Hybrid client-server implementation and microservice architecture of automatic documentation analysis software

Anastasia A. Dzyubanenko [1] and Alexey V. Rabin [1]

[1] *Saint-Petersburg State University of Aerospace Instrumentation, SUAI, 67, Bolshaya Morskaia str., Saint-Petersburg, 190000, Russia*

Abstract

An approach to the construction of an adaptive architecture for enterprise software has been developed in order to increase the efficiency of automated processing of documents using semantic and cognitive technologies. The proposed approach takes advantage of the existing methods of organizing the architecture of applied software. It is substantiated that the architecture of the developed software for automatic cataloging should have a hybrid client-server implementation, including elements of modular and microservice architecture. It is shown that a significant reduction in the costs of cataloging, checking the completeness and inventory of documentation, as well as an increase in the quality of design are provided through the semantic analysis of documentation using a knowledge base that is updated automatically.

Keywords

Weakly structured information, automatic analysis of documentation, semantic and cognitive technologies, data cataloging

1. Introduction

Documentation operations, performed even with the existing automation tools, are extremely time-consuming, especially when it comes to a fairly large enterprise. Only digitalization and cataloging of a paper archive of technical documentation of a concern-level enterprise is estimated at more than 200 million rubles and requires at least 5 years. Checking the integrity of the archive of one product, taking into account the entire structure of cooperation, takes more than 1.5 years and does not give a 100% result due to the lack of automatic verification tools for specifications [1, 2].

There are other problems as well. For example, solutions from 15 years ago can be used in modern products, and in the event of damage to paper documents, the situation becomes critical. There is also no mechanism for identifying similar solutions in order to reuse them, as a result, the efficiency and quality of design suffers. No less relevant is the issue of information security [3].

The significant reduction in the costs of cataloging, checking the completeness and inventory of documentation, as well as an increase in design quality are achieved due to semantic analysis of documentation using a knowledge base that is updated automatically.

Based on the analyzed typical architectures, the architecture of the developed software for automatic cataloging should have a hybrid client-server implementation, including elements of modular and microservice architecture.

The hybrid client-server implementation, including elements of a modular and microservice architecture, must have the following properties:

1. Support for many programming languages.
2. A module is a natural unit of localization of names.

Proceedings of MIP Computing-V 2022: V International Scientific Workshop on Modeling, Information Processing and Computing, January 25, 2022, Krasnoyarsk, Russia
EMAIL: aap@guap.ru (A.A. Dzyubanenko); alexey.rabin@gmail.com (A.V. Rabin)
ORCID: 0000-0001-7242-3131 (A.A. Dzyubanenko); 0000-0001-5641-0410 (A.V. Rabin)

3. Ability to localize the location of the error, which allows, with a good organization of modules, to correct defects in one module, causes errors in another module.
4. Fast recompilation while fixing the error.
5. Ability to reuse modules.
6. Tools are provided to solve each processing task.
7. High resiliency due to redundancy of critical services.

Availability of analytics tools, so it is easy to track dependencies between services [4].

The developed architecture includes elements from the hybrid architecture (Figure 1):

1. Element "Client".
2. Element "Application Server".
3. Element "Data warehouse".
4. Element "Complex of microservices".

