IOSR Journal of Engineering (IOSRJEN)
e-ISSN: 2250-3021, **p**-ISSN: 2278-8719
Vol. 3, Issue 8 (August. 2013), ||V2|| PP 16-24

HEDCPP: Data Structure for Application Programs Aimed to Engineering Education-Learning

Gilberto Gomes
Depart. Civil and Environmental Engineering University of Brasilia – UnB Brasília, Brazil

***Abstract*: –** The development and/or adaptation of computer systems through the resources of computer graphics and languages object-oriented, like C++ and JAVA, has been shown substantially important in building tools to aid the process of teaching and learning as it enables, both the student and the teacher, explore the essential elements of typical engineering problems, such as shape, configuration and structural behavior. The development of these tools requires, in addition to the resources mentioned, the use of data structures and algorithms that allow a sophisticated modeling, perception and solution to these problems. Thus, this paper presents a robust data structure, called Hedcpp, originally written in C++ language, which can be used as a basis for computer programs aimed at engineering education and addressing the physical problem of form as real as possible, as well as assisting the learning lessons and become much more productive and motivating. Some application examples are presented.

***Keywords:* –** *data structure; modeling; education; oop; c++*

I. INTRODUCTION

According [1], computing environments have proven, over the last decade, an alternative to aid the process of teaching and learning, facilitate the assimilation of content taught by the students and make these processes more dynamic, swift and pleasant. The use of computational tools in teaching holds more student attention, bringing the theory and practice of contributing to learning. Combine the interactivity computing environments and conducting practical activities in academic subjects is an important complement the theoretical part of these, besides giving students the understanding and fixing of concepts learned. Thus, the concepts communicated in lectures interactively between teacher and students and associated with situations abstracted from reality, makes the classes more productive and motivating students.

On the other hand Computer Graphics has assumed an important role as a support tool in the development of computer programs in the engineering field, mainly in geometric modeling, which are required studies and developments of efficient algorithms and data structures sophisticated. This requires innovation both in the programming tool as used in the way that program. In this regard, a new programming philosophy has stood out: the Object Oriented Programming (OOP) [2], which seeks to define the objects - entities involved in the system - and its classes and relations, instead of decomposing the program into procedures or functions.

Develop and adapt techniques Computer Graphics, Geometric Modeling and Computational and Data Structure for making object-oriented programs that allow computer simulation of engineering problems, or enable the understanding and learning in the different subjects of engineering courses, such Strength of Materials and Static Structures, interactively, didactic and visual, has been studied in [3], in which some programs preprocessing graphic addressing POO were developed, namely the TRUSS_GI and BEMOO_GI.

In this context, continuing works [3, 4], this paper presents a data structure called *HeDcpp*, based on the classical structure Half-Edges [5] and on the concepts of OOP, able to handle modeling, representation and manipulation of models plans a didactic, visual and interactive, whose purpose is its application in programs aimed at teaching and learning in engineering, especially in those subjects that has as objects of study in modeling physical-geometrical properties, mesh generation and structures analysis with beams, trusses and frames, for example. Originally, the *Hedcpp* was written in C++ language by fully supporting OOP.

The content of this paper is organized as follows: section II a brief description of the topology, modeling and data structure; in section III the main concepts of OOP and class model Hedcpp; in section IV, the main features and use of the Hedcpp are shown through a Graphic User Interface (GUI); application examples will be presented in section V and conclusions and final discussion in section VI.

II. TOPOLOGY, MODELING AND DATA STRUCTURE

A. Topology and Modeling

The topological description of geometric models requires the representation of relationships surrounding its primitive entities. In general, the planar topology model is formed from the combination of three primitive topological entities: vertices (V), edges (E) and faces (F), as illustrated in figure 1a. With these three primitive is

possible to establish nine fundamental relationships (Fig. 1b) which describe the proximity and arrangement of a specific group of entities in relation to any one entity.

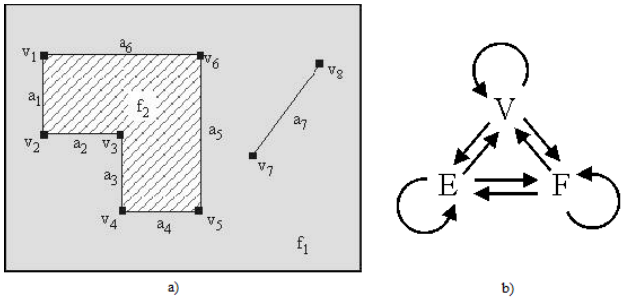

Figure 1 - a) A planar graph and its topological elements;
b) adjacency relations between topological elements.

