Methods for identification of objects of development of cross-platform on-board software for communication and navigation satellites

Dmitry Kovalev [1,2], Igor Kovalev [1,2,3,4,5], Nikolay Testoyedov [4,6], Vasily Losev [4] and Mikhail Saramud [3,4]

[1] Krasnoyarsk Science and Technologies City Hall, 61, Uritskogo str., Krasnoyarsk, 660049, Russia
[2] Krasnoyarsk State Agrarian University, 90, Mira pr., Krasnoyarsk, 660049, Russia
[3] Siberian Federal University, 79, Svobodny pr., Krasnoyarsk, 660041, Russia
[4] Reshetnev University, 31, Krasnoyarsky Rabochy Av., Krasnoyarsk, 660037, Russia
[5] China Aviation Industry General Aircraft Zhejiang Institute Co., Ltd, China
[6] JSC "Academician M F Reshetnev Information satellite systems", 52, Lenin street, Zheleznogorsk, Krasnoyarsk region, 662972, Russia

Abstract

The article discusses the architectural basis of the cross-platform onboard software for navigation and communication satellites. The functional characteristics of the designed software components are the queues used, the ports, the supported hardware, and the low-level inter-hardware communication protocols. The verbal description of the selected area of knowledge and the formalization of relations between individual entities allows you to move from the architectural basis, limited by physical parameters and the environment for the functioning of software tools, to a descriptive model, which served as the basis for a relational database. An identification method is proposed for unambiguously identifying a specific component of the on-board software with the possibility of performing further actions on it, for example, archiving, retrieving, duplicating, etc. An onboard software component is viewed as a complex entity that contains not only program code, but also a number of attributes that are used during operation. The structural diagram of a relational database is presented. This structure unites and links both information about software components and the electronic document management tool as a whole.

Keywords

On-board software, identification method, development, cross-platform software, navigation and communication satellite

1. Introduction

The purpose of the developed identification method is to unambiguously determine a specific component of the on-board software (OBS) for the possibility of further actions on it (archiving, retrieval, duplication, etc. [1-3]). The OBS component is considered as a complex object containing not only program code, but also a number of attributes that are used in the process of working on it (Figure 1).

Proceedings of MIP Computing-V 2022: V International Scientific Workshop on Modeling, Information Processing and Computing, January 25, 2022, Krasnoyarsk, Russia
EMAIL: kovalev.dw@gmail.ru (Dmitry Kovalev); kovalev.fsu@mail.ru (Igor Kovalev); office@iss-reshetnev.ru (Nikolay Testoyedov); basilos@mail.ru (Vasily Losev); msaramud@gmail.com (Mikhail Saramud)
ORCID: 0000-0003-2128-6661 (Igor Kovalev); 0000-0002-1996-2889 (Vasily Losev); 0000-0003-0344-9842 (Mikhail Saramud)

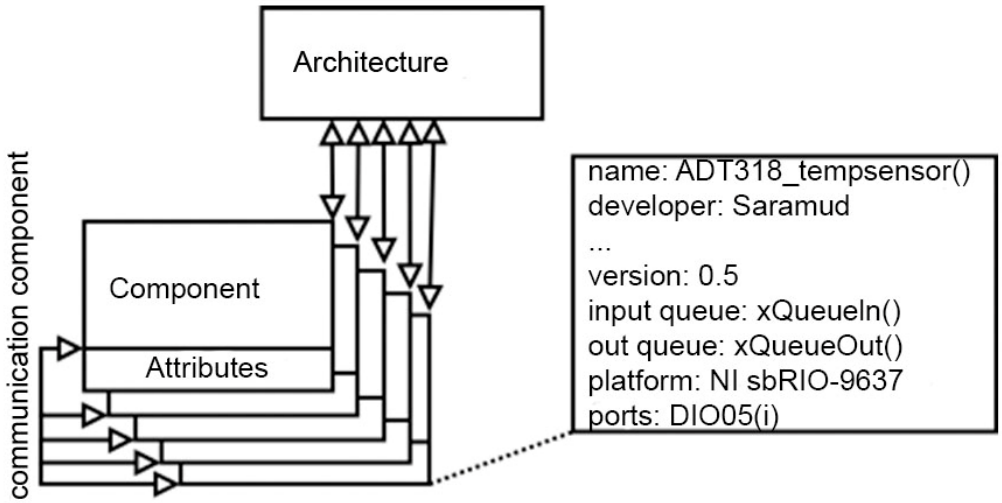

Figure 1: Development objects

There are several options for identifying the program code that can be used in the process of searching for development objects for the cross-platform OBS of navigation communication satellites:

- information extraction (IE);
- parsing of program code;
- the method of labeling;
- building the atomic structure of the project.

2. Method of information extraction

Information extraction (IE) is an applied direction in the field of computational linguistics, which studies the tasks of extracting certain information from natural language (NL) texts - terms and their relations, events and named entities (names, personalities, geographical names), etc. [4-6].

The solution of such problems requires the recognition of certain linguistic structures (for example, noun phrases) in the text, which is usually implemented on the basis of partial parsing, without full parsing of sentences of the text. The construction of specific AI applications is most often performed using specialized instrumental systems [7], the most famous of which is the GATE system [8]. At the same time, a characteristic feature is the use of special formal languages (in the GATE system, the Jape language) to specify information about the composition and grammatical properties of the recognized structures, usually in the form of special templates. With the help of templates, ready-made modules for the analysis of NL-text can be customized to solve the required word processing problem in a specific subject area.

