DOI 10.52209/2706-977X_2025_3_31 **IRSTI 55.01.11** **UDC 621.9:658.512**

Development of Software for Automated Selection of Machining Parameters

Nurzhanova O.A.[1], Berg A.S.[*1], Berg A.A.[1], Bakenov A.A.[2], Semerenko I.A.[1]
[1]Abylkas Saginov Karaganda Technical University, Karaganda, Kazakhstan
[2]«Maker» LLP, Karaganda, Kazakhstan
*corresponding author

Abstract. This paper describes in detail the research methodology, including the use of a local SQLite database to store information about drawings, materials, processing methods, and formulas. The need for dynamic generation of input fields, real-time calculations, and display of drawing thumbnails is emphasized. Particular attention is paid to the fault tolerance of the system, including exception handling (division by zero, missing variables, incorrect data entry) and ensuring stable operation without abnormal termination. The application architecture is organized according to the modular principle (interface, database, computational engine), which simplifies support and extension. The use of the eval() function in a strictly limited context for safe execution of formulas is noted. The application is developed in Python using PyQt5 and can operate autonomously, without access to the Internet or external servers. The presented work demonstrates a solution to the current problem of automation in mechanical engineering, offering a flexible, reliable and convenient tool for engineers and technologists.

Keywords: machining, software, SQL, Python, data

Introduction

Mechanical processing of workpieces is a key link in the production cycle in mechanical engineering [1]. It is at this stage that the geometric parameters and physical properties of parts are formed, ensuring their compliance with design specifications, tolerances and technical regulations [2]. Given the active introduction of digital technologies in recent years, the task of moving from manual and semi-automatic selection of processing modes to their full automation has become increasingly relevant [3]. This is especially important in the manufacture of products of complex configuration, where even a minimal error can cause defects, a decrease in the operational reliability of units or non-compliance with the requirements [4].

In the conditions of classical production, the process engineer is forced to spend a significant amount of time selecting parameters, based on regulatory reference books, design documentation and his own experience [5]. Such an approach is not only labor-intensive, but also prone to human error, especially when it is necessary to promptly recalculate parameters for several process scenarios. For example, changing just one initial parameter, such as the workpiece material, may require recalculating dozens of dependent quantities [6], which entails multiple revisions of all related calculations and reference data [7].

The use of software solutions in which formulas and input data are stored centrally and automatically updated allows for significantly faster calculations and increased accuracy [8]. This approach is of particular importance when working with new or non-standard design developments, when there are no ready-made process charts and the calculation of modes must be carried out “from scratch”.

An analysis of the current situation at machine-building enterprises allows us to identify a number of typical problems [9–11]:

- use of outdated tables of design parameters that do not take into account new materials and technological processes;
- lack of a centralized repository of formulas and reference data;
- high dependence on the level of qualification of a particular specialist;
- insufficient variability in the choice of parameters, especially when it is necessary to model several designs;
- low visibility of calculation results and their visual presentation.

The problem of integrating heterogeneous types of data deserves special attention: digital models, graphic representations of drawings, reference information, user variables and calculation formulas [12]. When processed manually, this information is usually distributed among disparate sources - Excel files, paper albums, oral instructions or the accumulated experience of individual employees.

Modern production conditions require software systems to be not only accurate in calculations and correct in operation, but also easy to use. Since the end users of such systems are often operators and technologists without deep knowledge in the field of programming, an intuitive interface, visual transparency, and minimization of actions required to obtain a result come to the fore [13].

Currently, the market offers various CAD (computer-aided design) and CAM (Computer-Aided Manufacturing) solutions that include modules for calculating processing parameters [14]. The most popular are Siemens NX, Autodesk Fusion 360, SolidCAM, Mastercam, SprutCAM, etc. These software products are primarily aimed at high-tech enterprises with a developed IT infrastructure, extensive equipment libraries, and a high degree of integration with production management systems (ERP, MES, PLM) [15, 16]. However, their functionality does not always fully meet the specific tasks solved within the framework of this study.

Firstly, most of the listed systems require significant financial investments in licensing, setup and personnel training. This limits their use in small enterprises or individual production units. For example, an annual license for Fusion 360 costs hundreds of dollars per user, and the implementation of Siemens NX requires the involvement of qualified specialists and lengthy setup [17].

Secondly, complex engineering calculations, such as selection of cutting modes taking into account the features of the part installation and the number of technological passes, are either absent in these systems or are performed separately by the technologist. In most cases, CAM solutions are focused on constructing tool paths, simulating the processing process and generating control programs for CNC, but not on deep local analysis of calculated data adapted to specific production conditions [18].

Thirdly, the flexibility of editing built-in reference books, adding custom formulas, introducing new variables and adjusting algorithms is often limited. Changing even one calculation dependency, such as feed rate, often requires working with closed source code, specialized macros or complex XML templates, which hinders rapid adaptation to changing production tasks [19].

As an alternative to large complex systems, enterprises often use internal developments in the form of Excel files with formulas, macros and connected databases [20].

