Analysis and application of engineering data in the STEP file format

A I Razumowsky
Trapeznikov Institute of Control Sciences RAS, 65, Profsoyuznaya street, Moscow 117997, Russia
E-mail: razumowsky@yandex.ru

Abstract. The method for encoding graphic data using the STEP standard was considered. This standard allows you to describe the full life cycle of a product from design to quality control. The possibilities have been investigated for transmission and transformation of STEP data. The order was analyzed for storage and synthesis of data based on dividing a software graph into subgraphs of local algorithms. There was an algorithm developed for syntactic analysis of STEP file format data, and software implemented for displaying graphic information, which allows you to control 3D visualization of machine-building parts.

1. Introduction
When they consider the issue how to transmit and encode information, CAD developers initially tried to present this issue as a set of conditions for the formation of homogeneous data structures bound in a graph. Otherwise, descriptors of such structures had to be displayed sequentially with assigning a unique number to each enumerated element.

The first method of data recording is typical for a wide range of CAD systems that provide not only fast input and output of information, but also data storage within the system, as well as accumulation and masking of information from the internal database (DB) upon request. Such information movements are characterized by zero data loss [1].

In other cases, during intersystem information interaction, a sequential recording of primitives of translated data is applied, where the identifier is either the ordinal number of a primitive, or a special unique index that additionally marks each primitive. Such index indicates the location of the primitive in the general structure of an object. In turn, the qualitative logical entity of the primitive is determined by its name. Thus, the name and unique number combined will provide a complete description of the simplest named object node. The entire object can have descriptions of dependencies and interactions for individual nodes, primitives (IGES, STEP) [2-4].

This standard developed by the ISO Institute is one of the first in a family of specialized CALS standards, and is a typical example of a new generation information standard, and subsequent CALS standards are built on its model [5,6].

2. Step format parsing methods
The specified goal is to display graphic 3D information, therefore, the main attention should be focused on analyzing and transforming graphic data contained in certain formats with omitting other information respectively.
When data are transferred from the system, one of the main requirements is the formation of a named node, the bond structure of which is known (published as a standard). At the same time, during the coding process, two directions of bonds can be added to the general description of the node (for example, coordinates): to the root and from the root of tree. In reading such a file, the program, in the general case, provides a search for the next node until it has no downstream bonds. In order to increase the performance of a program unit, binary search is widely used in a sorted input array of information.

Let's call a local input algorithm a data structure that contains distinctive quality of input information, and an array of links reflecting all kinds of bonds between data of the current structure that are typical for an input operation, and other program nodes. In other words, an input algorithm is a kind of logic for data distribution that characterizes the mechanism for a program reading an external file and encoding file information into software packages for storing a database.

In simplest cases (figure 1), finding structures and rules of a local input algorithm is not difficult. Each structure is a descriptive part of an input primitive or a family of similar primitives. Data inside a structure are accumulated sequentially, as the next translation string is re-encoded.

![Uniform storage tree](image1)

**Figure 1.** Uniform storage tree.

In general, the input operation is represented as follows:

$$ R = \sum_{i=0}^{N} (S_i, F_i) $$

where $R$ is a complete operation of reading data from the input stream (file), $S_i$ - $i$-data structure, $F_i$ - $i$-function of processing information received at the input to fill the $i$-structure. The meaning of parameter $R$ resulting in (1) can indicate the cyclical nature of a program code.

In case of a STEP stream, a link or a set of links to identifiers of other nodes is included in primitive data description, while the ability to have links both forward and backward forming a forest of link trees allows primitives of the object to be arranged in any sequence (figure 2).

![Fragment of the linked data storage tree](image2)

**Figure 2.** Fragment of the linked data storage tree.

If we take into account the simplified way to organize data storage of the input stream in a sequential form, then a simple connection between structures of a local input algorithm and storage structures of external data can be implemented through a single GET function. After that, the simplest local input algorithm can be written as follows:
\[ R = \sum_{i=0}^{N} (S_i, F_0) \]  

where \( F_0 \) - now the GET function implements an algorithm common for all data primitives for reading symbolic information, accumulation in internal structures of a program unit and in a 3D model storage unit, and the procedure for additional filtering of data (figure 3).