Template languages used in instrumental systems for building NL-applications have a number of problems that complicate their effective use.

When constructing the internal representation of the text, it is first broken up into fragments - words, punctuation marks and whitespace characters, and a morphological analysis of words is performed, the morphological (syntactic) interpretations of which form the edges of the graph. In this case, in the general case, a pair of adjacent vertices of the graph are connected by several edges due to morphological homonymy (for example, the nominative and accusative cases of nouns). For any structure highlighted in the text, its syntactic interpretation is represented in the text graph by an additional edge [9].

The considered graphical representation of the text is convenient for taking into account various combinations of morphological interpretations of words included in the text and allows, when imposing templates, to uniformly process both word elements and auxiliary templates. Selection of a structure

using the LSPL (Lexico-Syntactic Pattern Language) pattern is a search in a text graph, which includes three main stages.

Step 1. The set of edges from which the search can be started is determined; for this, the indexes of words, patterns and parts of speech are used (these indexes are built simultaneously with the construction of the text graph).

Step 2. Paths starting from found edges and corresponding sequences of template elements are defined in the graph. The search for these paths represents a depth-first traversal of the graph with a backward step, while at each step all admissible path extensions are considered that correspond to the current element of the template and the restriction on morphological characteristics specified for it. The conditions for matching elements are checked immediately after specifying morphological characteristics, which makes it possible to reduce the set of considered paths in the graph.

Step 3. The found paths are grouped together to form overlay variants that are added to the graph as new edges. The grouping of paths is carried out for more efficient operation of the method.

For the analysis, a software component was used, developed by the authors within the framework of the project "Technology for organizing the life cycle of cross-platform software for onboard equipment for unmanned aerial vehicles" (Reg. No. R&D AAAA-A17-117041910158-8).

The very structure of the handler scanner might look like this:

```
#include <fstream>
#include <iostream>
#include <string>
#include "FlexLexer.h"
std::string tmpstr;
int main(int argc, char* argv[])
{
    if (argc < 2) {
    std::cout << "USE: ccflexscan <source file>"  << std::endl;
          return 0;
                      }
    std::ifstream infile(argv[1]);
    FlexLexer* lexer = new yyFlexLexer(&infile);
    int iRes = 0;
    while ((iRes = lexer->yylex()) > 0)
    {
          std::cout << "Read " << lexer->YYText()
                    << " (token value is "
                   << iRes << ") << std::endl;
    }
    return 0;
}
```

When applying it, observing the conditions of the syntax and the rules of the dictionary, at the output we will receive a list of the following content:

```
Read ' RUN ' (token values is 286)
Read ' OPTIMIZATION ' (token values is 283)
```

The multiplicity of syntactic interpretations of one piece of text inherent in natural language inevitably increases the search space and the amount of memory required to store a graph of text. This is especially noticeable when the template contains a large number of elements that do not participate in the matching conditions, and also includes repetitions of elements without their matching.

3. Parsing

Parsing (often called parsing in computer science) is a technology for automatically collecting data based on a given attribute.

The information retrieved by parsers (parsers) is accumulated for the purpose of selection according to certain criteria, analysis of dynamics, backup storage in case of loss of access to primary sources, and is also used for cataloging.

The separation of the necessary information occurs through the syntactic and lexical analysis of its content. This process is called parsing. Parsing is part of many information processing processes, from translating text to translating high-level language code into machine code.

Parsers are most actively used in the processing of Internet pages by search engine algorithms. But parsing is also used for solving less global problems, for example - parsing is used by programs to automatically check the uniqueness of text information, quickly comparing the content of hundreds of web pages with the proposed text. Parsing can also be used to automatically search for information, for example, to fill in the fields of characteristics of hardware, sensors, etc.

As an example of a tool for analyzing program code, we present the following program:

```
#include <stdio.h>
#include <string>
extern FILE *yyin, *yyout;   // referenced from flex-generated scanner
extern int yylineno;
extern std::string tmpstr;
int yylex();
int main(int argc, char* argv[])
{
    yyin = fopen("test.txt","r");  //yyout will ptr to stdout
    int iRes = 0;
    while ((iRes = yylex()) > 0)
    { //read something useful from file
          printf ("Read ‘%s’ (token values is %d) at
                      line %d\n",tmpstr.c_str(), iRes, yylineno);
    }
    printf ("Scanner return %d code\n",iRes);
    return 0;
}
```

Its functions are similar to the program described in section 2 above, but the execution speed is lower, but this method has also proved to be an effective analysis tool.

As an intermediate result, we can conclude that the existing methods of parsing program code do not meet the requirements stated at the beginning - storage and retrieval of metadata, the possibility of hierarchical linking of components and memory of the final structure of projects as one of the elements of flexible search. To meet all the requirements for storing and retrieving useful information about the project, it is proposed to create a new identification method based on deterministic metadata.

4. Method of functional identification of cross-platform OBS components

At the stage of forming the basic description of the area of knowledge - the software components of the cross-platform OBS of communication and navigation satellites, the requirements for the description of the principles of functioning and the limitations of the physical environment of functioning were determined and formalized, such as: ensuring the operation of the power system, voltage level, board, etc.