The analysis shows that there is a significant gap between industrial CAM systems and such home-made solutions [21, 22]: at one extreme are expensive, powerful, but overloaded and inflexible systems, at the other – simple, but unsafe and unstructured tables. It is in this niche that autonomous, customizable and easy-to-use applications are in demand, which can be installed on local workstations without the need for expensive licensing, server capacity and complex IT support.

Among the few suitable solutions, we can mention open engineering platforms such as FreeCAD with support for Python scripts, as well as Node-RED or Qt-based projects focused on visual programming [23]. However, they usually do not contain ready-made calculation algorithms or structured reference data, playing the role of designers for creating applications rather than ready-made technological tools.

There are also attempts by individual organizations to develop their own utilities based on Access, Delphi or Java [24–26]. Such programs are most often developed within the enterprise, have weak documentation, are not distributed beyond its borders and often do not meet modern interface and scalability standards.

Thus, the aim of this work is to create software for automated selection of mechanical processing parameters, providing the presence of a centralized data storage, the possibility of multi-user work, flexible updating of algorithms and protection from incorrect input.

The system was developed taking into account a set of technological processes for processing parts, including the selection and calculation of parameters that affect accuracy. These parameters include cutting depth, feed rate, spindle speed and other characteristics that determine the tool operating modes and surface quality.

All the indicated values are calculated taking into account a number of factors:

- configuration of the part (the presence of complex internal contours or sharp transitions);
- workpiece material (steel, aluminum, copper, etc.);
- processing method (turning, milling, grinding, etc.);
- the supports and basing elements used (for example, methods for fixing a part during multi-axis processing);
- number of technological passes (rough, finishing);
- reference standards and calculation formulas in force in the organization.

As a result, the designed system must ensure operation with a wide range of input parameters and be able to adapt to various technological scenarios. The totality of these requirements determines the complexity of the subject area and the impossibility of its effective coverage without automated means.

1. Research Methodology

The system being developed provides for loading data from a local SQLite database, which stores information about drawings, materials used, processing methods, support types, calculation formulas, and reference variables. The relationships between these elements are shown in Figure 1.

The user should be able to select from drop-down lists:

- the required drawing (including a sketch of the part);
- support type (basing);
- material;
- processing method (e.g. rough milling, grinding, etc.);
- iteration number (for multi-pass processing).

Depending on the selected configuration, the system must extract the values of the variables from the Variables_dependent_on_material_processing_type table that correspond to the combination "material type - processing method - iteration - tooling" (Figure 1).

Based on the loaded formulas, the system automatically generates input fields for all parameters involved in the calculations. The user has the ability to change the values of variables manually, and the recalculation results are displayed instantly.

All calculations are performed in real time when input data changes. Formulas are stored in the Formulas table as strings and are processed in a secure mode using a limited context (math module, eval function).

The output values of formulas are displayed directly below the corresponding variables as numeric labels, which are updated with each recalculation.

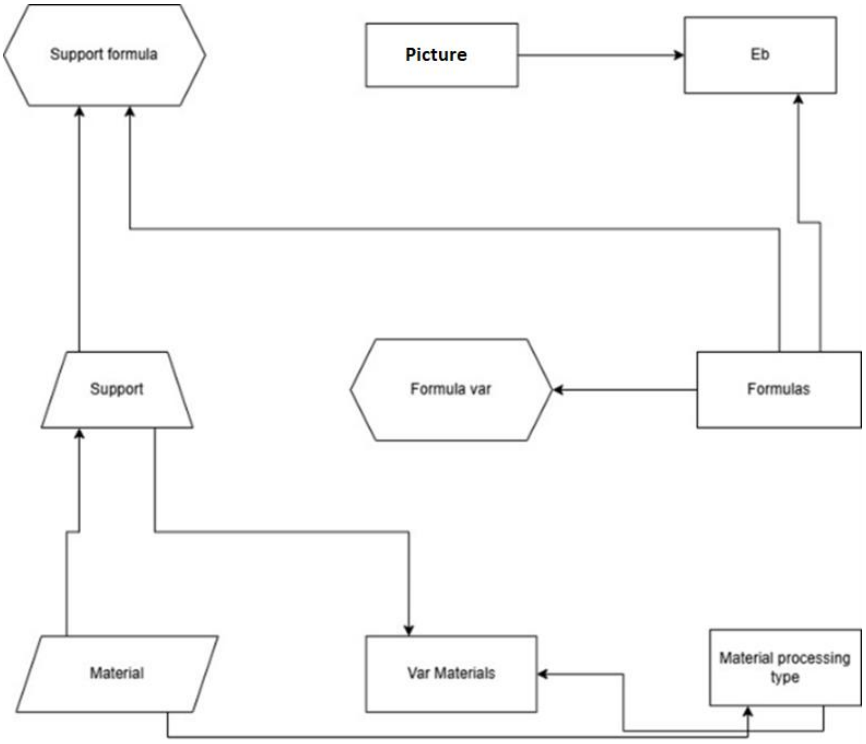

Fig. 1. – Relationships between tables

When a specific drawing is selected, its sketch is loaded, the path to which is specified in the Drawing table.