In addition to the geometric information contained in vertices, edges and faces, the data structures representing the contour consider the topological representation of the information model, which are considered by some explicit store nine adjacency relations. In turn, the creation and manipulation of these models can be performed through operations which consider the geometrical and topological properties of arbitrary planar models. According to the theory of manipulation planar models, a fewer number of operations is sufficient to describe and manipulate most models planes of practical interest. In the manipulation of contour models, the realization of these operations is done by entities called Euler Operators [6].

B. Data Structure

In [3], the representation of the topology of the model was treated through a data structure, based on the well-known structure Half-Edge due [5], capable of recording all steps intermediaries that describe the model in a sequential manner. In [4], the authors introduced some modifications to the original half-edge structure. These modifications were based on the concept of duality, aiming to provide similar relationships to the faces and vertices of a model. A first modification consists in disregard of cycles (loops) in the model hierarchy. It is thus apparent that the data structure has the resultant reduced form of figure 2a, while the analogy between the relations of vertices and faces can be checked, since both entities relate directly with half-edges. Moreover, it was necessary to use special edges, called nil edge, for the representation of faces convenient containing holes, as illustrated in figure 2b. Although these edges play a role similar to the cycles of the data structure of [5], they are considered to be entities like edges whose vertices are not identified.

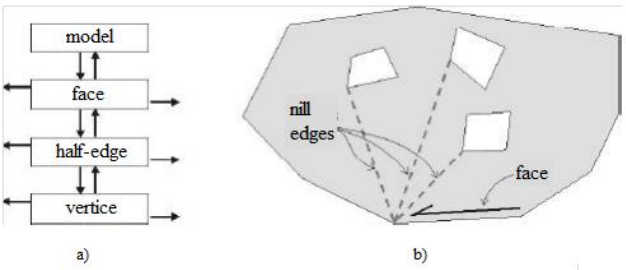

Figure 2 - a) Storage of the hierarchy of entities in the data structure used;
b) use of nil edges to represent faces containing holes.

The type of representation technique used in [3, 4] is a technique for boundary representation (B-Rep) [7], where the model surfaces (with their edges and vertices) are represented explicitly. In this type of modeling the contour of a three-dimensional solid is a two-dimensional surface which is usually represented as a collection of faces. In turn, the model faces are depicted in terms of dimensional curves that define the contours. The models typically represent contour faces in terms of explicit nodes a data structure boundary, thus enabling many alternatives to represent the geometry and topology of a contour model.

In this work, the data structure initially developed to represent planar models of boundary elements will be used to represent any models in engineering programs aimed at teaching and learning of the following content: Static Equilibrium and Structures Analysis at the undergraduate level and analysis through the Boundary Element Method-level graduate.

III. OOP AND HEDCPP CLASSE MODEL

A. Object Oriented Programming (OOP)

The object-oriented technology is based on object models, which encompasses the principles of abstraction, hierarchy, encapsulation, classification, modularization, relationship, concurrency, and persistence. According to [2], the paradigm of OOP is the real world of computer systems, consisting of several objects that interact with each other and has its emphasis on code reuse. In this sense, object orientation is a paradigm of analysis, design and programming of systems based on the interaction of objects - entities abstracted from the real world. Table I presents the main concepts of OOP.

TABLE I. KEY CONCEPTS OF OOP

Terms	Definition
Object	An instance of a particular class
Class	A template for objects to encapsulate data, functionality and behavior
Member	A variable or method (operation, service) within an object
Instance	An object of a particular class
Encapsulation	The binding of data and methods into a class of objects
Inheritance	Means that derived classes inherit the variables and methods from their base classes, while possibly redefining or adding new variables and methods; this creates a hierarchy of ancestor classes
Polymorphism	The ability of classes in a hierarchy to share names for methods that behave appropriately to the particular class for which they were designed

B. Hedcpp Class Model

Based on the concepts of OOP, the class model of figure 3, described below, was designed to represent all entities involved in the modeling.