In general, this will allow us to carry out effective identification of software components, since the presented restrictions and requirements can act as metadata that uniquely describe a specific component.

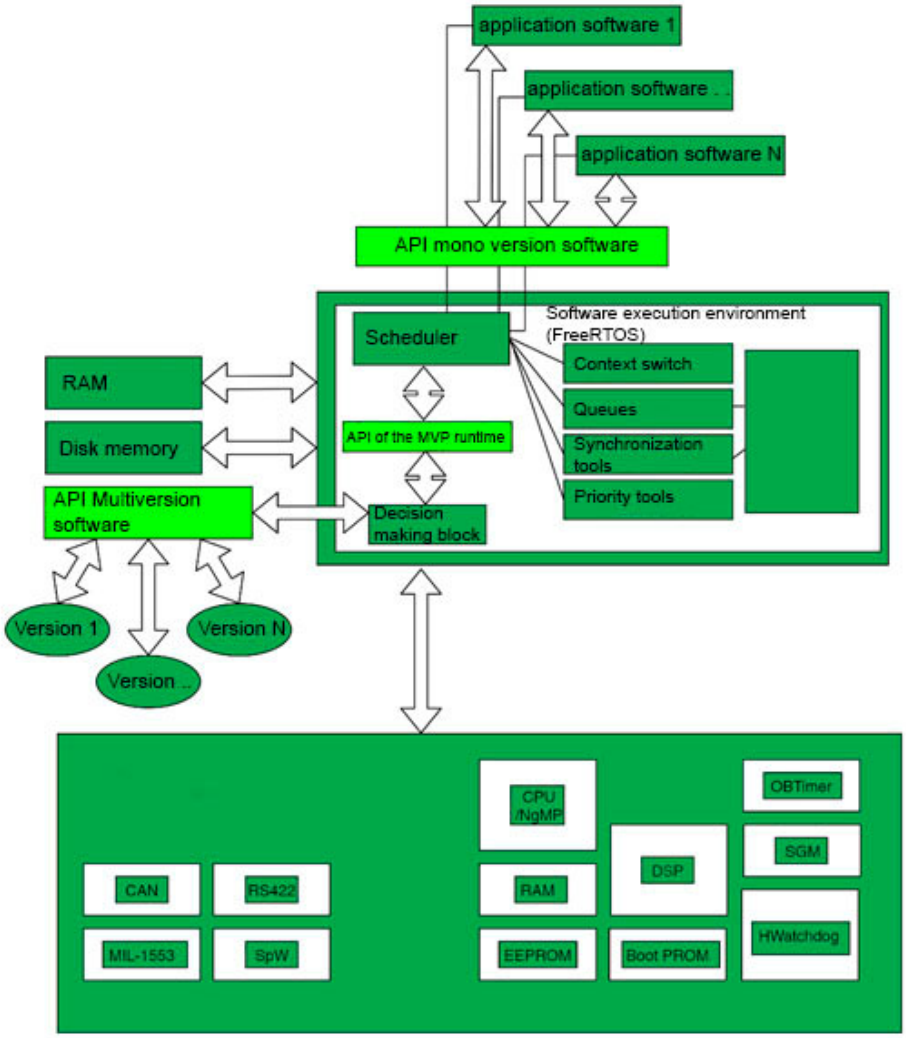

Figure 2: Architectural basis of cross-platform OBS of navigation communication satellites

The architectural basis of the cross-platform OBS of navigation communication satellites (Figure 2) is presented in [10-12]. The most important functional characteristics of the designed software components, as can be seen from the diagram presented in [13, 14], are: used queues, ports, supported hardware and low-level inter-hardware communication protocols.

The resulting verbal description of the selected area of knowledge and the formalization of relations between individual entities made it possible to move from an architectural basis, limited by physical parameters and the environment for the functioning of software, to a descriptive model that served as the basis for a relational database.

4.1. Database schema

Figure 3 shows a structural diagram of a relational database. This structure unites and links both information about software components and the electronic document management tool as a whole.

The projected components are assigned a set of information attributes, which will later be used to identify the desired component according to the functional requirements for it.

By functional requirements, we mean such parameters as the input / output queues used for data exchange, the used hardware platform and its hardware ports, etc.

Considering the area of design of the components of the OBS of communication satellites, let us turn to the developed architectural basis. Based on it, let us highlight the most important characteristics of the component for us, which will be the attributes of the database used to identify the components of

the cross-platform OBS being developed. These informational attributes are typical for both applied and executive software. because from the point of view of the programmer there is no fundamental difference in the development of a software component. When designing a component, it should be noted that a separate software function in the future can be used both as an independent component and as one of the versions of the multiversion pool. Technically, there is no difference between the development of one of the versions of the multiversion pool and a separate software component, since they will be executed in the same way as a FreeRTOS stream, and a separately developed module - the decision block - is responsible for the functionality of the multiversion execution.

Let's consider an example of a software module for processing readings of a digital temperature sensor.

Design Reference (Component): This module must be connected to an Analog Devices ADT7318 Digital Temperature Sensor or equivalent. The sensor output is connected to DIO pin 5 on the board. The module must poll the sensor every 2000ms, calculate the temperature value in Celsius and Fahrenheit. The temperature value in Celsius is placed in the xQueueCTemp () queue. The temperature value in Fahrenheit is placed in the xQueueFTemp () queue.