![Figure 3. Functional model of the software solution.](image)

When designing a program unit for data input-reading (figure 3) like other program components, the main attention is paid to finding a local input algorithm for the corresponding unit. Structures that will ensure storage of data being entered and a database structure of the system itself must be unambiguously connected by a dissemination algorithm from temporary files that include the logic of primary and secondary information processing to the storage of a program unit or, in the absence of which, to a data display screen.

When considering information as a collection of typed data, i.e. data enclosed in abstract envelopes, we can assume the existence of a simple algorithm for accessing information (operations for writing and reading a file, and operations for encoding and visualizing system statuses). This algorithm corresponds to the program graph, the nodes of which will be both the main components-units determined by the task requirements - reading, re-encoding, visualization a file - and additional ones allocated from the main units for autonomous independent implementation; for example, a unit for filtering primary information. Oriented edges of a graph are information transmission paths, while the internal data transmission is carried out through a common storage unit, which accordingly weakens bonds between components and simplifies the support of the software system life cycle. A single common storage unit can also help reduce homogeneous internal i/o. Thus, the GET function of a local input algorithm for each component-unit will be reduced to just an assignment operation with the exception of the reading and visualization units.

### 3. Functional blocks of software implementation

When decomposing a software system into separate components, one should proceed from the main requirement for the components, the maximum possible autonomy. Then it is obvious that the reading unit forms a stand-alone component that can communicate with the rest of the program through the database. The database must functionally perform a task of saving data in a certain form, and provide convenient and quick access to such data. Where data arriving at the input of a system have a complex structure as in the STEP file format, suggesting the need for additional calculations when receiving a complete description of a 2D or 3D graphic element, a database is additionally divided into three components: a unit for storing primary data, a filtration unit and a storage unit for sorted data (figure 3).

A primary data storage unit is organized to support initial computations, and actually to provide resources for the GET function of a reading unit. The GET function is implemented based on the cyclical content of the core of the function, and on the definition of a "single primitive" for STEP data. Such an information element in a STEP file is a line with a "#" symbol in a zero position.

The FACE_SURFACE and ADVANCED_FACE elements are the initial element of the description of a 3D object that is part of the CLOSED_SHELL. Defining elements – they are for example curves:
LINE, CIRCLE, BEZIER_CURVE and others; and also surfaces: CONICAL_SURFACE, BOUNDED_SURFACE and B_SPLINE_SURFACE_WITH_KNOTS. When identifying 3D elements mainly in the form of fragments of surfaces, their boundaries are found by defining curves with start and end points. Or, when such boundaries are not defined, the desired object will be a closed (partially closed) surface: a full cone or a sphere. The intermediate result of identifying a 3D element, for example, a conical surface, which is obtained from the STEP description, is, in this case, the radius and center of the cone base, the slope of its generatrix line, and two polygons that limit the conical surface. The final stage of obtaining a 3D element will be breaking the surface into last elements and finding the crossing of its boundaries specified as polygons with the surface itself.

Each separate section for estimating a 3D object is a local algorithm coupled loosely with other program parts, which is developed firstly based on graphic and geometric requirements for data structures and general organization of the code.

The get_step_token function implements the order and rules for parsing a single element of STEP data. Primary information is transferred by means of special structures, capsules, conventionally designated by X and Y. Their number may be different depending for example, on additional tasks solved at the file reading stage including the availability of debugging functions and resources in the program code, or functions for immediate transcoding, etc. In a simple case, there is only one structure, which is filled with data that correspond to a single element of the STEP file. Such structure includes links, indexes and valid data. It also includes boolean flags that are found in a STEP line, and a primitive identifier:

```c
struct X{
    long index;
    bool booling;
    char command [80];
    links* link;
    links* ilink;
    ddlinks* ddlink;
};
```

SHAPE_DEFINITION_REPRESENTATION objects designated by STEP format supergroups are characterized by the fact that they lie at the root of the tree of links that together form the final 3D element. Thus, an additional goal of a reading unit is to find such roots whose sequential calculation implemented in other units will lead to finding the entire tree of links of the final 3D element, and such links will be used to build a complete 3D model.

