

# RESEARCH AND DEVELOPMENT OF A KNOWLEDGE-BASED DESIGN SYSTEM FOR DESIGNING SELECTED ELEMENTS OF MECHATRONIC DEVICES

MILAN MIHOLA <sup>1</sup>, ZDENEK ZEMAN <sup>1</sup>, DAVID FOJTIK <sup>2</sup>

<sup>1</sup> VSB - Technical University of Ostrava, Department of Robotics, Ostrava, Czech Republic

<sup>2</sup> VSB - Technical University of Ostrava, Department of Control Systems and Instrumentation, Ostrava, Czech Republic

DOI: 10.17973/MMSJ.2021\_12\_2021105

e-mail to corresponding author: milan.mihola@vsb.cz

The design of mechatronic devices is a demanding process not only in terms of the time required but also of the demands placed on the knowledge and experience of development workers. The aim of this research and development was to create suitable procedures, algorithms, and databases of 3D models, with the help of which could this process be significantly shortened and simplified. The results of the development are a software tool for the design of electric drive units, procedures for creating 3D models with the possibility of using the SolidWorks software API, methods for automating the creation of assemblies of 3D models and a description of the knowledge database in which various data and algorithms are stored. The benefits of the proposed procedures, the Drive Picker software tool, and the knowledge database, are demonstrated on the design of a robotic arm with 5 degrees of freedom. Despite the complexity of the preparation of documents needed, it turns out that the chosen approach can significantly speed up and simplify the design of mechatronic devices.

## KEYWORDS

Knowledge-based, Parametric modelling, API, SolidWorks, Macro, Mechatronic

## 1 INTRODUCTION

Mechatronic devices, by definition, consists of four main parts: mechanics, electronics, control, and informatics [Isermann 2002]. The developers strive to integrate these four parts in such a way as to achieve a synergistic effect. Thus, the state where the final effect of the components acting together is greater than would be the case of a simple sum of the effects of the individual components [Kyura 1996]. Despite the availability of methodological procedures for the development of mechatronic devices (VDI 2206, Munich Procedural Model, etc.) [VDI 2004, Lindemann 2009], with which it is possible to achieve better results than would be the case with an intuitive development procedure, it is not possible to call it a trivial matter.

One of the ways to significantly accelerate and often simplify the process of developing mechatronic devices is to use available or develop new software tools, which can be used to design individual elements or subsystems of the device, or complete devices. Currently, there are several software tools developed for the design of standardized machine parts, such

as screws, bearings, gears, etc. Whether in the form of stand-alone applications (e.g. MitCalc or MESYS) or as part of one of the CAD systems (e.g. KISSsoft [KISSsoft 2021]). The calculations based on which machine parts are designed and inspected using these software tools are in most cases based on generally known standards and procedures related to the given types of machine parts (e.g. ANSI, DIN, ISO, etc.). Reddy et al. [Reddy 2016] presented a tool for bearing design. Reddy and Rangadu [Reddy 2018a] developed a gears design tool. Both tools were developed for SolidWorks CAD software and use its Application Programming Interface (API). The data needed to design these elements are stored in a knowledge database. From the point of view of development workers, these are handy tools, with the help of which it is possible to shorten the design and inspection time of individual machine parts up to tens of minutes.

In the case of non-standard elements, which are usually part of mechatronic devices, the situation is somewhat more complicated. Some manufacturers (e.g. Maxon Motor AG, Festo, Bosch Rexroth, etc.) provide online software tools, with which it is possible to select suitable elements of the proposed equipment. However, the applicability of these tools is usually limited to the products of the supplier. Subsequent comparison of products of individual suppliers then remains with the developer of the mechatronic system. This can be quite a time-consuming process. There are also software tools whose database contains products from various manufacturers (e.g. VisualSizer). However, there is still room for further improvement and innovation in this area.