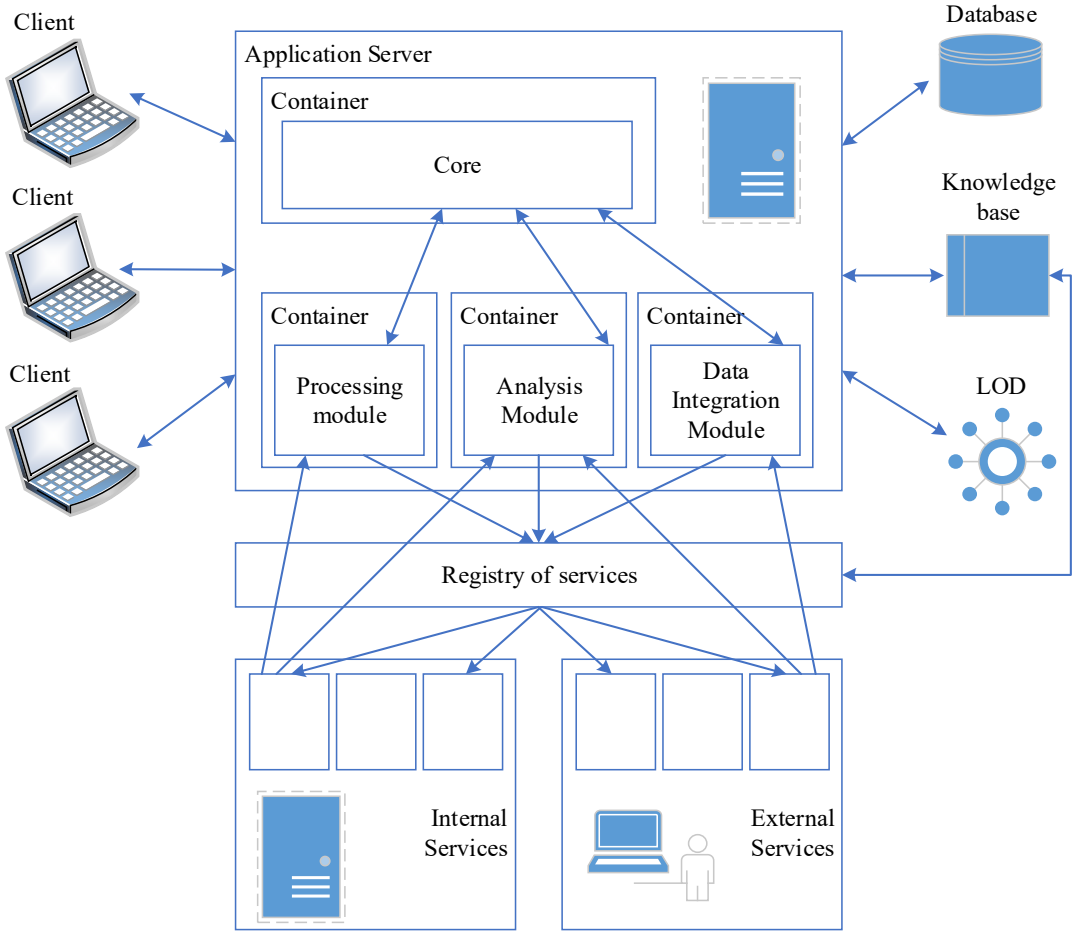

Figure 1: Hybrid software architecture

2. Development of modules for hybrid client-server implementation

2.1. Element "Client"

The client module interacts with personnel and provides data in various formats:

1. Tabular.
2. Graphic.
3. Graph.
4. Text [5].

To solve visualization tasks, built-in components for working with knowledge graphs are used, data is provided in each of the required formats with the ability to navigate the knowledge graph. Also, to

solve analytic tasks, components for working with multidimensional data are used. The graphical interface is also designed for automated structuring of knowledge with the participation of subject matter experts [6].

2.2. Application server element description

The application server is built on a modular basis and includes:

1. The core.
2. Modules of data analysis.
3. Modules of data processing.
4. Modules of data fusion.

2.3. Description core

The flexible core is the central component of the software being developed; it interacts with the rest of the modules and processes user requests from the graphical interface. The kernel, in the process of functioning, processes user requests, interacts with data stores and provides the user with requested samples or calls processing functions from plug-ins [7, 8].

2.4. Application server module description

Modules interact with external and internal services that implement various stages of working with data. The search for the required service is carried out by the service register (a software module that interacts with the ontological description of the service model), which is associated with the service model of the ontology.

2.5. Description of the data processing module

The data processing module includes three stages.

Stage "Data preprocessing".

Preprocessing is aimed at noise reduction in order to improve visual perception for subsequent data processing. For some elements (in particular, for numerical data), the preprocessing stage is skipped (Figure 2)