Formulas may include division operations, logarithm calculations, and other mathematical operations. The program must correctly handle exceptional situations - for example, division by zero, the absence of a variable value, etc. - by returning a neutral value (for example, 0.0) to ensure the robustness of the interface.

The system must be able to dynamically rebuild the set of active formulas with each change in the initial parameters: the selected drawing (Eb-function), the type of support (Support-function) or the processing method (dependent variables). This approach allows the calculation algorithm to be adapted to a specific technological scenario.

Before starting calculations, the application substitutes values from the Reference_data table if the user has not specified them manually.

It is possible to export the database in SQL format or create a backup copy of the file for recovery in case of failure (the function can be implemented by an external script).

The program operates in a completely autonomous mode, without the need for an Internet connection or external servers. All data, including images and formulas, are stored locally.

The system's performance ensures that calculations are completed in less than 0.2 s, even with a large number of formulas and variables.

To protect against input errors, such as entering text instead of a numeric value or incorrect formulas, a system for handling all exceptions has been implemented, preventing abnormal termination of work.

The user interface is designed with an emphasis on simplicity, compactness and scalability. All elements - input fields, lists, labels - automatically adjust to changing window sizes.

The system allows for the addition of new formulas, variables, materials and processing methods solely by updating the database, without interfering with the source code.

The architectural structure of the application is modular and includes three key components: interface (ui), database module (db) and computing module (engine). This simplifies further maintenance and expansion of functionality.

The eval() function works in a closed context that does not have access to the file system, system modules, and environment variables. The software is compatible with major operating systems (Windows/Linux/Mac) with Python and PyQt5 installed. The final build can be performed as executable files .exe or .app.

The application code is provided with comments and logging mechanisms, which simplifies debugging. In case of errors in formulas or at the initialization stage, the program outputs a message to the console or log file.

Only standard and proven libraries are used (for example, math, sqlite3, PyQt5), which minimizes dependence on external components.

Thus, the requirements formed cover all stages of the system life cycle - from data loading to visualization of results and fault tolerance. The following sections provide the rationale for the choice of technologies, the database structure, architectural solutions and a description of the interface.

The development of an application for automated selection of machining parameters was carried out in the Python environment.

2. Results and discussion

2.1 Project structure and code organization

At the logical level, the project structure is divided into three functional blocks:

- Initialization and startup block (main) - is responsible for creating the application object (QApplication), initializing the main window (MachineCalculationApp), setting the size and position on the screen, displaying the user interface and handling the shutdown (app.exec()).;

- User interface block (GUI) — implemented as the main class MachineCalculationApp, inheriting QMainWindow. Within this class, interface elements are created and managed: drop-down lists, text labels, input fields, scroll bars and graphic images.;

- Logic and data processing block — includes functions responsible for receiving information from the database (load_drawings(), load_materials()), forming a set of variables and input fields (create_input_fields()), collecting and calculating data (calculate_all()), safely executing formulas (safe_formula()), and handling events (on_drawing_change(), on_material_change(), etc.).

The class architecture is built with a clear division of function responsibilities, which facilitates the search, modification, and reuse of individual modules. The data exchange diagram between modules is shown in Figure 2.

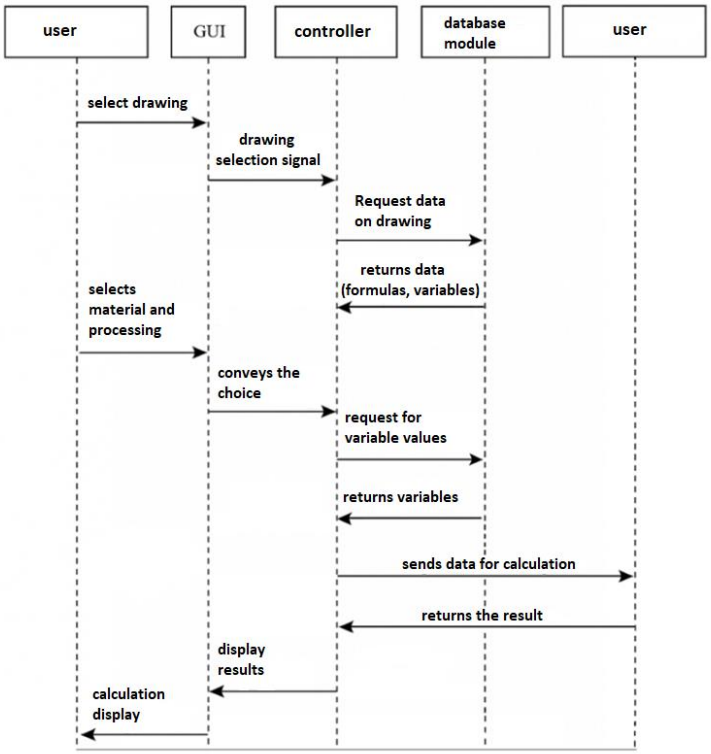

Fig. 2. – Sequence diagram: performing calculation of processing parameters

The project organization follows the principles of modularity: all code is grouped by purpose, structured and provided with comments. The use of a single centralized controller is justified for the current scale of development, and the presence of functions with a clearly defined purpose ensures predictability of the program's operation.