Another way to speed up the development process of the proposed devices is to use pre-prepared 3D models. Most CAD systems include databases of models of standardized components or other frequently used structural elements [Chen 2013, Sun 2011]. In the case of non-standardized elements, it is possible to use databases of 3D models of the given manufacturers, or databases specially created for the availability of 3D models from various fields of technics. Even though the availability of these databases significantly speeds up and simplifies the creation of 3D models of the proposed devices, there is still room for further improvement and innovation in this area. A possible way to do so is to use the API of available CAD software. By using this interface, it is possible to create or modify individual 3D models and their subassemblies and total assemblies by using appropriately designed sequences of commands. Farhan et al. [Farhan 2012] introduced an automated approach to assembling modular accessory elements used in manufacturing processes. Lad and Rao [Lad 2014] developed an application for product design and 3D model updating, whose function they presented on the example of a winding device. Reddy et al. [Reddy 2018b] presented the possibility of using a knowledge database and CAD software to design the layout of an industrial battery tray. By a suitable connection of software tools for the design of individual elements of mechatronic devices, knowledge database, and CAD system, it is possible to achieve significant acceleration of the development process and thus shorten the design time of the device. Another benefit can be smaller demands on the knowledge and experience of development workers.

## 2 KNOWLEDGE-BASED CONSTRUCTION SYSTEM TOOLS

The knowledge-based construction system is developed for the design of selected elements of mechatronic devices. From the previous chapter, it is clear that developers have a relatively wide range of tools for designing basic machine elements

(screws, bearings, gears, etc.). Also, there are tools to assist in the selection of electric, pneumatic, or hydraulic drive units. From this point of view, the currently neglected area is compact electric drive units, which integrate an electric motor, a gearbox, a speed sensor and optionally a brake. The output flange of these drive units is usually mounted in the drive unit's body by bearings capable of absorbing the action of not only radial and axial forces, but also tilting moments. This construction brings advantages in terms of the overall dimensions and weight of the drive units. However, the process of their design entails the need to inspect the individual integrated parts in a similar way as in the case of a drive unit consisting of a separate electric motor, gearbox, and other necessary elements. This not only brings increased requirements on the knowledge of the developer, but it is also a time-consuming process.

## 2.1 Knowledge database structure

One of the development goals is to shorten the design time of compact electric power units while reducing the demands placed on developers in terms of the necessary knowledge and experience. When designing this type of units, several technical data and algorithms are used. The basis of the software tool that is to be used for the selection of suitable compact drive units must, therefore, be a database in which everything needed is to be stored. Fig. 1 shows a diagram of a knowledge database, which was created not only to store the data needed for the design of the drive units, but also to contain suitably prepared 3D models of not only drive units, but also other elements used in the construction of mechatronic devices.

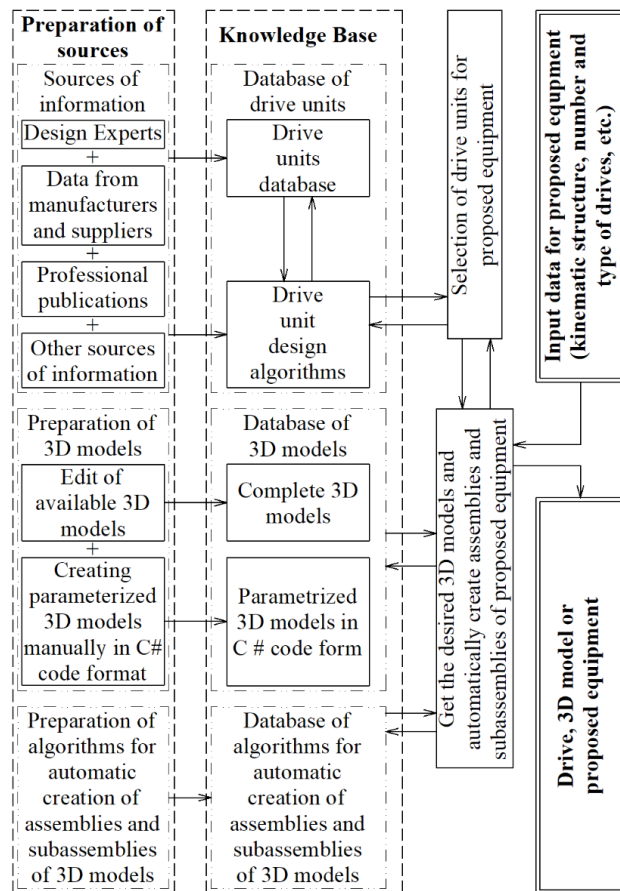


Figure 1. Knowledge database schema

Currently, the knowledge database is built on the MySQL platform and consists of three main parts. The first part contains technical parameters and data related to drive units

and algorithms needed for their design. The source of information for this part of the database is mainly data from suppliers of compact drive units.