- GraphicPrimitives class: this is a base class that represents all geometrical model primitives. The derived objects inherit the only two methods of this: Draw() and Select(), enabling them to draw and to guide the selection, respectively. Point, Line and Face are objects derived from this class.
- GeometricModel class: this class has methods and attributes that define themselves. Its functionality is directly related to the definition of the model geometry. An object of this class aims to perform the actions necessary to the process of building geometry. These actions, which are performed on the graphics primitives are represented by various methods that this class are related to the purpose of defining the geometry, in a process of associating and storing the primitive through its attributes. The proposed data structure is also implemented in this class, allowing your object can be updated every modification during the construction process. That is, the data structure enables the updating of geometry and topology of the model at runtime. The objects of this class have references to all objects derived from GraphicPrimitives with the only purpose of store.
- HalfEdge class: this is a base class without methods, just data members that reference other entities topological model. This class defines the objects half-edges with references to edges and faces. Therefore, no implementation of methods in this class is done.

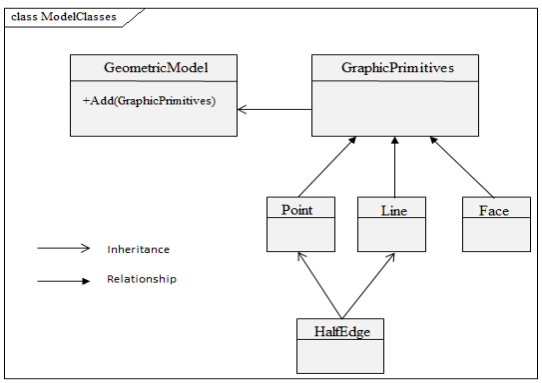

Figure 3 – Class diagram for modeling process.

IV. THE HEDCPP AND ITS CHARACTERISTICS

In order to visualize and understand the operation of the data structure, a graphical interface was developed solely for this purpose, in order to allow generation of model from a sequence of actions triggered by the user. The interface consists of a graphical program based on windows systems, where actions are controlled by menus, toolbar buttons, keyboard and mouse. In this interface, the user provides as a basic point via the mouse, with each new insertion whole model is updated to ensure consistency through the same data structure. Thus, the main features of the interface are:

- Generation of planar models;
- Automatic identification of the topology of the model (recognition of vertices, edges and faces) through the structure of half-edges implemented;
- Manipulation of the model at runtime through common editing commands (add, select, delete, etc ...). Storing primitive model (vertices, edges and faces) was performed by doubly linked lists. This type of storage is advantageous since these lists do not require a defined size, limited only by available memory of the computer system used. Briefly, the interface program is organized in five hierarchical levels, as illustrated in figure 4, where the first level allows the user to provide the type primitive vertices and edges that are likely candidates to be introduced into the model. Before introducing the candidate vertices and edges, the level of consistency check performs the following tasks:
- Check if the primitive candidate already exists in the model;
- Check possible intersections with the primitives in the model.

After the consistency check, the primitives can be stored in their lists. The next level is through the descriptions of the topological Euler operators [6], and some functions that assist in the appropriate call operator. Finally, the lowest level considers the topological data structure and attributes of each element. Access to this information is accomplished through Euler operators, the next higher level.

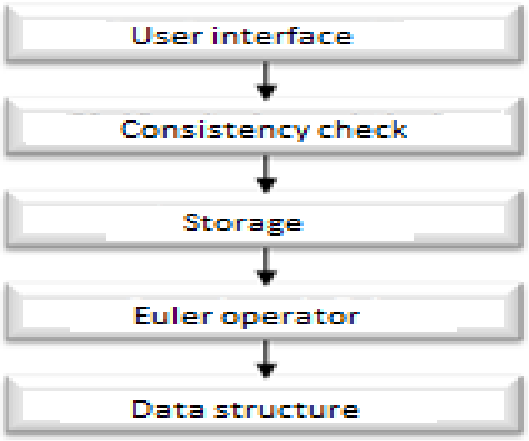

Figure 4 – Hierarchical levels program.

A. GUI Syntax

The model program interface is based on the concept of Actions. An action requires an object of a given class and requires that the left mouse button is pressed in the client area so that it is executed. These actions are defined in the handling functions of the buttons (located on the toolbar), and implemented the program in class View.

It has been that each button is associated with a treatment method and the same for each method are described in the number of shares, the object and the function that should run for each action. All of these functions must have the same format, namely a type CPoint argument and a void return type.