Later, such an array, which is the result of the reading unit, is controlled by internal functions of the storage unit for sorted data. First of all, consideration should be given to functions of allocating and freeing memory to store X-data (variable content) in a variable-length array, as well as error handling operations and root data focus functions that provide access to data.

A unit for coding operations is the main link of the program implementation, insofar as it is here that the main mechanisms are concentrated both in terms of quality and required resources, for processing information obtained at the previous stages. The result of operations performed by such unit is 3D information directly intended for visualization.

The main cycle of the coding operation unit includes functions to calculate geometry of, reproduce and transform 3D objects. The main cycle is located in the identification_bodies function that calculates closed shells of the STEP file. Most 3D objects are stored by means of the CLOSED_SHELL record. Therefore, the main line when searching for the beginning of geometric information is the named line CLOSED_SHELL. When corresponding lines are processed sequentially, a paired element of internal representation (Body, Points) is formed: - Body array of triangular surfaces, where a triangle is specified by three integer vertices, - Points, an array of vertex coordinates related to this instance of Body and listed in the order that corresponds to indices of these vertices in triangles of the instance.
The coding operation unit is closely related to the unit for storing 3D models as set forth in the specification, through the transfer of the link unit to the memory area for Body and Points data, the combination of which makes up description of 3D elements that form the model.

Due to the fact that 3D data are simplified, it is easy to apply an encoding operation to them. This operation will consist of the GET function that has only 2 arguments, the Body and Points arrays. The rendering process of 3D elements will also contain the GET function of the renderer based on the AddFaces method from Direct3D library. It should be noted that this may require additional configuration of the rendering environment, which complicates the renderer structure.

4. Discussion and results
At the stage of reading the STEP file, the program code will consist of a set of selection statements, in the body of which there is an output for processing CLOSED_SHELL. On the other hand, the process of decrypting this shell is simply normalized based on uniformity of character strings. The process details are such that, firstly, temporary information is accumulated about the indexes of communication with other objects and final data (or the data has already been accumulated by the time of visualization), and secondly, each accumulated index element is, in turn, considered as a separate object. If this object is complex, i.e. contains lists of reference elements and coordinates, then each of the corresponding elements and coordinates is examined consistently in the cycle. The recursive nature of the GET function of this unit is clearly observed here, although in a specific case, the implementation was carried out through step-type internal cycles.

The main goal of each specific local algorithm is to perfectly identify a graphic element. Consequently, the availability of one or another local algorithm in a program will depend on the qualitative composition of initial data, simply, on the possible existence of the relevant element in the STEP description. So, the description of conical surfaces will correspond to the conical_surface algorithm, and the description of flat polygons will correspond to handle_plane. In order to summarize the study of all kinds of 3D curves, the local algorithm edge_curve is used.

The edge_curve procedure receives input parameters as an array of points and, especially for complex curves of the B_SPLINE_CURVE_WITH_KNOTS type, additional data on the quality that these points are arranged with. As a result, we obtain a polygon or a set of polygons, and a number that indicates the nature of interaction between polygons. If a curve on a plane is considered, then all polygons will belong to such plane. If a curve refers to a surface (for example, cylindrical), polygons will either be the boundaries of this cylinder, or will cut holes in its shell.

Of particular interest are local algorithms for transforming 3D surfaces bounded by various curves into flat polygonal images that can be directly triangulated and rendered. When developing these algorithms, it turned out that it was possible to use a principle common to all surfaces, the flat pattern principle.

We proceed similarly when we find an algorithm for dividing β - spline surfaces that have similar restrictions in the form of various curves. When representing a given β - spline surface as an ordered set of flat plans, a curve that belongs to such surface will consist of fragments, each of which will lie in a certain plan and can be processed in a timely manner within this plan.