2.2 Implementation of the user interface

The main window of the application is formed on the basis of the QMainWindow class with a central QWidget widget, divided into two zones using a horizontal QHBoxLayout container:

Left panel– a control area that includes drop-down lists for selecting processing parameters (drawing, support, material, method, iteration) and a drawing preview area.;

Right panel– a scrollable area for displaying variables corresponding to the current configuration, their input values and calculated results.

This layout clearly divides the interface into a block for selecting conditions and a block for displaying calculations, which reduces the cognitive load on the user and increases the logicality of interaction.

Interface elements are created dynamically: after the user selects the material and processing method, the system generates a list of necessary variables and displays lines with their name, input field, and result label. When the configuration is changed, irrelevant elements are deleted.

The program implements a reactive operating principle (Figure 3): any change in input data immediately causes a recalculation of values and an update of the interface, which eliminates the need for additional confirmation buttons and speeds up the interaction process.

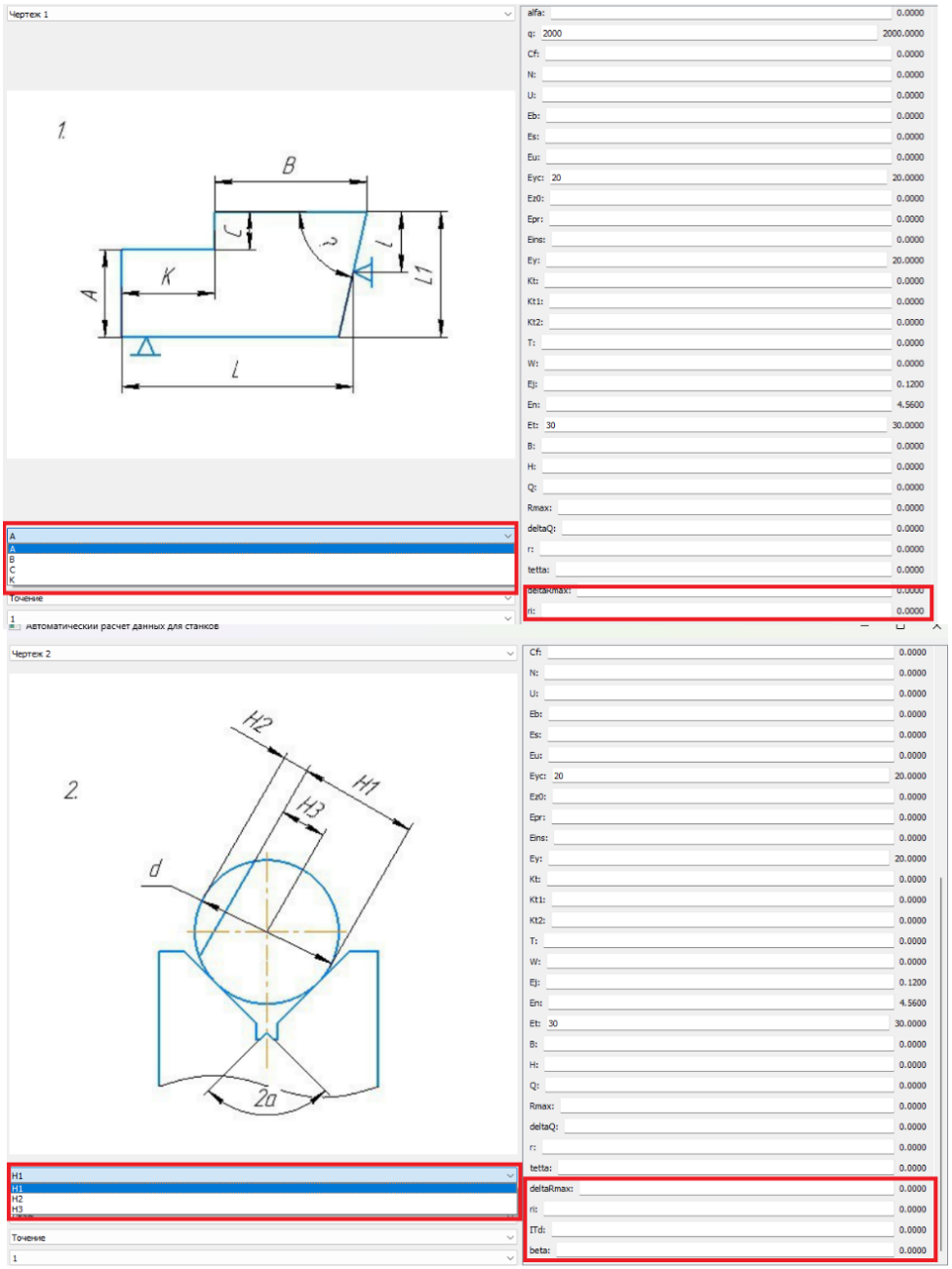

Fig. 3. – Interface of the developed program

Thus, after each change of value or selection, the user instantly sees the recalculated results. This eliminates the need for intermediate buttons and makes interaction with the program faster.