The second part of the knowledge database contains 3D models and is divided into two main parts. In the first part, already finished 3D models are saved. In some cases, these are models taken over from the manufacturers of the given elements, which are modified into a form so that they can be used in the simplest possible way in the design of the required equipment. However, these are not so-called living models. Therefore, it is not possible to change their dimensions in a parametric way or to add or remove individual parts of the model in a simple way. However, they are assigned parameters in terms of weight, or moments of inertia and centre of gravity, to best match the real elements. The second main part stores parameterized 3D models in the form of codes written in the C# programming language. The actual generation of 3D models is realized using the CAD interface of the SolidWorks CAD software. The 3D models created in this way not only contain dimensions, with the help of which it is possible to change proportions of a given 3D model, but it is also possible to add or remove individual geometry elements in a simple way. From the developer's point of view, there is no difference between a 3D model created in this way and a model created manually in the graphical environment of the SolidWorks software.

The third part of the knowledge database contains sequences of codes in the C# programming language, based on which it is possible to create assemblies or subassemblies of proposed devices or partial structural nodes from generated or selected 3D models. The actual generation of 3D models is again realized using the API interface of the SolidWorks software. Based on the input data, it is, therefore, possible to use this knowledge database not only for the design of drive units and the preparation of 3D models but also for complete 3D models assemblies, if the knowledge database contains sequences of codes suitable for their creation.

## 2.2 Design of compact electric drive units

The design of compact electric drive units should be carried out according to the instructions provided by their manufacturers. These instructions are prepared in various forms. Here [Harmonic Drive 2021], the design procedure of the drive unit Harmonic Drive AG CanisDrive series, is outlined. The drive is designed from the course of the load by the output torque, the moment of inertia and output speed. The unit designed in this way must then be inspected from the load of the outlet flange by radial and axial forces and tilting moment, resp—their courses. To be able to automate the design process of this type of power unit, it is necessary to create computational algorithms that include all the necessary calculations. Fig. 2 is an example of an algorithm developed just for the CanisDrive series drive units.

To automate the design process, it is necessary to convert parts of the data in the form of graphs related to the individual drive units into a form with which the proposed algorithm could work.

The Drive Picker software tool was created to work with the documents prepared in this way (Fig. 3). Its task, based on the input data of the load of the proposed power unit, is to find in the database the most suitable drive units of various manufacturers. In this process, it is always taken into account that the most suitable drive unit is the weakest in performance from the given series of the given manufacturer. The output is,

therefore, a list of drive units from which the developer can select based on other criteria, such as weight, overall dimensions, or price.

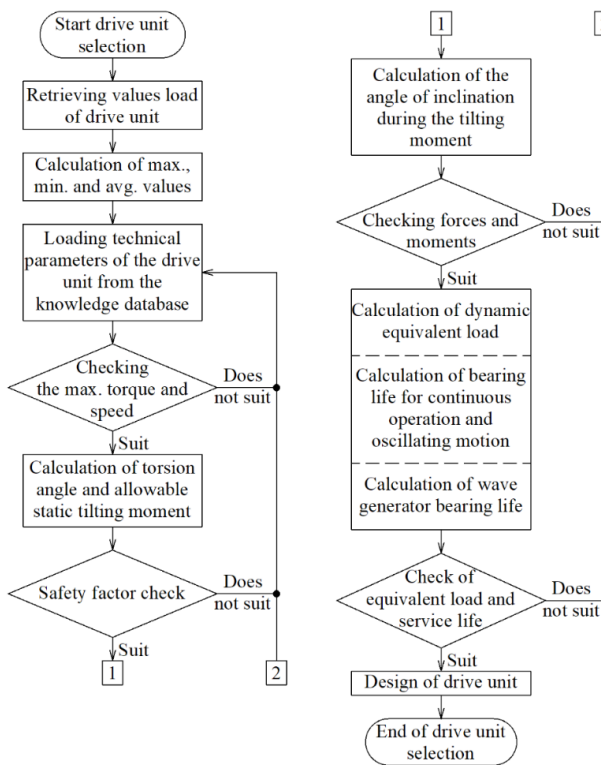


Figure 2. Algorithm for designing the CanisDrive series drive unit [13]