The task itself is an element of electronic document management and is stored in the database in the form of a document signed with an electronic signature. Inside the document, there is also meta-information for the unambiguous definition of the parameters of the developed software block and is used to create the "skeleton" of the project. The metadata related to the design assignment for the temperature sensor readings processing module are given below (Table 1). Supported hardware - ADT7318 digital temperature sensor or equivalent. The range of input values is presented in Table 5

Table 1
Example of an assignment for the development of a software component

Project	No. 223
Module name	ADT7318_tempsensor
Responsible for development	DK
Responsible for testing	GN
Target platform	NI sbrio 9.6
Check date	31.12.2018

Table 2
Required values of functional characteristics

	CPU timems	RAM kb	MTTF sec	R/W op
Required value	600	200	6000	20

Table 3
Output queues

Queue name	Data type	Description of data
xQueueCTemp()	double	Temperature value received from the sensor in degrees Celsius
xQueueFTemp()	double	Temperature reading from sensor in Fahrenheit

Table 4
Physical ports

Port	Protocol	I/O	Description of data
DIO5	i2c	entrance	Digital temperature sensor output

Table 5
Input range

Port / Queue	from	before	step
DIO5	-271 C	600 C	0.01

The metadata themselves uniquely identify the block when searching for information through the database, and also serve as keys for searching for block variants close to the criteria.

Storage order for later retrieval. To break many-to-many data, intermediate tables are used, in which two foreign keys of the tables being linked are used as the primary key of the table itself.

Table 6 is an example of a table for breaking a many-to-many relationship.

Table 6
Example of the relationship

queue_id	modules_id	i/o
101	1	1
101	2	1
101	3	0

Since the primary keys are the id of the queue and the id of the module, a many-to-many relationship is formed when the tables are directly linked. To break this connection, an intermediate table is used, which also serves to facilitate the process of finding the necessary information. As a request, let's try to find out which modules work with the queue at index 101.

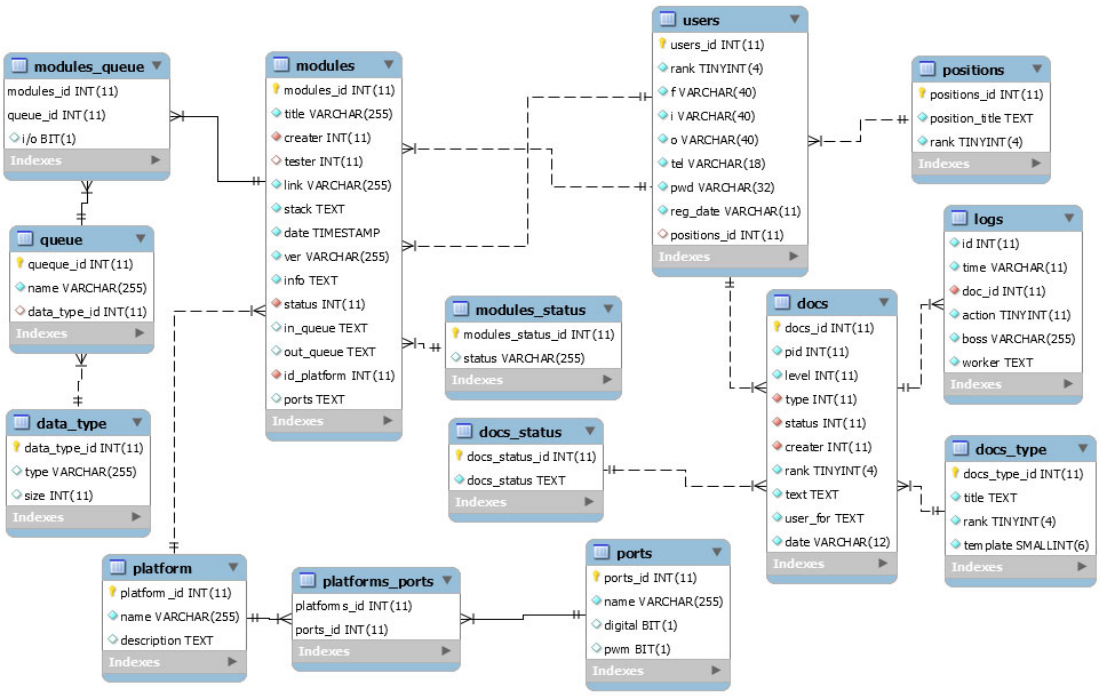

Figure 3: Visual representation of the relationship between the entities of the developed relational database

Figure 3 shows the following tables:

Table 1 . The **modules** entity serves as a repository of the main attributes of the components being developed and the **platform, modules_status, users** "dictionaries" are connected to it.

Table 2 . The docs entity serves as a repository of data about the elements of the electronic document management system, the “dictionaries” **docs_type, docs_status, users** are connected to it.

Table 3 . The **modules_queue** entity serves to separate many-to-many relationships between queue and modules.

Table 4 . The **queue** entity is used to describe the queues for input / output of data from individual blocks, and its structure is organized according to the requirements of the selected RTOS. The **data_type** entity serves as a "dictionary".