2.3 Processing drawings and loading parameters

Each drawing is stored in the database as a record with a unique identifier, name, and path to the image. The Drawing table has the following structure (Figure 4).

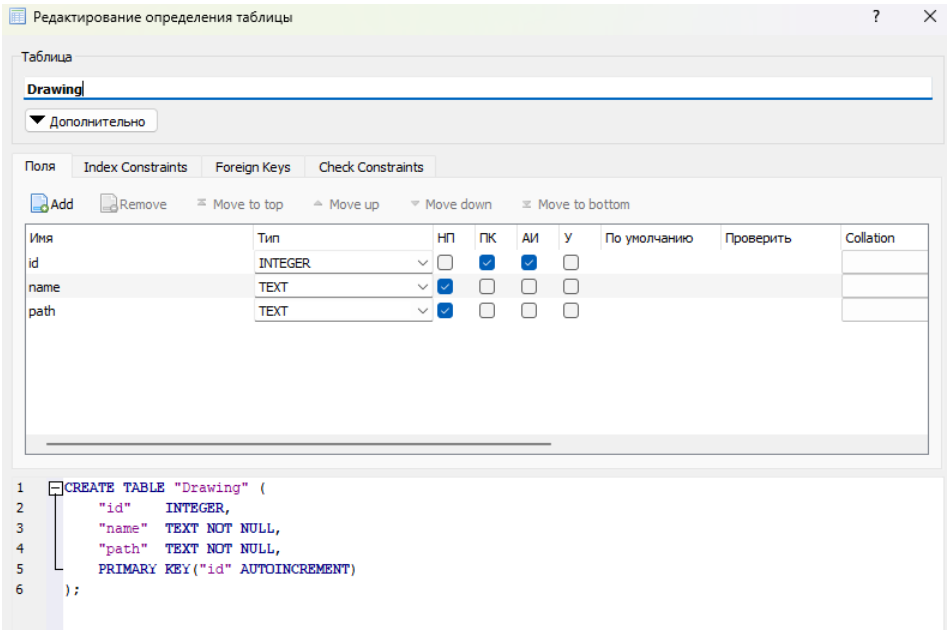

Fig. 4. – Structure of the Drawing table

When the program starts, a query is made to all Drawing records, after which the drawing names are added to the interface drop-down list.

Drawings are linked to the base elements (Eb), for which calculation formulas are defined. The Eb table contains foreign keys linking records to the Drawing and Formulas tables, which ensures logical data integrity (Figure 5).

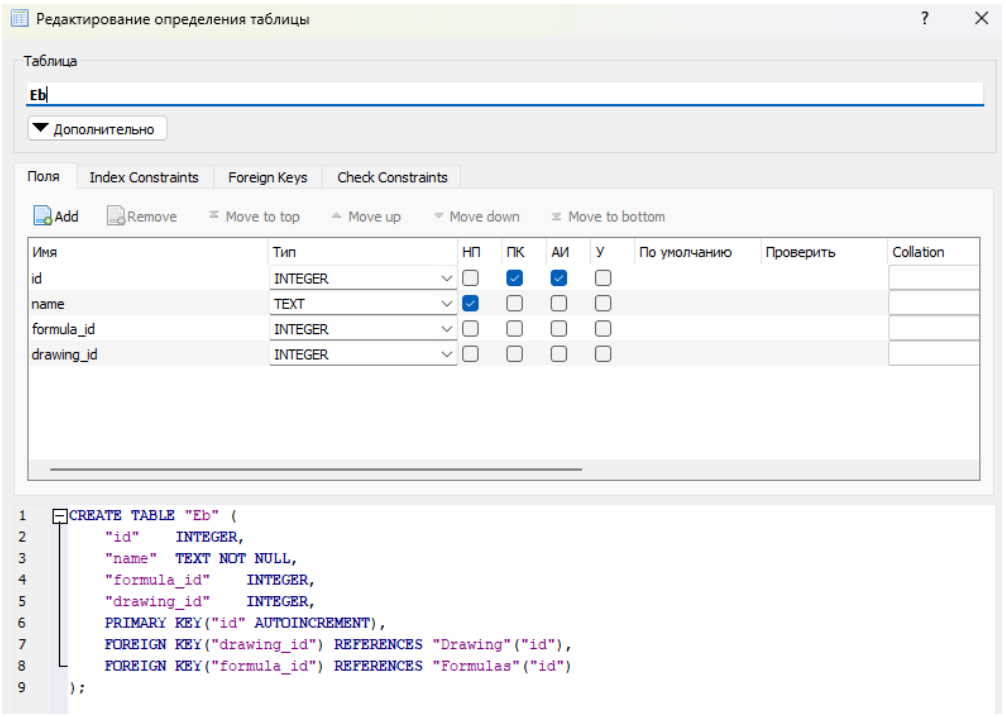

Fig. 5. – Structure of the Eb table

After selecting a drawing, the application generates a selection from the Eb table, defining a set of formulas related to this image. These formulas are included in the active list of calculations. Thus, the interface is adjusted to a specific drawing and uses the corresponding dependencies.

