sim P_{i+1} \), where \( \propto \) represents the relation of precedence, and \( \sim \) represents the relation of indifference. If \( n < m \), \( m \) is the component number of \( A \), then \( S \) is called *selective sequence*, otherwise \( S \) is called *complete sequence*. If \( S \) has at least one relation of indifference (\( \sim \)), then \( S \) is called *parallel sequence*; otherwise \( S \) is called *sequential sequence*. Fig. 5 shows the disassembly sequence to disassemble part \( s21 \) of augmenter subassembly: \( ol11 \rightarrow ol1 \rightarrow s1 \rightarrow ol33 \rightarrow ol3 \rightarrow ol2 \rightarrow s21 \). It is a sequential sequence. The paths to disassembly above parts are defined as disassembly directions.

**Sequence Diagram for Check Interference**

**Use Case**

In UML, interaction diagrams are models that describe how groups of objects collaborate. Usually interaction diagram captures the behavior of a single use case. There are two kinds of interaction diagrams: sequence diagram and collaboration diagram.

Sequence diagram emphasizes the order in which things occur. Collaboration diagram emphasizes on the layout of statically connected objects. Within a sequence diagram, objects are shown as boxes at the top of dashed vertical lines called objects’ lifeline which represents the object’s life during interaction. Each message is represented by an arrow between the lifelines of two objects. The order each message occurs is shown top to bottom. Activation box is used to show when an object is active. In this paper, we will use sequence diagram to describe use case Collision Checking.

A Native Geometry Server (a server process native geometries formats defined in ACIS or Parasolid) is needed to process native geometries, such as faceting, Boolean operations. Native geometries include product geometric models in the format of Parasolid or ACIS. In Virtual Assembly/disassembly system, the Native Geometry Server may be installed on one of the networked server. The sequence diagram of for use case of native geometry collision detection is shown in Fig 6. *Geometry Processing Protocol* is defined to handle the communication between networked servers. Interference checking is one example of *Geometry Processing Protocol* communication. In check interference use case, a user triggers *interferenceChecking* event at client. The event is delegated to object of *NativeGeometryHandlerProxy*. *NativeGeometryHandler* is an instance running at the Native Geometry Server. *NativeGeometryHandlerProxy* works with *NativeGeometryHandler* to complete the task. Thus, the process of interference checking is transparent to the user.
Activity Diagram for Motive3D@Visualize

In UML, the activity diagram describes the sequencing of activities which support for both conditional and parallel behavior. The activity diagram represents a state machine of a behavior itself in which the states represent the performance of actions or sub-activities and the transitions are triggered by the completion of the actions or sub-activities. The core symbol (rounded rectangle) in an activity diagram is the activity which is a state of doing something: either a real world process, or the execution of routine. Conditional behaviors are delineated by branches and merges. A branch has a single incoming transition and several guarded outgoing transition. A merge has multiple input transition and single output. Parallel behavior is indicated by forks and joins. A fork has one incoming transition and several outgoing transition. A join is a counterpart of a fork. Parallel behavior means that the sequence between transitions is irrelevant.

In Motive 3D, UML activity diagram is introduced to describe the activities of Motive3D@Visualize (the visualization module of Motive3D), as shown in Fig 7. There are eight activities for Motive3D@Visualize. Initialize Universe activity transits from Build Interface activity. Subsequently, there is a fork transition, transiting to parallel activities of Listen Events, Download Images, and Download 3D Objects. The parallel behavior helps users to view the downloaded models before the assembly is fully downloaded from the server. After activities Download Images and Download 3-D Objects join into Update Universe, user can perform other interactions, such as generating assembly path, sequence, or check assembly instructions. The activity Show Status is common to all other activity, that is, other activities will transit to this activity any time.

Motive3D@Visualize is installed on client machine, which will be described in section 4 in more detail. Fig 8 shows the user interface of Motive3D@Visualize.
**3D ASSEMBLY MODEL TRANSFER AND VIEWING OVER NETWORK**

3D assembly models files are usually quite large and they will cause traffic congestion when transferred through Internet. It’s essential for Virtual Assembly/disassembly system to allow the incremental transfer 3-D models from authorizing server/web server and simultaneous viewing of 3-D models at visualizing web client. A tool called Motive3D@Synthesize that codes assembly models progressively into bit streams and allows the user to view such progressive files as they are downloaded from the web.