Table 5 . The **users** entity serves as a repository of data about users and their access rights to data, and also creates a logical hierarchy of subordination and formation of orders. There is a “dictionary” of **positions**.

Table 6 . The **platform** entity serves as a repository of the characteristics of the used hardware platforms, their number of physical outputs, structure, etc.

There is a **platform_ports** link table, which in turn has a relationship to the **ports** entity.

Table 7. The **logs** entity serves as a "log" of all events occurring with copies of electronic documents in an electronic document management tool. Associated through a foreign key with the **docs** entity.

To ensure data integrity and efficient functional identification, one-to-many and one-to-one relationship types are used with the ability to cascade deletion. The constructed structure of the relational database provides connectivity, reliability of storage and retrieval, as well as structural verification of the entire architecture of the software being developed.

The relationships between the tables in Figure 3 are characterized by the following:

- The modules entity has the modules_id main key and the creater, status, idplatform foreign keys.
- The platform entity has a platforms_id master key.
- The modules_status entity has the modules_status_id main key.
- The users entity has the users_id master key.
- The docs entity has a main key docs_id and foreign keys type, status, creater.
- The docs_type entity has a docs_type_id master key.
- The docs_status entity has a docs_status_id master key.
- The modules_queue entity has an associated master key from the modules_id and queue_ id fields.
- The positions entity has a master key, positions_id.
- The platform_ports entity has an associated master key from the platforms_id and ports_ id fields.
- The ports entity has the ports_id master key.
- The logs entity has a master key logs _id and a foreign key docs_id.

Also, in the environment of electronic document management, database objects can be created, retrieved and modified by means of the PHP language. Below is an example of a script that generates a report on the stage of module readiness.

```
<?
if (!is_numeric($rank) || $rank <= 0) {header("Location:
./check.php");
die;
}

$mods_status=["<span style='color:gray;'>Indeveloping</span>",
              "<span style='color:pink;'> Readytotest</span>",
              "<span style='color:blue;'> Ontesting </span>",
                "<spanstyle='color:red;'> Submitted for revision
              </span>",
              "<span style='color:green;'> Ready</span>"];

if (!$_GET[id]){
    echo "<table border=0 width=100% class='docs-table'>";
```

```
     echo                                                    "<tr><th
width=30px>Ver</th><th>Modulename</th><th>Status</th><th>Author</th>
<th>Data</th></tr><tr><td colspan=5><hr></td></tr>";

     $q_mod_all=mysqli_query($link, "SELECT * FROM `modules` ORDER
BY date DESC");
     for ($c=0; $c<mysqli_num_rows($q_mod_all); $c++)
     {
          $new='normal';
          $mods = mysqli_fetch_array($q_mod_all);
               if                    ($mods['creater']==$user['id'])
$author=$user['f']."                    ".substr($user['i'],0,2).".
".substr($user['o'],0,2).". ";
               else {

     $creater=mysqli_fetch_array(mysqli_query($link,         "SELECT
`f`,`i`,`o` FROM `users` WHERE id='{$mods['creater']}' LIMIT 1"));

                    $author=$creater['f']."
".substr($creater['i'],0,2).". ".substr($creater['o'],0,2).". ";
               }

               echo         "<tr         style='font-weight:{$new}'
onMouseMove=(this.style.background='#EBF4FD')
onMouseOut=(this.style.background='white')>
                                    <td>".$mods[ver]."</td>
                                    <td
onClick=(document.location.href='?act=modules&id=".$mods['id']."')>"
.$mods[title]." ::</td>

     <td>".$mods_status[$mods['status']]."</td>
                                    <td>".$author."</td>

     <td>".date('d.n.Y',$mods['date'])."</td>
                                </tr><tr><td
colspan=5><hr></td></tr>";
     }
     echo "</table>";

     } else {
  //echo          "<div          class=noprint><a          href='#'
class=btnsaleonClick='window.print() ;'>Print</a></div>";

     $mod=mysqli_fetch_array(mysqli_query($link,   "SELECT   *   FROM
`modules` WHERE id=".intval($_GET[id])."));
     $author=author($mod, intval($_GET[id]));
     $stack_in=explode("=>", $mod[stack])[0];
     $stack_out=explode("=>", $mod[stack])[1];

     echo "<table width=100% cellpadding=6>";

     echo   "<tr><td><h3>{$mod[title]}</h3>{$mod[date]},   {$author},
{$mods_status[$mod[status]]}<br></td></tr>";
     echo "<tr><td><h3> description</h3>{$mod[info]}<br></td></tr>";
     echo "<tr><td><h3>Inputqueues:
  </h3>{$stack_in}<br></td></tr>";
```

```
      echo "<tr><td><h3>Outqueues:
  </h3>{$stack_out}<br></td></tr>";
      if ($mod[status]==4)
      {
            //$test=mysqli_fetch_array(mysqli_query($link, "SELECT *
FROM `docs` WHERE type=pid=".intval($_GET[id])."""));
            echo "<tr><td><h3>Test results:</h3><a href='#'>Test
report</a><br></td></tr>";

            echo "<div class=noprint><a href='#'
class=btnsaleonClick='#'> Archive</a></div>";
      }
      echo "<tr><td><h3> Programcode:</h3><br></td></tr>";
      echo "</table>";
      //echo "<div class=noprint><a
href='?act=modules&id={$mod['id']}' class=btnsale>
Newmodulebasedoncurrent</a></div>";
      //echo $mod[link];
      $file= file_get_contents("./modules/{$mod[link]}.txt");

      echo "<div class=print_noview>";
      echo "<imgsrc='./tmp/1.png' width=45%><imgsrc='./tmp/2.png'
width=45%><br><br>";
      echouser_for_print( $mod['creater'], $user_position, 0);
      echouser_for_print( 3, $user_position, 0);
      echo "</div>";

      echo '<div style="overflow: auto; height: 80%; width: 65%;
padding: 8px; border: 1px solid #ccc;"class=noprint><xmp>';
      echo $file;
      echo '</xmp></div>';

      }
  ?>
```