The execution of the sequence of actions is controlled by the identifier object class View through current actions "IdCurAction". Each time a button is pressed tools, the treatment function initializes the identifier of the current action. All actions are performed through LButtonDown() function object class View. The implementation of each action requires that the left mouse button is pressed. Figure 5 illustrates the layout and operation of the interface through the concept of actions.

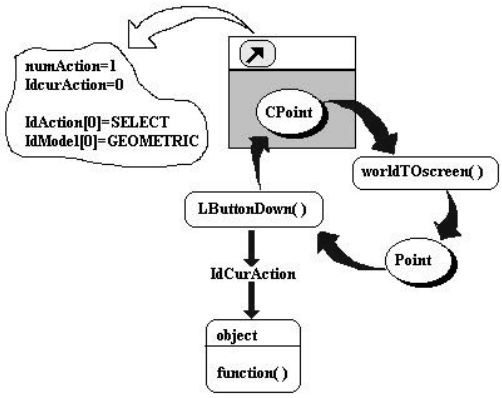

Figure 5 - Syntax interface program.

V. APPLICATION EXAMPLES

The following examples illustrate the application data structure Hedcpp in three different programs listed below. Initially its main features are shown through the interface developed.

- TRUSS_GI: Preprocessor graphical object-oriented for building and understanding of physical and geometric modeling of Plane Trusses. This program was written in C++, keeping the characteristics of Hedcpp in its entirety;
- BEMOO_GI: Preprocessor graphical object-oriented for building and understanding of physical and geometric modeling by Boundary Element Method. This program was written in C++, keeping the characteristics of Hedcpp in its entirety;
- SENG2D: Computational Platform for Teaching and Learning in Engineering. This program is written in Java due to the need to work multiplatform and to extend it in applications like web while retaining, in part, the characteristics of Hedcpp.

A. Main Features

Fig. 6 illustrates the generation, updating and creating new models at runtime, that is, if there is intersection between segments, new segments are created. The recognition of faces (regions) or holes are also automatically identified from the relationship between half-edge and vertex or nil edge.

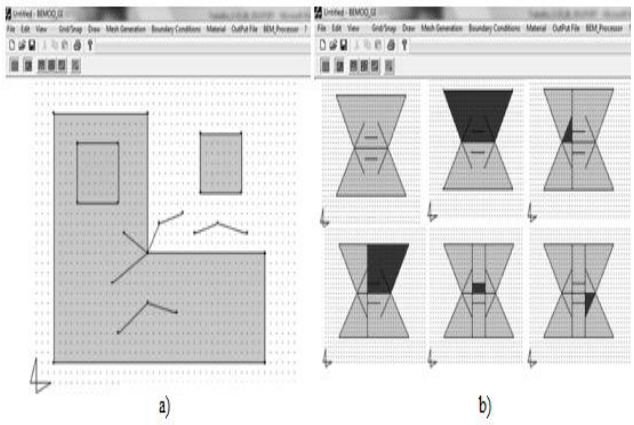

Figure 6 – a) geometry of the primary elements V-E-F; b) intersection, updating and creating new models.

B. TRUSS_GI

Fig. 7 shows the geometric model for a plane truss by using only the Line object. Initially (Fig. 7a), the truss consists of three bars and, while drawing the other three bars, the intersection algorithm and updating the topology of the model produced a truss of figure 7b with nine bars. In Fig. 7c the selection object interacting with support and loading objects for physical modeling of the truss. This interaction is achieved through friendly dialogue between user and program.

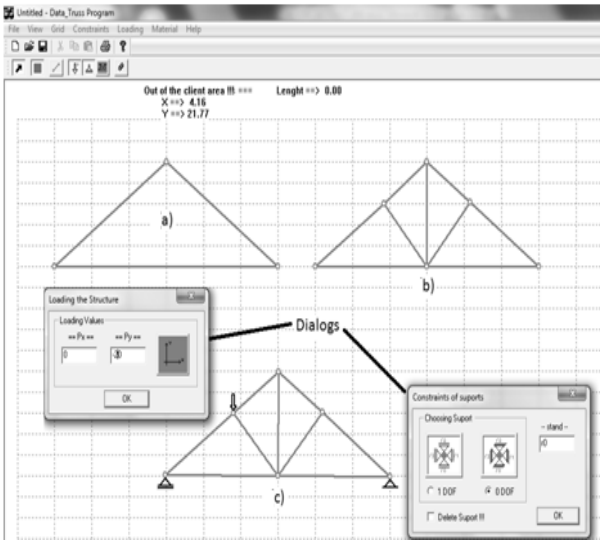

Figure 7 – TRUSS_GI program for modeling of planar trusses.