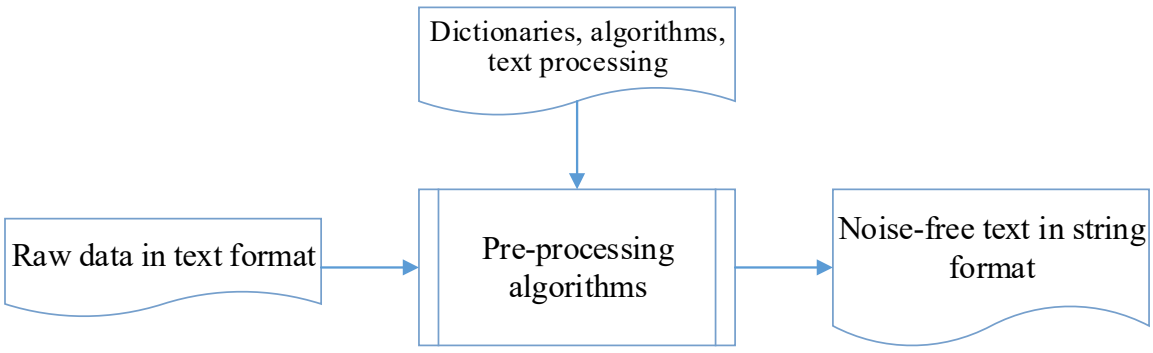

Figure 2: Data preprocessing diagram

Stage "Normalization".

Normalization refers to the process of converting incoming data to a single format. For example, for numerical data, normalization means the unification of the separators of the integer and fractional parts (Figure 3).

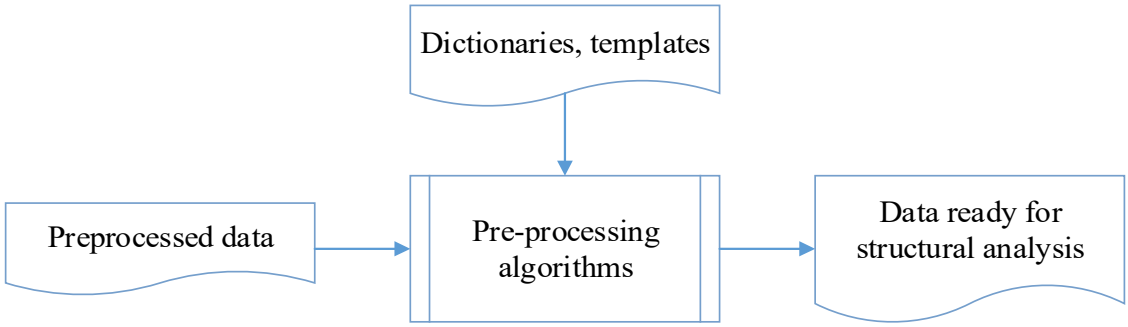

Figure 3: Data normalization scheme

Stage "Processing".

The processing stage is aimed at extracting useful knowledge from the normalized data. The algorithm is selected depending on the type of normalized data (numeric data or text) [9].

The word processing stage includes three schemes, which are designed to solve three types of applied problems:

1. Creation of dictionaries of concepts.
2. Creation of dictionaries of templates.
3. Analysis of the syntax tree [10].

Figure 4 shows the data processing scheme of the proposed algorithm.