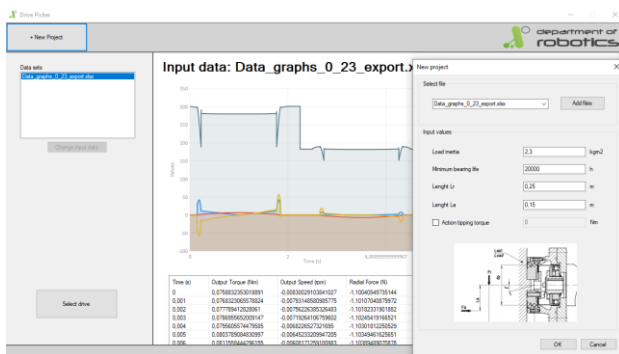


Figure 3. Drive Picker software tool

The course of the load on the proposed drive unit is defined using an input file in which the data are stored in a column format in a precisely defined order (time, output torque, output speed, radial force, axial force). The Drive Picker software tool then converts this data into a graph, allowing for easier visual inspection. In the middle part of the New project window it is possible to enter input values in the form of an external moment of inertia load, minimum bearing life of the drive unit output flange (and thus also mechanical life of the drive unit itself) and distance of radial and axial force from the flange. Instead of these distances, it is also possible to enter the amount of tilting torque acting on the output flange of the drive unit.

Then it is possible to proceed to your design of a suitable drive unit. The software tool gradually reads algorithms and data from the knowledge database, with the help of which the least powerful drive units from the type series of individual manufacturers, which with their parameters meet the requirements imposed on them, are searched for. A possible result of the search for a suitable drive unit is shown in Fig. 4.

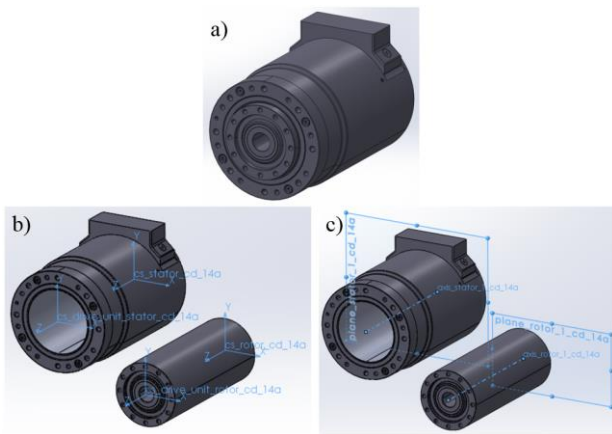
Selected Drive: HarmonicDrive CanisDrive 17A, ratio100			
Technical Data			
Motor winding	AO	Electrical time constant (20 °C)	3,4 ms
Motor feedback system	ROO / MGS / DCO	No load current (20 °C)	0,09 A
Ratio	100	No load current constant	8,5 *10 <sup>-3</sup> A/rpm
Max. output torque	70 Nm	Max. motor speed	7300 rpm
Max. output speed	73 rpm	Rated motor speed	3500 rpm
Max. current	2,3 A	Resistance (L-L, 20 °C)	4,9 Ω
Continuous stall torque	51 Nm	Synchronous inductance	8,3 mH
Continuous stall current	1,5 A	Number of pole pairs	5
Max. DC bus voltage	680 V	Hollow shaft diameter	16 mm

Figure 4. Result of a search for a suitable unit

The amount and type of information about the proposed drive unit given by this software tool depend on the information provided by its manufacturer. At present, the design time of a single drive unit is in the range of 21 to 31 seconds, if we limit the search to only drive units from Harmonic Drive AG, the CanisDrive series. The design time, according to the manufacturer's catalogue, lasts approximately 50 minutes in the case of a development worker who already has experience with this issue. The design of drive units using the created Drive Picker software tool is, therefore, approximately 96 to 142 times faster. It is, therefore, a very effective tool.