C. BEMOO_GI

Fig. 8 shows a rectangular plate modeled by the Boundary Element Method. In this, the geometry was designed using only the primitive line, and automatically the Hedcpp recognizes primitive Face (shaded region, Fig. 8a). The physical modeling (Fig. 8a) was drawn from the relationship between the select object and primitives with its dialogues (Dirichler and Newman variables, Fig. 8c). In Fig. 8b, we have the boundary element mesh that was generated from the relationship between the Select object and primitives with the mesh generation dialogue, which starts from choice of the element and splitting edges or selected face.

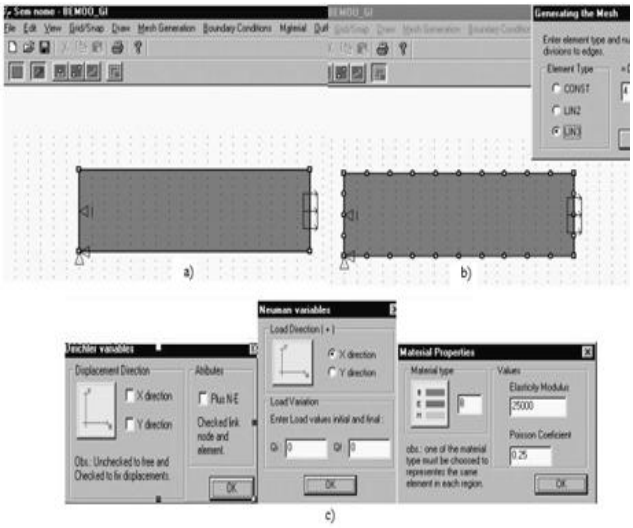

Figure 8 – BEMOO_GI program for BEM modeling.

D. SENG2D

Fig. 9 presents the visual model of the SENG2D platform with their shortcut buttons (tasks/roles) to assisting the user, which consists of:

- Client Area - for model construction engineering (beams, frames and trusses) through the primitive and Select, Supports and Loads objects;
- Area Diagrams – for assembly Free Body Diagrams and Internal Forces;
- Equilibrium equations - for writing the equations of static equilibrium;
- Report - print the drawings, calculations and results in pdf/jpeg formats.

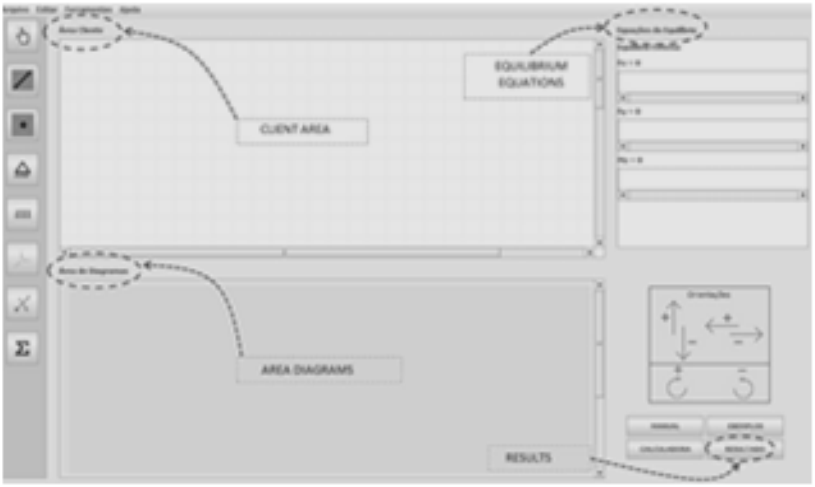

Figure 9 – SENG2D visual model and shortcut buttons.

In order to illustrate the features of SENG2D, consider the model calculation of the beam of figure 10.

Figure 10 – Beam model of the example.

Figures below illustrate the tasks for the module I, understand the external equilibrium, which are:

- Task 1: Physical-geometrical modeling (Fig. 11);
- Task 2: Assembly Free Body Diagram (Fig. 12);
- Task 3: Writing the equations of equilibrium (Fig. 13);
- Task 4: Calculation of the reactions (Fig. 14).