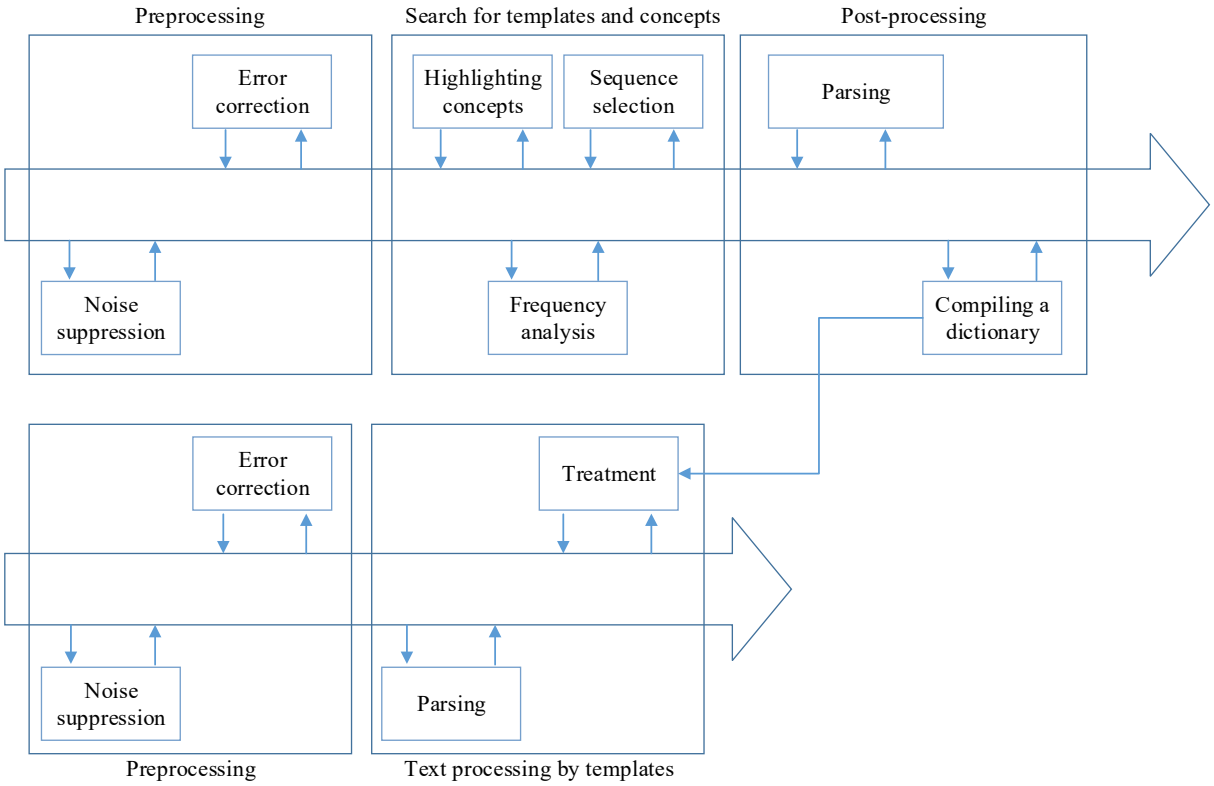

Figure 4: Scheme of data processing of the proposed algorithm

2.6. Description of the data fusion module

This module implements data fusion. Merge refers to the process of data binding in accordance with an ontological model in order to ensure the integrity and consistency of data. The data integration diagram is shown in Figure 5.

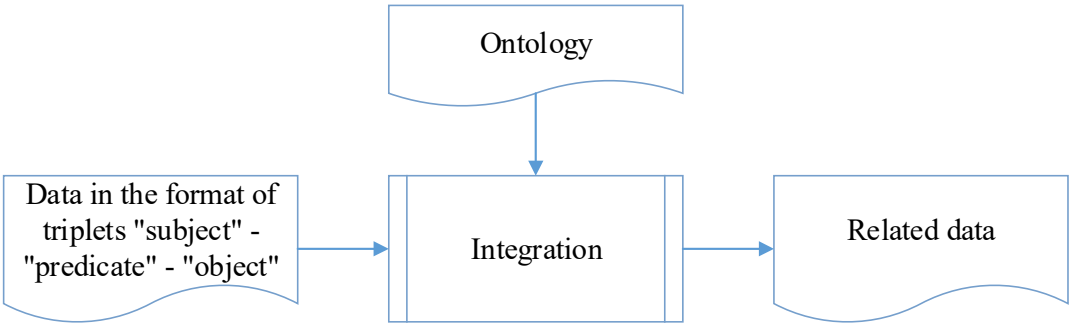

Figure 5: Data integration scheme

3. Development of storage modules, diagnostics and microservice architecture

3.1. Description of the data warehouse element and the data storage object model

Repositories are represented by a database for primary preprocessed data stored for statistical and multivariate analysis, a knowledge base for complex inferences, and an external resource of related data [11].

The module for storing data and knowledge is built on the basis of components for storing semi-structured and semantic data, and includes a set of services imported from cloud services - sources of knowledge for algorithms for processing and structuring data. Data storage components must provide the ability to store and perform efficiently with data arrays accumulated since the beginning of the operation of the IS.

Any database that meets the requirements of performance and stored data volume is suitable as a structured data storage in accordance with the domain model. The connection to the database is made through the API.

The Semantic Data Warehouse is an RDF-storage that allows you to store data in accordance with the ontological model. An important requirement for RDF storage is support for a sufficient number of triplets and high performance. Reference data is represented by third-party ontologies and hierarchically structured reference books [12].