### 2.3 Preparation of taken 3D models

The knowledge database used by the Drive Picker software tool also contains 3D models of available drive units. In most cases, these are models taken over from the manufacturers. The main disadvantage of the models thus obtained is that they are available in the form of a single element. This can be quite limiting in the case of creating assembly models, which are subsequently to be subjected to, for example, kinematic or dynamic analyses. In such cases, the drive unit model must be divided into at least two parts, the stator, and the rotor. It is then necessary to assign weights and moments of inertia to these parts, which are corresponding to the actual drive units. Sometimes it is necessary to modify some 3D models more fundamentally, to achieve the required parameters, especially in terms of moments of inertia. Preparing models in this way is often a time-consuming process. But in the case of repeated use of such prepared models, the initially invested time and effort are returned in the form of reducing the time needed to design additional equipment. The addition of reference elements in the form of coordinate systems, axes and planes, with the help of which it would be possible, in this case, stator and rotor, to be connected to an assembly corresponding to a real drive unit, also proved to be very suitable. It is also appropriate to add reference elements, with the help of which it would be possible to insert the own drive units into the models of the proposed devices. Fig. 5a shows a modified 3D model of a CanisDrive 14A drive unit, Harmonic Drive AG, consisting of a stator and a rotor.



**Figure 5.** Modified model of Harmonic Drive CanisDrive 14A (a) assembly model of a drive unit, b) stator and rotor supplemented by coordinate systems, c) stator and rotor supplemented by axes and planes)

Fig. 5b, the stator and rotor are supplemented by the coordinate systems *cs\_stator\_cd\_14a* and *cs\_rotor\_cd\_14a*, which serve to form a strong bond between the two elements. The coordinate systems *cs\_drive\_unit\_stator\_cd\_14a* and *cs\_drive\_unit\_rotor\_cd\_14a* are designed to create a link between the drive unit and other elements of the proposed device. In FIG. 5c, the stator and rotor are supplemented by the axis *axis\_stator\_1\_cd\_14a* and *axis\_rotor\_1\_cd\_14a* and the planes *plane\_stator\_1\_cd\_14a* and *plane\_rotor\_1\_cd\_14a*, with the help of which it is possible to create a movable rotational connection between these two elements. When needed, the model can be supplemented with other suitable reference elements.

It is possible to proceed similarly when preparing other 3D models and elements, which can then be stored in a database and subsequently used in the creation of models of other devices. Models prepared in this way can also be used to automate the creation of assembly 3D models, as described below.

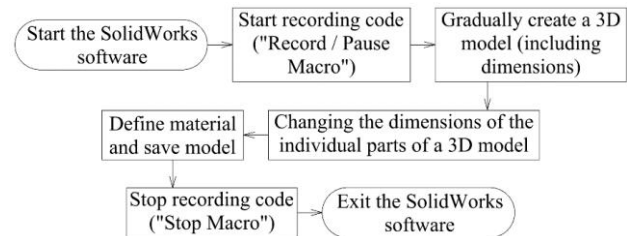
## 2.4 Preparation of parametric 3D models in the form of code in C# language

The knowledge database also contains its 3D models in two forms. The first are models created in SolidWorks software. It is possible to change individual dimensions, add or remove individual elements of geometry, change materials, etc. These are, therefore parameterized. They can be used both in the form in which they are stored in the knowledge database and as a basis for new models. By using such prepared models, it is possible not only to reduce the design time of the device but also to make better use of the time of developers working in CAD software [Reddy 2016].

The second form is 3D models prepared in the form of codes. These codes can be prepared manually or using the tools included with SolidWorks software. However, the first case is a process that is demanding in terms of knowledge and skills of the employee, who would create the codes, as well as in terms of time needed to write them. As the complexity of the models increases, so does the complexity and volume of the code. From the point of view of time-intensity, this is not a suitable way of preparing documents for the knowledge database. A more convenient way is to use the tools available in the SolidWorks CAD software, more precisely the possibility of recording the process of creating 3D models using the "Record

/ Pause Macro" function. This function records the individual actions that are performed when creating a 3D model.

As part of the preparation of the database of own 3D models, the VSTA C# format was chosen for storage. The choice of this format resulted from the effort to maintain a uniform programming language throughout the development, i.e. C#. Fig. 6 shows a modelling procedure for generating the required code.