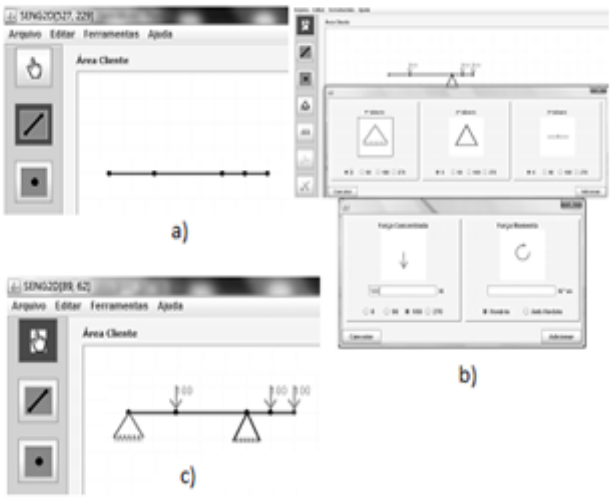

Figure 11 - Task 1: a) defining the geometry, b) interacting with dialogs for physical modeling, and c) final drawing of the beam.

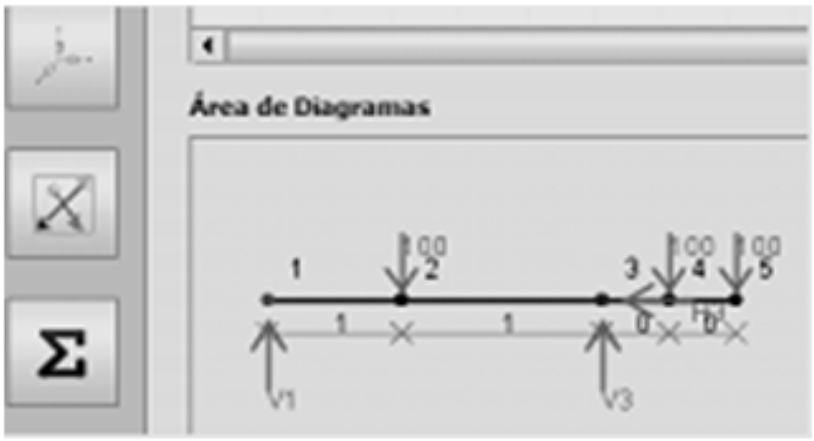

Figure 12 - Task 2: Construction of the Free Body Diagram.

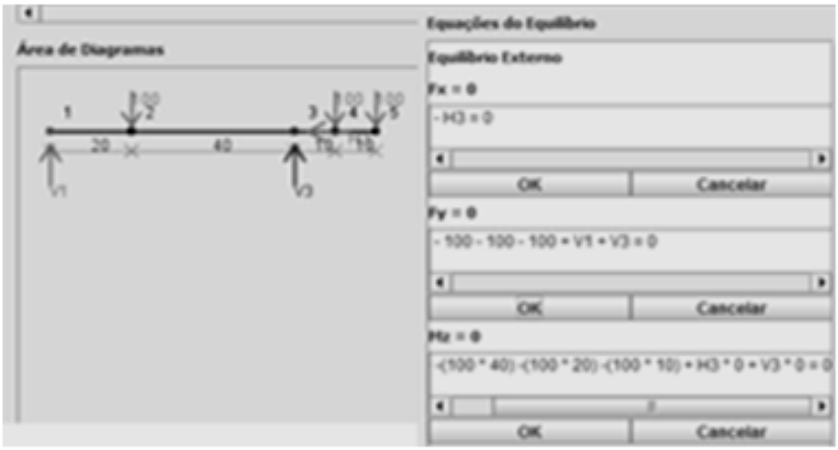

Figure 13 - Task 3: Writing the three equilibrium equations.