Data enters the database from a source of "raw", raw data or M2M platforms through the data collection module, which is an adapter. The module is designed to monitor data changes, provides services for downloading data from third-party platforms, or independently extracts the incoming data in case of receiving a notification of the change. The data comes in a variety of formats. The format depends on the data collection method used or the type of raw data source. The purpose of the adapter is to transform the received "raw" data into objects of the subject area through a number of syntactic analysis functions using the processing technologies described earlier in the algorithms for graphematic analysis and preprocessing of text resources, which allow to extract semantically significant constructions from semi-structured resources.

To create a hierarchical data storage model in the developed software, it is proposed to use the object model [13].

An object model is a hierarchy or several related hierarchies of classes corresponding to ontologies and phenomena of the domain and describing the interaction between them. The software object model is developed to represent semi-structured data (as a rule, data after OCR in documents have a weak structure), as well as when solving problems involving the processing of large amounts of data. The data stored in the object model and in the ontological model are mutually convertible. The object model is designed to provide fragmentation capabilities for more efficient statistical processing of data.

Let us consider the construction of a hierarchical data storage model for an example, the main subject area of which is high-tech production [14].

The object model includes:

1. Role model.

2. Model of diagnostic tools.
3. Data model.
4. Model of the institution.

3.2. Description of the "role model" module

The role model (Figure 6) describes the roles involved in the processes. The role model includes the staff of the institution, subdivided into laboratory assistants, management personnel, and engineering workers.

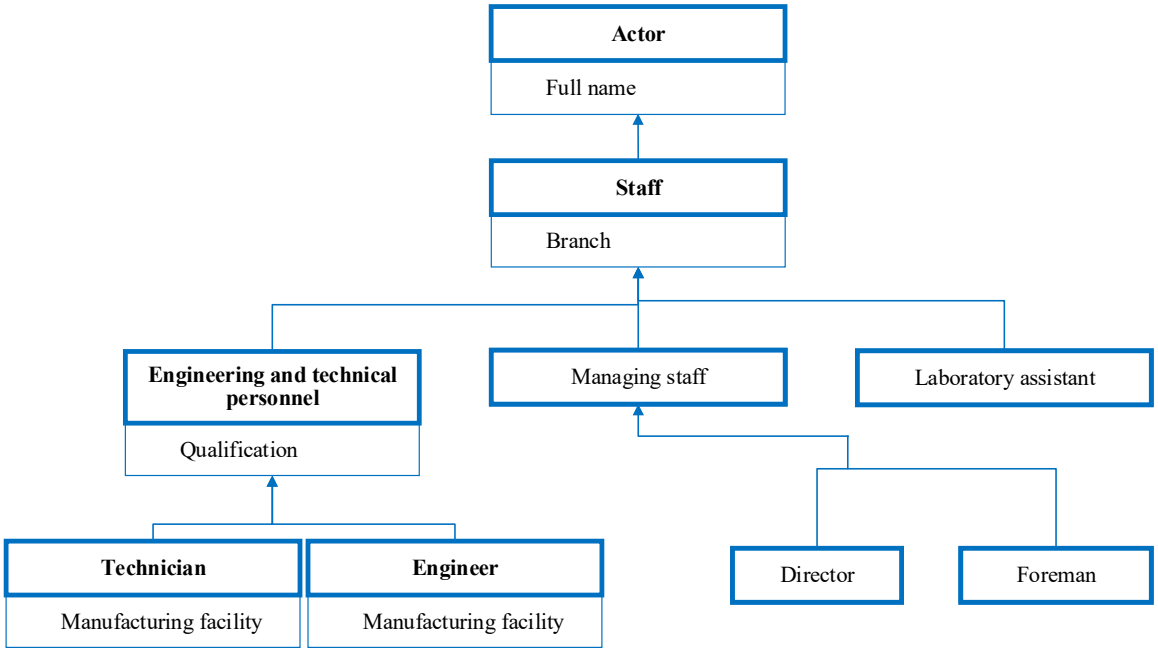

Figure 6: Object model of institutional personnel

Diagnostic model description

The diagnostic model (Figure 7) is a hierarchy of diagnostic tools used to study the state of the equipment. Diagnostics is carried out by an employee of the institution.

