A UML Documentation

for an Engineering Drawing Software

COP 290 Project by Shantanu Verma

And

Manas Meena

A UML documentation for E.D. Software

1.Preface

This preface and the following introduction added by Shantanu Verma and Manas Meena in support of the first assignment of COP 290 at IIT Delhi 4th semester.

This document illustrates one way in which the design of an Engineering Drawing software system can be captured.

The assumption behind this assignment is that a good design is much easier to follow than to create.

2.Goals of the assignment

Design and implement a software package for *Engineering drawing*. The package should have the following functionalities:

1. We should be able to interactively input or read from a file either
   i) an isometric drawing and a 3D object model or
   ii) projections on to any cross section.
2. Given the 3D model description we should be able to generate projections on to any cross section or cutting plane.
3. Given two or more projections we should be able to interactively recover the 3D description and produce an isometric drawing from any view direction.

3.A Brief Introduction To UML

The Unified Modelling Language (UML) is the industry standard language for specifying, visualizing, constructing, and documenting the artefacts of software systems, as well as other non software systems. UML simplifies the complex process of software design, making a "blueprint" for construction, and is now the standard notation for software architecture.

UML provides both the structural views and behavioural views of the system. A set of diagrams with different graphical elements is the core part as well as the most expressive presentation in UML. The UML includes nine kinds of diagrams, for the sake of grasp the most representative aspects of the design of elevator system, in this paper only following UML diagrams are used and analyzed:

Use Case diagram shows a set of use cases and actors (a special kind of class) and their relationships. Use case diagrams address the static use case view of a system, these diagrams are important in organizing and modelling the behaviours of a system.

Class diagram shows a set of classes, interfaces, and collaborations and their relationships. Class diagrams are the most common diagrams used in modelling object oriented systems. Class diagrams address the static design view of a system.

Sequence diagram is an interaction diagram. Interaction diagrams address the dynamic view of a system, besides sequence diagram, the other interaction diagram in UML is the Collaboration diagram. Sequence diagram emphasizes the time ordering of messages between objects in the system, while collaboration diagram emphasizes the structural organization of the objects that
send and receive messages.

State chart diagram shows a state machine, consisting of states, transitions, events and activities. State chart diagrams address the dynamic view of a system. State chart diagrams are especially important in modelling the behaviour of an interface, class, or collaboration and emphasize the event ordered behaviour of an object, which is especially useful in modelling reactive systems.

The rest four kinds of UML diagrams are:
Object diagram, showing a set of objects and their relationships; Activity diagram, a special kind of State chart diagram showing the flow from activity to activity within a system; Component diagram, showing the organizations and dependencies among a set of components; and Deployment diagram showing the configuration of runtime processing nodes and the components that live on them.

4.Modelling the Different Aspects of the System

Engineering Drawing is one the most important yet tedious job prone to all kinds of human errors. One of the most effective solution to this problem would be to be mathematically model this problem and let the computer do the job. Since the computing power of the computers has increased tremendously in past years we can save a lot of labour and man power by giving the task of engineering drawing to the computers and saving a lot of time and errors.

1. **Projection of 3D objects to 2D planes**
   The process of converting 3D to 2D is called Projection. There are multiple types of projection but our concern remains on parallel and orthographic projection of 3D objects. Parallel projection are used to create working drawings of objects which preserves the shape and scale of objects which is very important in terms of minimizing any loss of important piece of information.In parallel projection, image points are found at the intersection of the view plane with a ray drawn from the object point and having a fixed direction. The direction of projection is the same for all rays (all rays are parallel). A parallel projection is described by prescribing a direction of projection vector v and a viewplane. The object point P is located at (x,y,z) and we need to determine the image point coordinates P(x,y,z) . If the projection vector v has the same direction as the view plane normal, then the projection is said to be orthogonal, otherwise the projection is oblique.