Formulas can be linked not only to the drawing, but also to the selected equipment (Support) and processing conditions. The program generates a final list of calculations, combining mandatory formulas (mandatory_formula = 1), formulas for the drawing and formulas for the support.

After selecting a drawing, the user specifies the material, processing method, support and iteration, which determines the choice of variables. Their values are stored in Variables_dependent_on_material_processing_type and are specified by four keys: method, iteration, material, tooling.

2.4 Implementation of business logic: formulas and calculations

In the project under development, formulas are stored in the Formulas table as string expressions. Each entry in this table contains:

– uniqueid;
– output variable name (target_value);
– the expression text in Python syntax (formula);
– flag of obligationmandatory_formula, which determines whether the formula should always be applied;
– binding to the support (Support_formula_id) if necessary.

Example of a formula line (Figure 6).

```
target_value = 'Rz', formula = 'd["f"] * 15 + math.sqrt(d["Vc"])'
```

Fig.6. – Example of a formula line

This means that the parameter Rz is calculated based on the current values of the variables f and Vc stored in the dictionary d. When substituting data, the formula performs a mathematical operation and returns the result, which is displayed in the interface.

Since formulas are loaded from the database in text form, their direct execution may be unsafe. For protection, the wrapper function safe_formula() is used, which ensures execution in a limited context, access to which is only available to the math module and the d dictionary. This allows flexible interpretation of dependencies, while preventing the execution of malicious code or critical commands (Figure 7).

Applies hereeval()in a strictly limited context where only the library is availablemathand a dictionary of variablesd. This approach allows for flexible interpretation of formulas and at the same time protects the system from failures: any error returns the value0.0, and the program continues to work.

The mechanism for selecting formulas for calculation takes into account:

– mandatory dependencies (mandatory_formula = 1);
– formulas associated with the selected drawing (via tableEb);
– formulas associated with the selected support (tableSupport_formula).

The combined formula list is used to build a list of variables (Formula Variables) and interface generation, its code is presented in Figure 7.

```
def update_active_formulas(self):
    """Обновление списка активных формул"""
    formulas = self.mandatory_formulas + self.eb_formulas + self.support_formulas
    self.active_formulas = []

    # Получаем ID формул для получения переменных
    formula_ids = [f[0] for f in formulas]

    # Получаем используемые переменные
    self.cursor.execute(f"""
        SELECT DISTINCT variable_name
        FROM FormulaVariables
        WHERE formula_id IN ({','.join('?'*len(formula_ids))})
    """, formula_ids)

    self.all_variables = [row[0] for row in self.cursor.fetchall()]
    self.create_input_fields()
```

Fig. 7. – Update code for formulas involved in calculations

Thus, the user sees on the screen only those fields that are necessary for calculations in a specific configuration, and the system processes only relevant formulas.

Before performing calculations, the application generates a data dictionary, which contains:

– values entered manually by the user;
– values substituted from the reference bookReference_data;
– values extracted from the tableVariables_dependent_on_material_processing_type.

Next, each formula from the active list is passed to the formula_wrapper wrapper, and the result is placed in data under the name of the target variable.

2.5 Ensuring fault tolerance and graceful shutdown

In real production conditions, it is impossible to completely eliminate operator errors, incomplete data, or disruptions in the sequence of operations. Therefore, the system was designed to handle any such situations without stopping the application.

One of the most common cases is when the user enters data that does not match the expected type - for example, text instead of a number or an empty value. Such situations are handled in the calculate_all() function (an example of implementation is shown in Figure 8).

```
# Инициализация данных: сначала берем из Reference_data, затем переопределяем пользовательскими
for var in self.all_variables:
    try:
        data[var] = float(self.inputs[var].text())
    except (ValueError, KeyError):
        data[var] = 0.0
```

Fig. 8. – Example of code implementation of fault tolerance

In case of incorrect input, the variable is automatically assigned the value 0.0, which prevents a crash. This mechanism is especially important considering that the end users are engineers and technologists who may not be aware of all the internal restrictions on the data format.

Since all calculation dependencies are stored in the database as strings and executed via the eval() mechanism, there is a risk of errors - division by zero, access to missing variables or syntax violations. To minimize these risks, a protective wrapper has been implemented, which in the event of a failure returns a safe value (0.0) and records the error in the console. Thanks to this, the program continues to work, maintaining the stability of the interface.