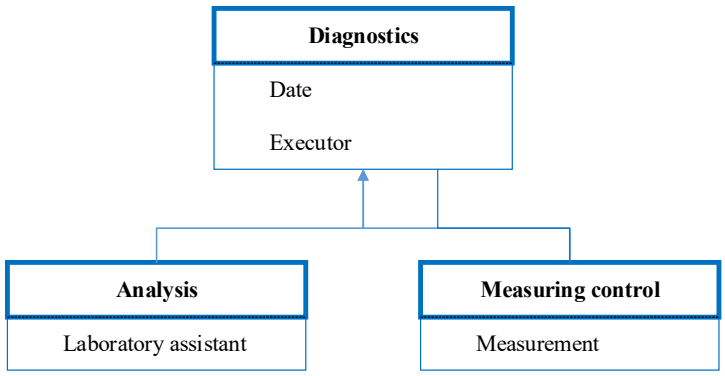

Figure 7: Object model of diagnostic tools

3.3. Description of the data model

Data on the documents collected in the institution's IS are represented by text records and numerical values. Numerical and qualitative data are highlighted. The data type hierarchy includes subjective and objective data. Each type has a qualifying field.

Figure 8 depicts the data type object model.

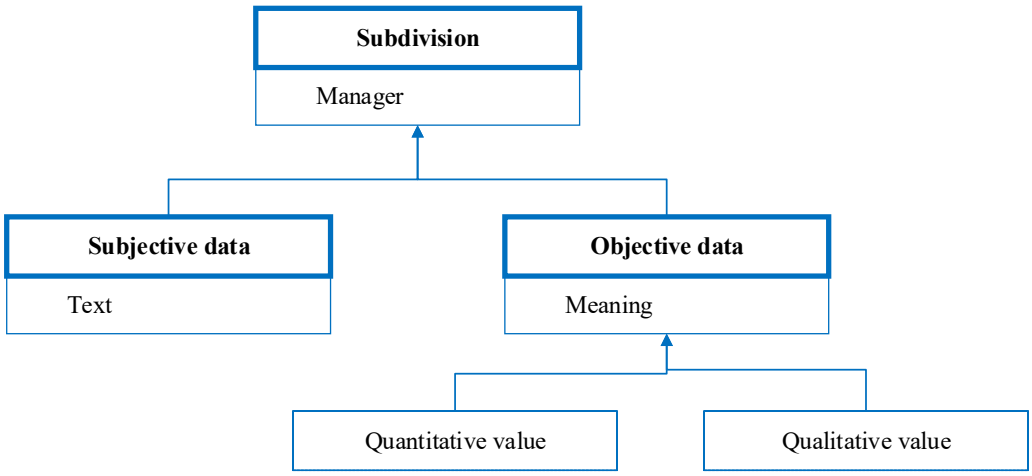

Figure 8: Object model of data types

3.4. Description of the institution model

The company's activities are carried out within the divisions of the institution. In addition to the production process carried out in the workshops, the laboratories carry out analyzes of the manufactured products. The model is shown in Figure 9.

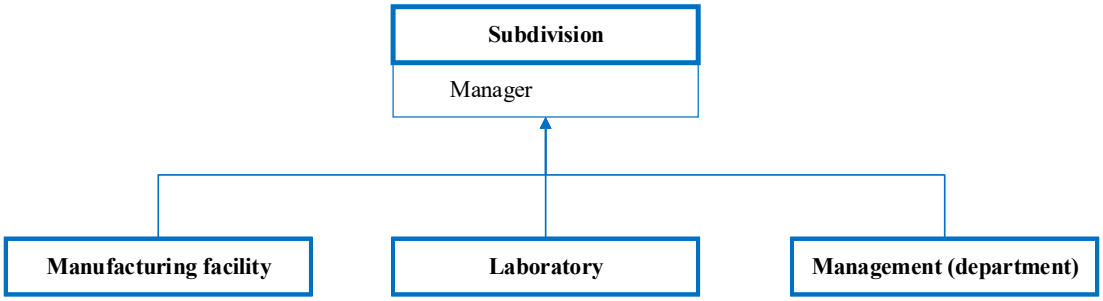

Figure 9: OU object model

3.5. Microservices bundle item description

The source of algorithms in the technological process of processing are services, which are accessed through the corresponding modules. Services are used both in the process of preprocessing data and in solving problems received from users. The results of processing on demand from users are also saved to the knowledge base in order to speed up the execution of similar tasks in the future. Thus, despite the close connection between the modules, their independence is preserved, and the module itself remains operational, provided the kernel and data stores are preserved [15, 16].