Thus, the obtained scheme of interactions within a relational database will allow efficiently performing functional identification both in the form of SQL queries and third-party tools, if necessary.

4.2. Identification procedure

In the process of developing the components of the cross-platform onboard software, their functional attributes will be entered into the database. According to these attributes, it becomes possible in the future to perform functional identification of the required module. Let's consider an example of functional identification of a component based on specified parameters.

These parameters are input / output queues (their IDs), platform selected, author, etc. In general, the set of attributes should be a strict filter list that filters out mismatched base lines. If a strict search did not give the necessary results, then a flexible search is used, which makes it possible to include in the search process the possibility of identifying components that do not strictly correspond to all specified search restrictions, but are the most suitable from those available in the database, for example, they correspond to 4 out of 5 parameters.

If the search is completed successfully, then a number of actions can be performed on the found component: print it, send it, open it for editing (the old version of the OBS component is saved, a separate copy is created), create a document in the electronic document management tool.

As an example of retrieving information from a database, let's take the creation of a query in the SQL language - finding out all the components working with the queue under index 4. Figure 4 is shown below, showing both the structure of the query and the result of the query. As a result of executing the

requests, we see that the queue of interest to us is used by the 19 and 20 components as the output data queue, and the module 25 as the input data queue.

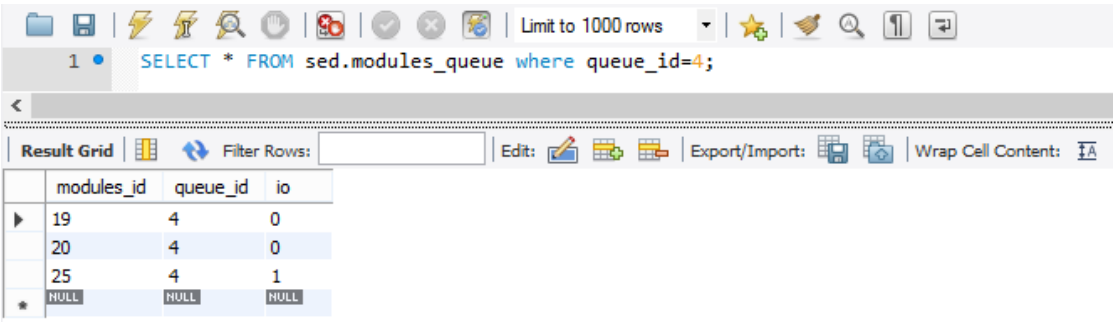

Figure 4: The structure of the request and its result

The developed method makes certain requirements for the process of entering information into the database, which are dictated by the need for clear structuring and building links between metadata for flexible search. Figure 5 depicts these relationships as relational database relationship diagrams.

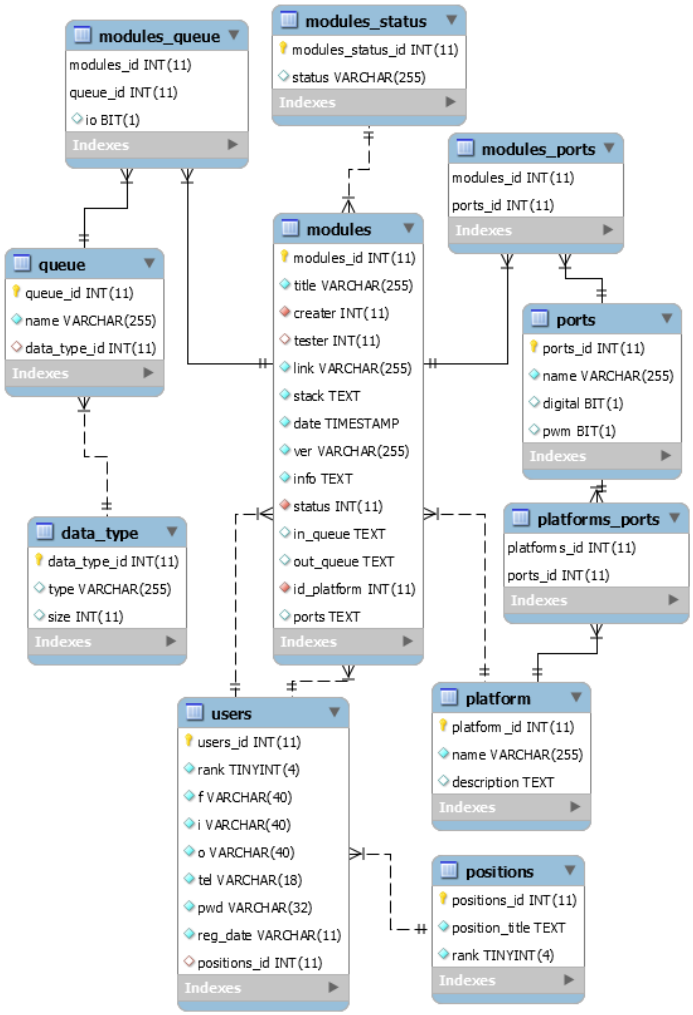

Figure 5: Diagram of relationships for entities responsible for the description of a specific component