With the introduction of progressive meshes [24], more and more Internet applications adopt this approach. In the current system, the mesh simplification algorithm follows the edge collapse operation (as shown in Fig. 9) and quadric metric that was firstly proposed by Garland et al [25]. As shown in Fig. 9, the edge collapse operation reduces triangle numbers by remove edges. This example displays an edge collapse operation by eliminating the darker edge in the left to produce a simplified geometry in the right. Considering flexibility and performance of Motive 3D system, no other compression algorithms are applied.

![Fig 9](image)

**Fig 9.** The edge collapse simplification operation and its inverse vertex split

The following pseudo-code is an example of multi-level representation for a single object. Normal, materials and other appearance properties are not included in this example for simplicity:

```java
class MultilevelObject {
    int vertexNum, meshNum; //vertexNum and meshNum for base mesh
    int historyLen; //the edge collapse number from original mesh to the base mesh
    float coordinates[3]; //coordinates data for base mesh
    float coordinateIndices[3]; //triangle mesh data
    HistoryLog historyLogs[]; //records of edge collapse operation
}
```

Assuming vertices v1 and v2 are involved in edge collapse, v2 will be merged into v1.

After this operation. Thus, all the reshaped triangle meshes have a third vertex besides v1 and v2.

```java
class HistoryLog{
    int v1ID; //record the ordinal no of vertex
    v1 in edge collapse operation
    float dv2[3]; //record the coordinate difference between new and old vertex
    int[] removedMeshVID; //record the ordinal no of third vertex of the removed meshes
    int[] reshapedMeshID; //record the ordinal no of the reshaped meshes
}
```

Fig. 10 shows an example of multi-level rendering for a wheel object. The original object are shown in Fig. 10 (a). The original wire frame model, as shown in Fig. 10 (b), has 677 vertices and 632 triangle meshes. The simplified lever of rendering, as shown in Fig. 10(c) and (d), has 313 vertices and 260 triangle meshes.

![Fig 10](image)

**Fig 10.** An Example of Multi-level Geometry Model

**IMPLEMENTATION – MOTIVE3D**

Motive3D is a three-tier application: an authorizing server (Motive3D@Synthesize, or A3D), a web visualizing client (Motive3D@Visualize) and a Native Geometry Server, as shown in Fig 11.

The authorizing server is an authorization tool that loads geometry models (including native geometry models/polygon geometry models), generates assembly/disassembly sequences automatically/interactively, tests assembly/disassembly paths and sequences, and calculates cost/benefits. Functional features include: 1) support different assembly model formats such as PARASOLID, SAT, IGES, STL, DXF, RENDER, WAVEFRONT, 3D Studio and VRML. 2) Selective assembly/disassembly sequences and paths generation, editing, validation, and simulation. 3) Design-rule checking, e.g. assembly for intersections (interlocking components) checking, clearances between components and accessibility of components. 4) Provide scene navigation manipulation such as Zoom/Pan/Rotate of viewpoints and transformation of assembly model and 5) allow modification of visualization effects such as modifying colors, transparency, materials, background, and geometry rendering style for easy authorizing operations. 6) Streaming the 3D geometry and animation data into a compressed file to avoid traffic congestion on the Internet. 7) Java™ based platform independent coding.

The Native Geometry Server is used to process native geometries, such as collision checking, faceting, and Boolean operation. The native geometry server communicates with the synthesizer via network sockets. A geometry processing
Geometry Server is implemented in C/C++. A protocol is developed for server communication. Native Geometry Server is implemented in C/C++.

**Fig. 11 Deployment Diagram of Motive3D**

The visualizing client is implemented in Java™. It can be accessed through Web browsers to visualize 3D objects interactively. Visualization features include Zoom/Pan/Rotate of viewpoints and transformation of assembly model and visual effect modification, such as changing component color, transparency, materials, background, and geometry rendering style for better collaboration over Internet. The visualizing client allows interactive animation of assembly/disassembly sequences and paths. According to assembly/disassembly analysis results, web client also provides steps and cost to disassembly a selected component.

**CONCLUSION**

Collaborative Virtual Assembly/disassembly provides a distributed solution to simulate and resolve the disassembly/recycling process bottlenecks and to evaluate operational sequences, thus to determine the effect to the environment at each manufacturing and de-manufacturing life-cycle stage of the product. The methodology for Collaborative Virtual Assembly/disassembly for Virtual Manufacturing will result in a new way of looking at product design, manufacturing, and publicizing with respect to the overall structure. After systematically analysis and design with UML, a demonstrative system for Collaborative Virtual Assembly/disassembly. Motive3D, has been developed and described in this paper. It is concluded that the robustness, extensibility, scalability and efficiency of Motive3D system has been greatly improved and hence the product design cycle is expedited and the assembly and disassembly process bottlenecks are reduced.

**REFERENCES**