4. Conclusions

The modules considered earlier include a set of components intended for processing data within a module and interacting with each other through interfaces. Internal kernel modules interact by calling the API methods of the components, the interaction of the graphical interface with the kernel, and the kernel with the data source is carried out by sending GET RESTAPI requests or through the Web-socket (Figure 10).

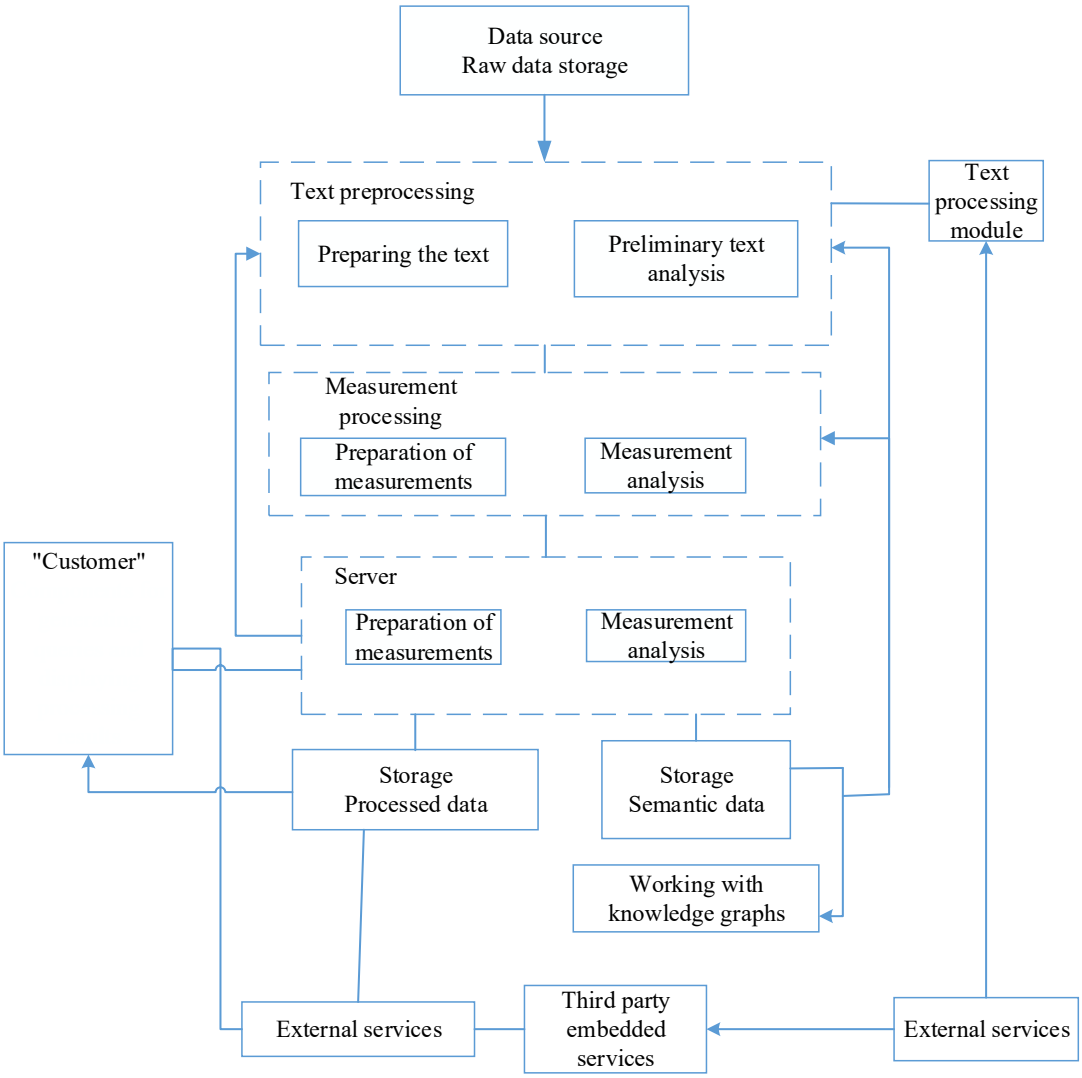

Figure 10: Modules of the developed software

5. Acknowledgements

The paper was prepared with the financial support of the Ministry of science and higher education of the Russian Federation in the course of the applied research «The comprehensive project to create high-tech production of software tools for automatic analysis of documentation on paper and digital media using semantic-cognitive technologies for cataloging poorly structured information» (unique identifier of the project 075-11-2019-055, Decree of the Government of the Russian Federation N 218, 09.04.2010).