The project also provides for situations where certain parameters — variables, iterations, or formulas — may be missing from the database. Instead of stopping execution, the application checks for the required information and, if it is missing, simply does not display the corresponding elements. An example of implementation is shown in Figure 9.

```
self.cursor.execute("""
    SELECT iteration FROM Variables_dependent_on_material_processing_type
    WHERE material_processing_type_id = ?
""", (processing_id,))
iterations = self.cursor.fetchall()
if not iterations:
    self.iteration_selection.clear()
    return
```

Fig. 9. – Example of code when the required data is not found

This approach guarantees the correct operation of the system even with a partially filled database, which is especially important at the implementation stage, when not all information can be entered in advance.

Conclusion

Within the framework of this article, a software system was designed and implemented, intended for automated selection of mechanical processing parameters, corresponding to all established requirements. The main task of the development was to create an adaptive, reliable and convenient tool for engineers and technologists, providing autonomous operation and high accuracy of calculations.

The following key results were obtained during the work:

- a system was created that uses a local SQLite database to store all the necessary information - drawings, materials, processing methods, support types, calculation dependencies and reference variables. This made it possible to eliminate the fragmentation of information typical of traditional approaches using Excel tables or paper media, and to provide a single reliable source of data;

- a reactive interface based on PyQt5 has been implemented, forming a set of input fields depending on the current configuration (drawing, material, processing method, iteration). Calculations are performed almost instantly (less than 0.2 s), allowing the user to see the result of changing the input parameters in real time. This significantly speeds up the process of selecting optimal processing modes and increases the productivity of specialists;

- mechanisms for handling exceptional situations (division by zero, missing variables, incorrect input) have been implemented, which prevent abnormal termination of the program. When an error occurs, the system returns a

safe value (0.0) and registers the event in the log, ensuring continuous operation. This approach is especially important for production environments, where software stability is a critical factor;

- the program architecture is built on a modular principle and includes three main areas - the interface (GUI), the database (DB) and the computing engine (Engine). This simplifies maintenance, modernization and adding new functionality. For example, the integration of additional formulas, variables or materials is carried out by changing the contents of the database without editing the source code;

- to interpret formulas stored as strings, the eval() function is used in a strictly limited and secure environment, which allows for flexible specification of mathematical dependencies with complete protection from the execution of unwanted code;

- the software suite functions completely autonomously, does not require a connection to the network or external servers, and all necessary data and graphic files are stored locally. This makes the system especially convenient for production sites with limited access to the Internet.

Thus, the developed software demonstrates an effective solution to the current problem of automating the selection of mechanical processing parameters in mechanical engineering, combining the accuracy of calculations, ease of use and resistance to errors.

References

[1] Wang Z. The milling parameters of mechanical parts are optimized by NC machining technology //Frontiers in mechanical engineering, Volume 10, 2024,https://doi.org/10.3389/fmech.2024.1367009

[2] Nurzhanova O., Zharkevich O., Berg A., Zhukova A., Mussayev M, Buzauova T., Abdugaliyeva G., Shakhatova A Evaluation of the Structural Strength of a Prefabricated Milling Cutter with Replaceable inserts During Machining //Material and Mechanical Engineering Technology, No. 4, 2023. – P.10-17

[3] Klackova I., Ivanova T., Kuric I., Korshunov A., Koretckiy V. Application of progressive technologies based on digitalization in mechanical engineering //MM Science journal, December, 2022 DOI: 10.17973/MMSJ.2022_12_2022140

[4] Berg A., Nurzhanova O., Vytautas T., Vitushenko D. Improvement of Base Sets for Complex Configuration Parts when Assessing their Manufacturability within Industry 4.0 //Material and Mechanical Engineering Technology, 2, 2024. P. 25 - 35 DOI: 10.52209/2706-977X_2024_2_25

[5] Gupta D.P., Gopalakrishnan B., Chaudhari SA, Jalali S. Development of an integrated model for process planning and parameter selection for machining processes //International Journal of Production Research, Vol. 49, No. 21, 1 November 2011, 6301–6319 DOI: 10.1080/00207543.2010.523722

[6] Senapati A.K., Mohanty S. A Review on the Effect of Process Parameters on Different Output Parameters During Machining of Several Materials //International journal of engineering sciences & research technology, 3(3): March, 2014. – P.1 – 11

[7] Kumar A., Gori Y., Dutt N., Singla YK, Maurya A. Advanced Computational Methods in Mechanical and Materials Engineering. – NY: "Taylor & Francis Group", LLC, 2022. – 71 p.

[8] Zhetessova G., Yurchenko V., Nikonova Y., Zharkevich O. The development of the computer-aided design system for production processes of component part machining for single-piece production and repair conditions //Journal of Applied Engineering Science, 17, 4, 651, 2019, P. 599 - 609 DOI: 10.5937/jaes17-21470

[9] Kozlova EP, Kuznetsova SN, Romanovskaya EV, Andryashina NS, Garina EP Automation of technological processes in mechanical engineering //IOP Conf. Series: Materials Science and Engineering 1111 (2021) 012030, doi:10.1088/1757-899X/1111/1/012030

[10] Trokhymchuk I., Svirzhevskyi K., Tkachuk A., Zabolotnyi O., Zablotskyi V. The Structure of Automated Control Systems for Precision Machining of Parts Bearing // Innovations in Mechatronics Engineering II21 June 2022, P. 182–192