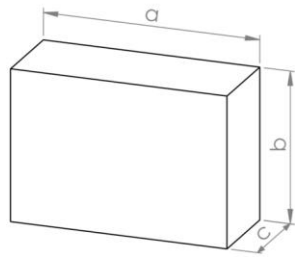
**Figure 6.** Modeling procedure when recording code in C# format

After the SolidWorks software is started, the "Record / Pause Macro" function is used to record the individual actions leading to the creation of the desired 3D model. To be able to modify the code in the required way at a later stage, it is necessary to create dimensions of the individual dimensions of the model. In the next step, the individual dimensions of the model are modified accordingly to the tree structure of the model from the first element to the last. This way, information on the dimensions of individual parts of the created model is arranged relatively clearly in one part of the code. Then the material or other properties are assigned to the model. The model created in this way is then saved. The last step is to end the function for recording the macro and then save it.

However, the code obtained in this way contains many lines that have no significance for the actual creation of the 3D model, for example, lines created by rotating the model or enlarging/reducing the view. One rotation of the model can be recorded in the form of tens of lines of code. Their execution can then significantly affect the time needed to generate the 3D model. Therefore, it is advisable to remove as many unnecessary lines as possible from thus obtained code.

However, in the case of more advanced functions, not all actions performed are recorded (e.g. for the "Circular Pattern" function, the code does not record information that the 3D model creator marked some elements of the array in such a way that they are not created). In these cases, it is necessary to approach the creation of a 3D model in such a way that it is possible to avoid problematic functions or to modify the generated code as necessary. However, these alternative ways are usually from the point of view of professionals less advantageous and also time-consuming.

To be able to use the code created from macros to create a parameterized 3D model, it is necessary to replace the values for individual dimensions with variables into which the required dimensions of the model are inserted before the actual generation. In Fig. 7 is an example of a portion of modified code for a parameterized model of a block.



```
// Change of dimensions
myDimension = ((Dimension)(swDoc.Parameter("D1@Sketch1")));
//myDimension.SystemValue = 0.1;
myDimension.SystemValue = a;
myDimension = ((Dimension)(swDoc.Parameter("D2@Sketch1")));
//myDimension.SystemValue = 0.07;
myDimension.SystemValue = b;
myDimension = ((Dimension)(swDoc.Parameter("D1@Boss-Extrude1")));
//myDimension.SystemValue = 0.04;
myDimension.SystemValue = c;
swDoc.ClearSelection2(true);

// Defining material
swPart = ((PartDoc)(swDoc));
swPart.SetMaterialPropertyName2("Výchozí", "C:/Program Files/"
    "SOLIDWORKS Corp/SOLIDWORKS/lang/czech/sldmaterials/" +
    "solidworks materials.sldmat", material);
swDoc.ClearSelection2(true);
```

Figure 7. Part of the modified code for creating a block

The block's dimensions a, b and c are defined as data variables of type double. By assigning numerical values to these variables, it is possible to create a model of a block of the required dimensions. In the second part of the code, the generated model is assigned material from the SolidWorks software database.

The code generated in this way can be used both to generate one type of 3D model and as a basis for creating multiple derived model types. The only difference is in the parts of the code that would be used to generate the 3D model. An example of the basis model (middle) and its derived types are shown in Fig. 8.

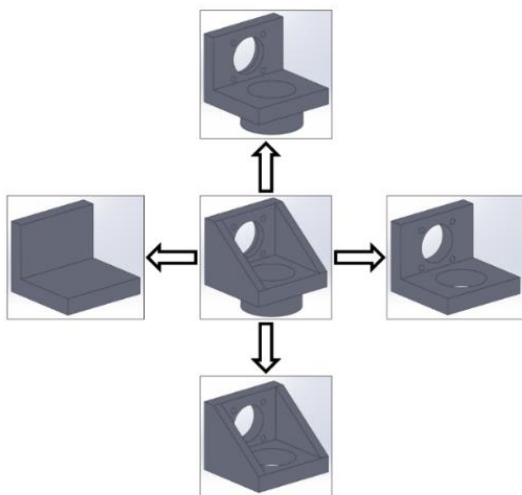


Figure 8. Basis model and its derived types

This way, it is possible not only to expand the usability of the generated code significantly but also to reduce the amount of code stored in the database and thus achieve its better clarity.