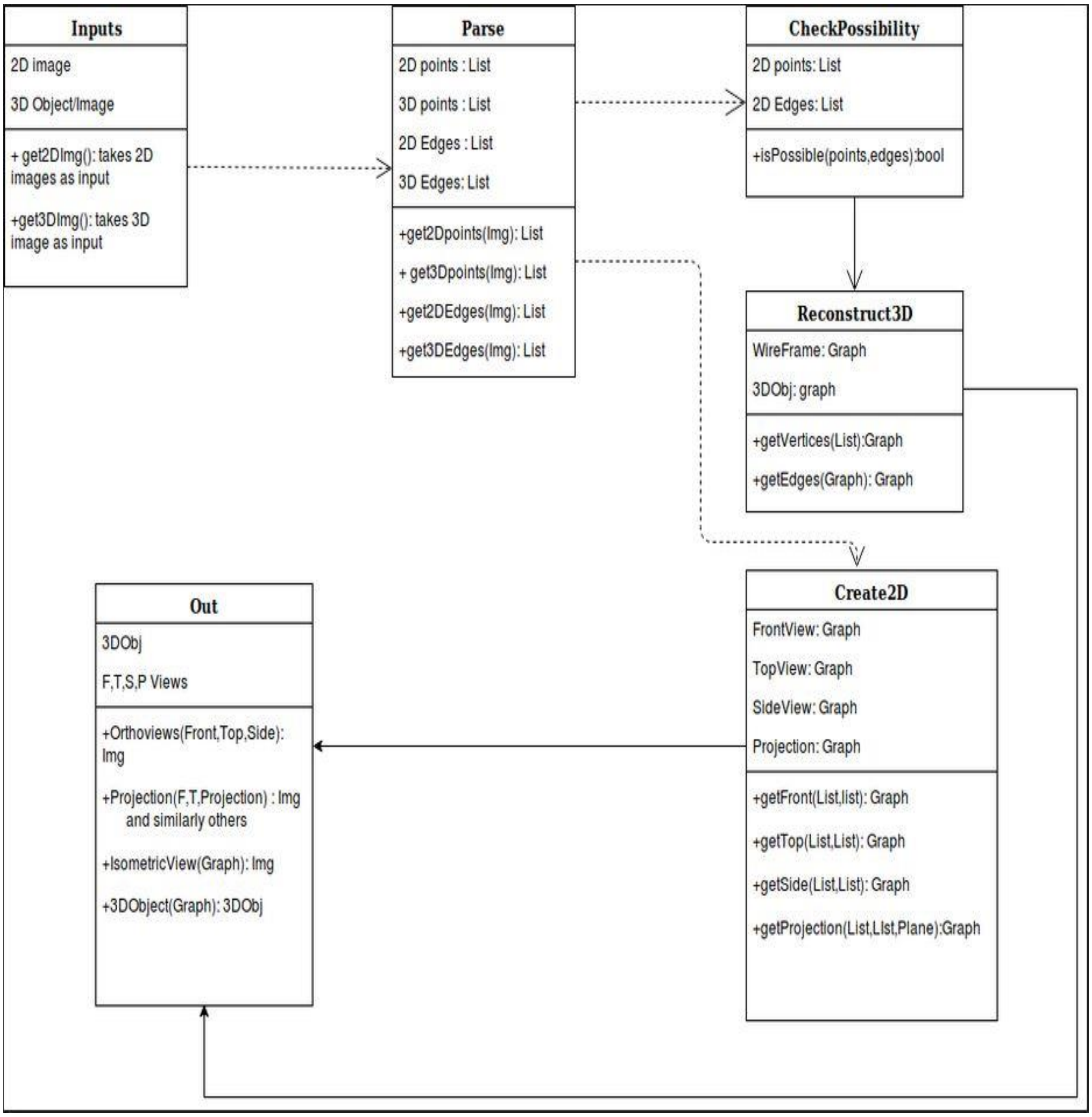
2. 2D to 3D orthographic projections

In this section we will consider the case of 3D object reconstruction from given projections. The first thing that comes to anybody's mind would be how do we do it? How many views are necessary? What is the most effecient approach to reconstruct the 3D model/isometric view with minimal loss of information.

All of these concerns are addressed here. We shall first address the question of how many views are necessary to create 3D model or the isometric view from the 2D orthographic views for the simplicity. A number of techniques have been proposed for the reconstruction of the 3D objects among them B-rep (Boundary Representation) methods are the most studied till now. It consists of following steps: 1. Generate 3D vertices from the 2D projections. 2. Generate possible 3D edges from 3D vertices. 3. Construct possible 3D faces from possible 3D vertices obtained. 4. Construct 3D object from the possible/candidate faces

We will use a bit of different approach to solve the problem and the steps would be really simple. 1. The first step would be the processing of the Engineering drawing given in the input. Processing would involve checking the data for the validity, each view is identified and extracted from the input provided. 2. The next step would be to create 3D candidate edges and vertices from the three orthographic views. We will be analysing the relationship between 3D edges and their projections using the theory of the matrices and would then try to construct a wire frame model. 3. The last step would be an attempt to reconstruct the 3D object using the wire frame model. The information about depth would be used to remove false edges and faces that we would extract from the input engineering drawing. And then finally we will assemble the all the true faces to form the 3D object

5.UML Diagram for the system



Our Model of the package is very simple and free of complications and hassles. We take two types of input i.e either as 2D Image/Projection or as 3D Object then ask user what they want the program to do. The next step is parsing the data or extracting information like 2D points , 3D points , edges etc from the images provided as input from the user. If 2D Image is provided in the input the we run a check on whether a 3D object can be created from the given 2D Projections or if a 3D Object is provided we simply skip this step. Moving further in the analysis, we now come to the step of reconstruction/ projection. We run our algorithm on parsed data and generate the desired output by the user.

6. Conclusion

Given the popularity and notational robustness of the current version of UML, OO technology can be reasonably exploited in real time developments. Current object oriented analysis and design methods focus solely on the software of a system, which is not quite acceptable to real time systems, which are demanding a more pragmatic and comprehensive approach to system

development, rather than just software. There are some aspects of real time systems that need to be addressed:

- Definition of hardware elements and their characteristics;
- Definition of task and task communication;
- Time constraints;
- Modeling of the network

UML, however, if used properly with attention paid to the real time features of a system and
combination of different notes, helps a lot the design and analysis of real time system and can to some extend address above realtime system aspects.

Every diagram in UML is just a graphical presentation of some of the aspects of a system. No single diagram could capture everything about a system’s design view. The UML diagrams have to be combined to express a complete description of a real time system. The three different views of class diagrams of the system can help to understand better the structural aspect of the system.

Some pragmatic methods are given in this paper based on our project experience, which may help filling the gap between requirement and design.

7.References

1. Lecture notes on Computer Aided Designing by Prof. Hegde.

2. A Matrix-Based Approach to Reconstruction of 3D Objects from Three Orthographic Views by Shi-Xia Lid, Shi-Min Hua, Chiew-Lan Taib and Jia-Guang Sun