[11] Savelyeva N., Nikonova T., Zhetessova G., Irina K., Yurchenko V., Cernašˇejus O., Zharkevich, O., ˙ Dandybaev E., Berg A., Vassenkin S., Baimuldin M. Implementation of Simulation Modeling of Single and High-Volume Machine-Building Productions //Designs, 2024, 8, 24.https://doi.org/10.3390/designs8020024

[12] Zadeh AY, Shahbazy M. A Review into Data Science and Its Approaches in Mechanical Engineering //5th International conference in and engineering Applied research on science, 2, 2020, DOI: 10.48550/arXiv.2012.15358

[13] Xie, K. Research on the Application of Automatic Control System in Mechanical Engineering 3rd International Workshop on Materials Engineering and Computer Sciences (Advances in Computer Science Research, volume 78, 2018, P. 79 - 82

[14] Kamrani A., Abouel Nasr EA Computer-Based Design and Manufacturing Engineering //Design and Rapid Prototyping, 2009 DOI: 10.1007/978-0-387-95863-7_7

[15] Grabowik C., Kalinowski K., Kempa W., Paprocka I. A method of computer aided design with self-generative models in NX Siemens environment //IOP Conference Series Materials Science and Engineering, 2015, 95(1):012123 DOI: 10.1088/1757-899X/95/1/012123

[16] Song P.P., Qi Y.M., Cai D.C. Research and Application of Autodesk Fusion360 in Industrial Design //OP Conf. Series: Materials Science and Engineering 359, 2018 012037 doi:10.1088/1757-899X/359/1/012037

[17] Nikonova T., Zharkevich O., Dandybaev E., Baimuldin M., Daich L., Sichkarenko A., Kotov E. Developing a measuring system for monitoring the thickness of the 6 m wide hdpe/ldpe polymer geomembrane with its continuous flow using automation equipment //Appl. Sci. 2021, 11, 10045. https://doi.org/10.3390/app112110045

[18] Lysek K., Gwiazda A., Herbus K. Application of CAM systems to simulate a milling machine work IOP Conf. Series: Materials Science and Engineering 400, 2018, 042037 doi:10.1088/1757-899X/400/4/042037

[19] Zhang K., Xiao W., Fan X., Zhao G. CAM as a Service with dynamic toolpath generation ability for process optimization in STEP-NC compliant CNC machining Journal of Manufacturing Systems, Volume 80, 2025, P.294-308 https://doi.org/10.1016/j.jmsy.2025.03.004.

[20] Ghosh A.K., Fattahi S., Ura S. Towards Developing Big Data Analytics for Machining Decision Making. //J.Manuf. Mater. Process, 2023, 7, 159.https://doi.org/10.3390/jmmp7050159

[21] Radhakrishnan P., Subramanian S., Raju V. CAD/CAM/CIM. Edition third. – Deli: New Age International (P) Limited, Publishers, 2008. – 690 p.

[22] Sazonova A.O., Drozdov A.A. Classification and place of cam systems in computer-aided design systems //Vestnik Pstu, No. 2, 2014. P.34 – 42.
[23] Machado F., Malpica N., Borromeo S. Parametric CAD modeling for open source scientific hardware: Comparing OpenSCAD and FreeCAD Python scripts //PLOS ONE, December 5, 2019, https://doi.org/10.1371/journal.pone.0225795
[24] Ling-Lin C., Qi C., Jiacheng Z. Application of Delphi Software in the Teaching of Basics of Mechanical Design // LNEE, volume 112, 2011, P. 443–447
[25] Hibino H., Fukuda Y., Yura MA Study on Java Base Simulation System for Manufacturing System Designs //The proceedings of the JSME annual meeting, July 2000, 2000.3, P. 563-564 DOI: 10.1299/jsmemecjo.2000.3.0_563
[26] Kuryło P., Frankovský P., Malinowski M., Maciejewski T., Varga J., Kostka J., Adrian Ł., Szufa S., Rusnáková S. Data Exchange with Support for the Neutral Processing of Formats in Computer-Aided Design/Computer-Aided Manufacturing Systems //Appl. Sci. 2023, 13, 9811.https://doi.org/10.3390/app13179811

Information of authors

Nurzhanova Oxana Amangeldyevna, PhD**,** acting associate professor, Abylkas Saginov Karaganda Technical University
e-mail: nurzhanova_o@mail.ru

Berg Alexandra Sergeevna, PhD, Abylkas Saginov Karaganda Technical University
e-mail: kibeko_1995@mail.ru

Berg Andrey Alekseevich, senior lecturer, Abylkas Saginov Karaganda Technical University
e-mail: 22526633@mail.ru

Bakenov Adilkhan Amangeldievich, m.t.s., design engineer, "Maker" LLP
e-mail: bakenov.work@yandex.kz

Semerenko Ilya Andreevich, master student, Abylkas Saginov Karaganda Technical University
e-mail: i7776455656@gmail.com