4.3. Pre-identification method

Using the previously developed design method for the components of the cross-platform OBS of communication and navigation satellites, which involves the stage of architectural and detailed design, we define the attributes of the components as metadata elements that allow identification, and the set of metadata for each component is complete and sufficient to unambiguously identify an individual component and create a flexible system search by partial match. We will also include in the metadata the information about the component being developed that is necessary in the development process to specify the binding of components to specific equipment (names of supported ports, types of ports, names of queues). The developed descriptive information model of the knowledge area related to the elements of the components of the cross-platform OBS of communication and navigation satellites will be stored in a relational database.

The above procedure for development, formalized recording and functional identification, implemented by means of manipulating an electronic relational database, allows you to effectively organize the creation, storage and retrieval of software components. In general, the process of working with a separate component can be divided into stages for which you can create rules for filling in the database:

- creation of a new component (all numbers of administrative documents become elements of metadata associated with the components). Rule - it is necessary to fill in a complete list of metadata related to administrative document flow, physical parameters of the selected development platform and specific parameters, data when forming a task for creating components;
- search for a component similar to the one under development. Rule - if the search for strict compliance did not give any results, launch a flexible search for partial compliance, carried out by means of the main and foreign keys of the base;
- editing / creating the required component, observing the decomposition rules for elements sent to the database for storage (information about the developer also becomes a metadata element). Rule - one component can have several developers, it is necessary to save intermediate versions;
- component testing, test results also become metadata elements belonging to a specific component. Rule - it is necessary to fill in all fields in a structured manner that are responsible for certain characteristics of the testing process;
- sending the document to the archive (the electronic version of the archive document is tied to the component). Rule - after sending to the archive, it is impossible to change the metadata or to carry out a cascade deletion by the component number;
- component search (performed by searching for a set of metadata associated with a specific component or by its id, if known). Rule - when searching, use standard templates written in the development environment;
- further work with the found component (after sending the component to the archive, there is only one editing option - creating a new instance, which can be linked to the original block through metadata). Rule - when creating a child block based on an old instance, the new component is assigned its own index and metadata space.

The result of going through all the stages and the execution of the rules will be a built structure of a relational database that has the ability to fully describe all the necessary metadata and has the correct links between them. Figure 6, Figure 7, Figure 8 show an example of the description of the modules entity, created using the presented pre-identification method.

Figure 8 shows the types of data stored in attributes, as well as the parameters for the uniqueness of these attributes.

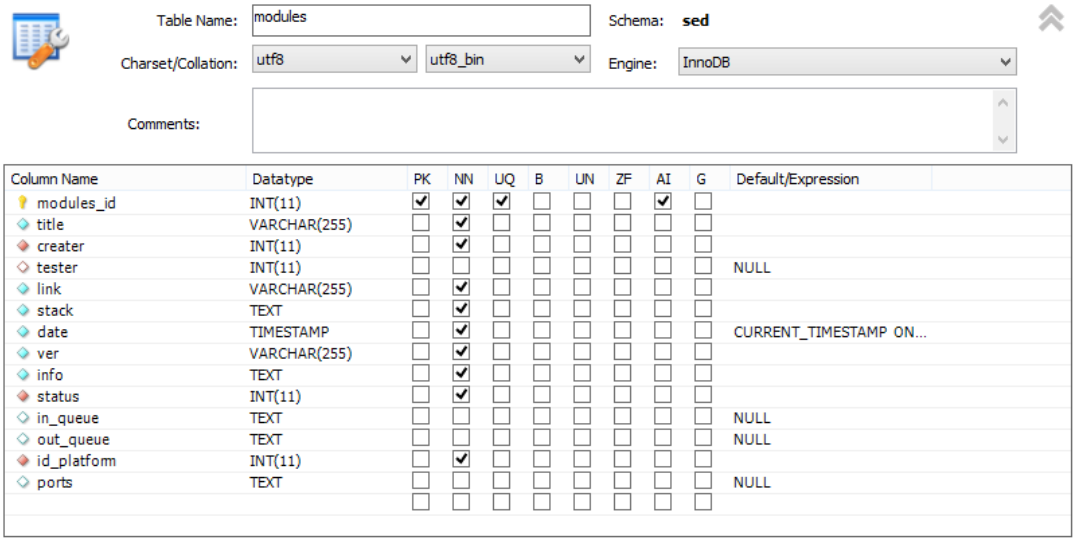

Figure 6: Data types and parameters of attributes of the modules entity

Figure 7 shows all the indices belonging to the modules entity.