6. References

[1] Saurabh Gupta and Anil Kumar Meena, A Practical Implementation of Automatic Document Analysis and Verification using Tesseract, International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS), December 2018. doi:10.1109/CTEMS.2018.8769310.

[2] S. Marinai, Introduction to Document Analysis and Recognition, in: S. Marinai, H. Fujisawa (eds), Machine Learning in Document Analysis and Recognition. Studies in Computational Intelligence, Springer, Berlin, Heidelberg, 2008, vol. 90, pp. 1-20. doi:10.1007/978-3-540-76280-5_1.

[3] R. S. Renu, G. Mocko and A. Koneru, Use of big data and knowledge discovery to create data backbones for decision support systems, Procedia Computer Science, vol. 20, pp. 446-453 (2013). doi:10.1016/j.procs.2013.09.301.
[4] A. V. Rabin and A. A. Petrushevskaya, Development of the formal model of the optical recognition mechanism for tabular documents, IOP Conference Series: Materials Science and Engineering, 2020, 862, 052078. doi:10.1088/1757-899X/862/5/052078.
[5] D. Rajpathak, R. Chougule and P. Bandyopadhyay, A domain-specific decision support system for knowledge discovery using association and text mining, Knowledge and Information Systems, vol. 31, pp. 405-432 (2012). doi:10.1007/s10115-011-0409-1.
[6] Yuan Y. Tang, Chang D. Yan, M. Cheriet and Ching Y. Suen, Automatic Analysis and Understanding of Documents, Handbook of Pattern Recognition and Computer Vision, pp. 625-654 (1993). doi:10.1142/9789814343138_0022.
[7] A. V. Rabin and A. A. Petrushevskaya, Development of the formal model for the presentation of poorly structured and unstructured information, IOP Conference Series: Materials Science and Engineering, 2020, 862, 052076. doi:10.1088/1757-899X/862/5/052076.
[8] A. V. Rabin, A. A. Petrushevskaya and O. V. Sinitsin, Methods and formal models of intelligent analysis of weakly structured data, IOP Conference Series: Materials Science and Engineering, 2020, 734, 012159. doi:10.1088/1757-899X/734/1/012159.
[9] D. Chen D and C. D. Manning, A fast and accurate dependency parser using neural networks, Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, January 2014, pp. 740-750. doi:10.3115/v1/D14-1082.
[10] A. A. Zarubin, A. R. Koval, V. S. Moshkin and A. A. Filippov, Construction of the problem area ontology based on the syntagmatic analysis of external wiki-resources, 2017. URL: http://ceur-ws.org/Vol-1903/paper26.pdf.
[11] A. V. Rabin and A. A. Petrushevskaya, Hybridization of linguistic and statistical methods of text analysis for describing the subject area in the form of a fuzzy ontology, Journal of Physics: Conference Series, 2020, 1679, 4, 042008. doi:10.1088/1742-6596/1679/4/042008.
[12] H. Ltifi, C. Kolski, M. B. Ayed and A. M. Alimi, A human-centred design approach for developing dynamic decision support system based on knowledge discovery in databases, Journal of Decision Systems, Abingdon, Taylor & Francis, 2013, vol. 22, pp. 69-96, ISSN 1246-0125, ZDB-ID 2093544-4.
[13] A. V. Rabin and A. A. Petrushevskaya, Development of the algorithm for graphematic analysis and isolating of semantically significant constructions in poorly structured text, Journal of Physics: Conference Series, 2020, 1679, 4, 042002. doi:10.1088/1742-6596/1679/4/042002.
[14] AstraVer: reliable and secure software, URL: http://astraver.linuxtesting.org/review/.
[15] The Tesseract open source OCR engine, 2018. URL: http://code.google.com/p/tesseract-ocr-13/11/2018.
[16] K. Y. Wong, R. G. Casey and F. M. Wahl, Document analysis system, IBM Journal of Research and Development, 1982, vol. 26, iss. 6, pp. 647-656. doi:10.1147/rd.266.0647.