Examples of visualized complex engineering items of resulting software implementation are shown in figures 4 and 5.
5. Conclusions
The method presented in the article for the analysis and transformation of STEP machine-building standard data, as well as the software implementation of 3D data visualization and management has a number of advantages: the strategy of using local algorithms to identify objects and their properties simplifies data parsing; the use of a generalized structure to describe elements of data analysis unifies the process of developing a software solution; and breaking a software solution into units reduces functional dependencies between the internal storage and external requests to its data.

Future work is related to the study of possibilities for parallel processing of STEP data, as well as the creation of special data storage structures of enhanced reliability in order to ensure information security [7]. Issues of reducing a data volume in files while increasing the read speed remain relevant [8]. A significant prospect for this approach is guessed in big data management [9,10,12,13] due to introduction of heterogeneous and hybrid storage systems such as Hadoop and Spark. Finally, a wide horizon opens up in the creation and processing of hierarchical data related to the Internet of Things.

Figure 4. Engine.

Figure 5. Hydraulic unit.
(IoT) [11,14], which promises significant advantages in the perception, analysis and synthesis of distributed information.

References
[1] Chao P Y and Wang Y C 2001 A data exchange framework for networked CAD/CAM Computers in industry 44(2) 131-40
[2] Shishigin D S 2016 On choosing the technology of application software integration with a CAD-system Trudy SPIIRAN 47 211-24
[3] Schweitzer M A and Randrianarivony M 2011 Treatment of General Domains in Two Space Dimensions in a Partition of Unity Method Meshfree Methods for Partial Differential Equations V (Berlin, Heidelberg: Springer) p 27-49
[4] Xu X W, Wang H, Mao J, Newman S T, Kramer T R, Proctor F M and Michaloski J L 2005 STEP-compliant NC research: the search for intelligent CAD/CAPP/CAM/CNC integration Inter. J. of Prod. Research 43(17) 3703-43
[5] Aleksandrova S V, Vasiliev V A and Letuchev G M 2018 Digital technology and quality management Proc. 2018 IEEE Int. Conf. "Quality Management, Transport and Information Security, Information Technologies" (IT&QM&IS) (St. Petersburg, Russia: IEEE) p 18-21
[6] Ulrich K T 2003 Product Design and Development (New York, US: McGraw-Hill Education)
[7] Xu J, Wei L, Zhang, Y, Wang A, Zhou F and Gao C Z 2018 Dynamic fully homomorphic encryption-based merkle tree for lightweight streaming authenticated data structures Journal of Network and Computer Applications 107 113-24
[8] Schellenberg F M 2017 Resolution enhancement techniques and mask data preparation Electronic Design Automation for IC Implementation, Circuit Design, and Process Technology (Boca Raton, FL, US : CRC Press) p 541-86
[9] Krish K R, Wadhwa B, Iqbal M S, Rafique M M and Butt A R 2016 On efficient hierarchical storage for big data processing 16th IEEE/ACM International Symp. on Cluster, Cloud and Grid Computing (CCGrid) 403-8
[10] Wang X, Yang L T, Kuang L, Liu X, Zhang Q and Deen M J 2019 A tensor-based big-data-driven routing recommendation approach for heterogeneous networks IEEE Network 33(1) 64-9
[11] Xiang X, Liu W, Wang T, Xie M, Li X, Song, H, and Zhang G 2019 Delay and energy-efficient data collection scheme-based matrix filling theory for dynamic traffic IoT EURASIP J. on Wireless Communications and Networking 1 168
[12] Choi D, Jeon M, Kim N and Lee B D 2017 An enhanced data-locality-aware task scheduling algorithm for hadoop applications IEEE Systems J. 12(4) 3346-57
[13] Hu F, Yang C, Jiang Y, Li Y, Song W, Duffy D Q and Lee T 2020 A hierarchical indexing strategy for optimizing Apache Spark with HDFS to efficiently query big geospatial raster data Inter. J. of Digital Earth 13(3) 410-28
[14] Riad K, Huang T and Ke L 2020 A dynamic and hierarchical access control for IoT in multi-authority cloud storage J. of Network and Comp. Applications 160 102633