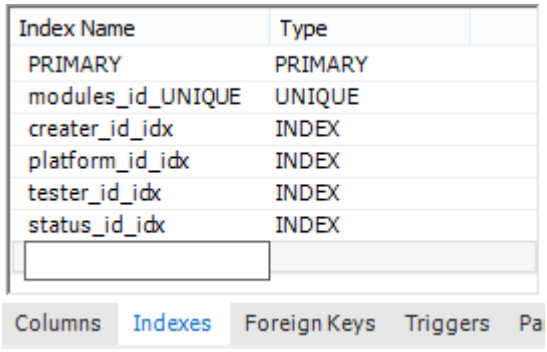

Figure 7: Indices of the modules entity

Figure 8 shows all the foreign keys working in the modules entity and their corresponding tables.

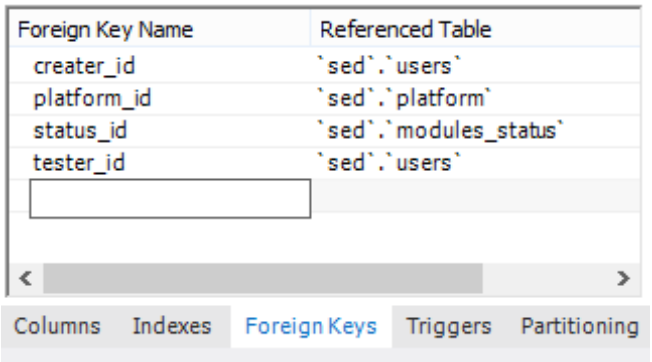

Figure 8: Foreign keys of the modules entity

5. Conclusion

Thus, the developed identification methods provide the ability to identify previously developed components in order to reuse them. They also allow verification of components in the process of project development for the correspondence of the attributes of the components to the architecture of the project.

6. Acknowledgements

The research is carried out with the support of the Krasnoyarsk Regional Fund for Support of Scientific and Scientific and Technical Activities, within the framework of the project "Control of the flight trajectory of aircraft in the extreme conditions of the Arctic and the Far North" in accordance with application 2021110907918. The authors are grateful to the China Aviation Industry General Aircraft Zhejiang Institute Co., Ltd for supporting this study and creating favorable conditions for research.

7. References

[1] I. V. Kovalev, M. V. Saramud, V. V. Losev, A. A. Koltashev, Method and tools for verification of cross-platform onboard software, Modern innovations, systems and technologies 1 (2) (2021) 26-37.

[2] I. V. Kovalev, V. V. Losev, M. V. Saramud et al., To the question for formation of a block-modular structure of the control system for unmanned aerial vehicles, Modern innovations, systems and technologies 1 (3) (2021) 54-71. DOI 10.47813 / 2782-2818-2021-1-3-48-64.

[3] I. N. Kartsan, Construction of ground control points for spacecraft using optimization-simulation model, Modern innovations, systems and technologies 1 (2) (2021) 69-76.

[4] H. Ji, Information Extraction. In: LIU L., ÖZSU M.T. (eds) Encyclopedia of Database Systems. Springer, Boston, MA, 2009. https://doi.org/10.1007/978-0-387-39940-9_204

[5] H. Ji, D. Westbrook, R. Grishman, Using semantic relations to refine coreference decisions. Proc. Conf. Human Language Tech. and Empirical Methods in Natural Language Proc. (2005) 17-24.

[6] P.S. Jacobs, G.R. Krupka, L.F. Rau, Lexico-semantic pattern matching as a companion to parsing in text understanding. URL: http://pdf.aminer.org/000/511/641/lexico_semantic_pattern_matching_as_a_companion_to_parsing_in.pdf (дата обращения: 10.01.2015).

[7] H. Cunningham, D. Maynard, V. Tablan, JAPE: a Java Annotation Patterns Engine. (Second Edition). Technical report CS--00--10, University of Sheffield, Department of Computer Science (2000).

[8] H. Cunningham, K. Bontcheva, Computational Language Systems, Architectures, Encyclopedia of Language and Linguistics. 2nd Edition. Elsevier (2005) 733-752.

[9] D. N. Kolisnichenko, PHP and MySQL, Web Application Development, BHV-Petersburg 6 (2017).

[10] I. V. Kovalev, M. V. Saramud, V. V. Losev Simulation environment for the choice of the decision making algorithm in multi-version real-time system, Information and Software Technology 120(106245) (2020). DOI:10.1016/j.infsof.2019.106245.

[11] V. A. Obukhov, V. A. Kirillov, V. G. Petukhov et al., Problematic issue of spacecraft development for contactless removal of space debris by ion beam, Acta Astronautica 181 (2021) 569-578.

[12] T. V. Radionov, E. B. Molodan, A. S. Mikhalev et al., Automated system design for experimental research of the technological process parameters of spacecraft waveguide paths induction soldering, Iop Conference Series: Materials Science and Engineering 734 (1) (2020) 012137.

[13] M. V. Saramud, I.V. Kovalev, V.V. Losev, P.A. Kusnetsov, Software interfaces and decision block for the execution environment of multi-version software in real-time operating systems, International journal on information technologies and security 1(10) (2018) 25-34.

[14] V. V. Losev, I. V. Kovalev, M.V. Saramud et al., Methodological approaches to increase the fault tolerance of software system in multiversion environments, Journal of Physics: Conference Series 1679 (5) (2020) 052088.