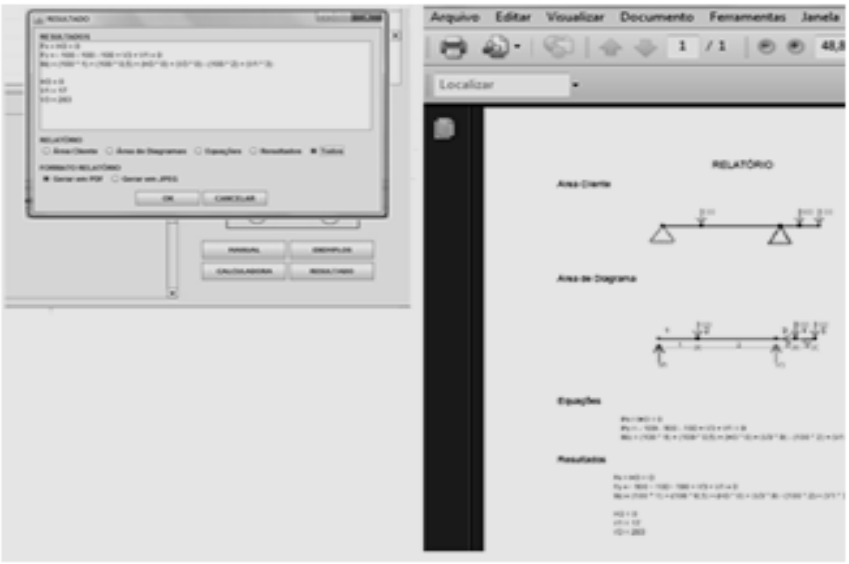

Figure 14 - Task 4: Calculation of reactions and report in pdf.

VI. CONCLUSIONS AND FINAL DISCUSSIONS

This work demonstrated the use of an alternative data structure based on the concept of dual planar models. The framework developed was consistent as the dual character, providing topological information similar to the model vertices and faces.

The program developed for verification of the functionality of Hedcpp allows the geometry of the model is introduced in a friendly manner, sequential and interactive. This is due to the versatility of the proposed data structure and the classics algorithms of Computer Graphics. Moreover, the program also allows instant viewing of the model generated, thus avoiding the tedious task of defining the geometry in text form (data file). Once generated the geometric model, the program allows you to enter the boundary conditions (loads and restraints) and assign the specific material (Poisson's ratio and Young's modulus, for example) the same, through dialogue quite friendly.

The platform SENG2D presented here is initially focused on understanding the external equilibrium structures - the object of study of engineering disciplines such as Structural Statics and Mechanics of Materials, and aims to support teaching and learning in a friendly, interactive and sequential allowing both the teacher and the student immediate viewing the generated model, the stages of analysis and results. His versatility in teaching and learning external equilibrium is achieved due to the elegantly didactic, visual and interactive of its objects, when they interact among themselves and with others, searching, retrieving and exchanging information, and simulating faithfully reproducing content.

An advantageous feature of the platform is developed reuse process, that is, when it is desired to use a class already ready and enhance its functionality, e.g., the account equilibrium internal forces diagrams applicants would be easy to implement, since OOP allows the insertion of code with versatility by simply implement the method in the appropriate class and enjoy the results for the reactions.

Finally, it is worth noting that programs TRUSS_GI and BEMOO_GI are not yet complete, being objects of research phases and post processing. As for SENG2D, so the finished module II on the internal forces, it will be presented to teachers and students of civil engineering at the University of Brasilia with the proposal to apply the disciplines mentioned.

REFERENCES

[1] Silva, M. A. da, “Protótipo de uma ferramenta para auxiliar no ensino de técnicas de programação”. Universidade do Planalto Catarinense, Lages, 2003.
[2] Booch, G., “Object-Oriented Analysis and Design with Applications”, The Benjamin/Cumming Publishing Company, Inc., 1994.
[3] Gomes, G., “A data structure for representing two dimensional boundary element models”, Master’s thesis, University of Brasilia, Brazil, 2000. (In Portuguese)
[4] Gomes, G. & Noronha, M. A. M, “Estrutura de Dados para Geração de Malhas Bidimensionais de Elementos de Contorno”, XXI CILAMCE, Rio de Janeiro, 2000.
[5] Mäntylä, M., “An Introduction to Solid Modeling”, Computer Science Press, Rockville, Maryland, 1988.
[6] Baumgart, B. G., “A Polyhedron Representation for Computer Vision”, AFIPS Proc.,1975.
[7] Lienhardt, P., “Topological Models for Boundary Representation: a Comparison with n-Dimensional Generalized Maps”, Research Report R90-10 Department d´Informatique, Université Louis Pasteur, Strasbourg, France, 1990.