When preparing models in the form of codes, it is again appropriate to take into account how they are inserted into subsequently created assemblies and subassemblies, as was the case with models taken over from the manufacturers of the

given elements and equipment. In Fig. 9a is an example of a console model with a pair of coordinate systems. By a suitable location of these coordinate systems and orientations of their axes, it is possible to achieve a state where to create the desired connection with other elements of the proposed device it is enough to create only one connection with the help of these reference elements. Fig. 9b shows a possible result in the form of a connection of a console model with a pair of drive units. The models prepared in this way again open the way to automate the creation of assembly 3D models. However, the correct set of rules for the creation of reference elements in the form of coordinate systems, axes, planes, etc. plays an important role.

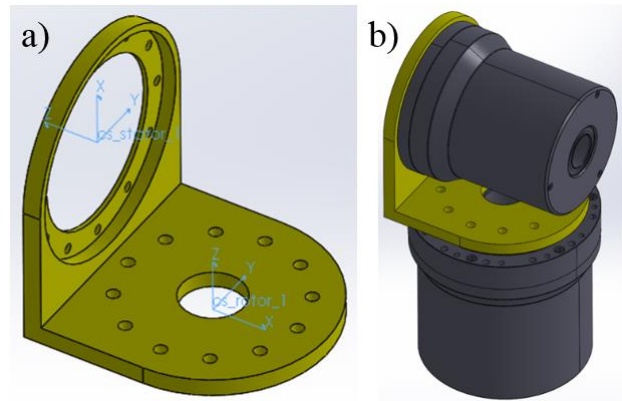


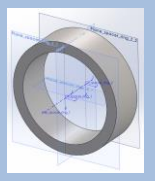
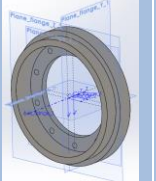
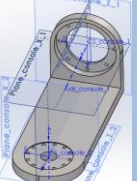
Figure 9. a) a console model with a pair of coordinate systems, b) an assembly model of a console with a pair of drive units

At first, glance, preparing 3D models in the form of codes can seem complicated and time-consuming. However, their creator, familiar with the appropriate procedures and limitations, can create the necessary materials in a relatively short time. In Table 1 is a comparison of the times required to manually create selected 3D models and the times required to create and edit codes into such a form that they can be stored in a database. The times it took to generate selected 3D models by using the codes created are also included. The difference between them is whether these models were generated with or without the use of the graphical environment (GUI) of SolidWorks software. Individual times were obtained as average values from five measurements. The creation of 3D models took place on a computer set equipped with an AMD Ryzen 5 2600X processor, 16GB of RAM, a graphics card equipped with NVidia GeForce GTX 1650 chip and a 500GB SSD hard drive.

In the case of the spacer ring, the preparation of the 3D model alone took approximately 1 minute. The remaining time was devoted to the preparation of reference elements in the form of coordinate systems, axes and planes. It is similar to the time-consuming preparation of reference elements for other models. From previous experience, it is possible to conclude that the preparation time of a 3D model in the form of a code is on average approximately two to three times more demanding than the manual creation of a given model. When modifying the code to achieve the result indicated in Fig. 8, or in the case of the need for additional modifications of the code to a form that cannot be achieved by recording the manual process of creating models, it is necessary to take into account the extension of the time required to achieve the desired result.

Models created based on codes can be further worked in the same way as models created manually. It is, therefore, possible to change dimensions (models contain dimensions), add,

remove, or modify their parts, etc. In this respect, there is no restriction.

Model name	Spacer ring	Flange	Console
3D model			
Manual creation	7.42 minutes	10.50 minutes	14.82 minutes
Code preparation	14.18 minutes	24.68 minutes	37.75 minutes
Model generation (GUI on)	10.5 seconds	13.7 seconds	19.8 seconds
Model generation (GUI off)	5.5 seconds	7.0 seconds	10.9 seconds

**Table 1.** Comparison of times needed to create selected 3D models and prepare 3D models in the form of codes

## 2.5 Automation of the process of creating assembly 3D models

Creating assemblies or subassemblies can also be a relatively time-consuming process. In the case of models with a repeating structure, but with different dimensions of individual elements, there is room for automation of this process. One of the possibilities is the use of parameterized models of elements of given assembly and their connection to a Microsoft Excel file, which stores a design table containing the dimensions of individual elements [Yong 2011], or references, which are used to interconnect selected dimensions of individual elements. A subsequent change of values in the design table can achieve the desired change in the dimensions of selected parts of the assembly. For work productivity, this can be a very attractive way of automating the design process of various devices or their components. In many cases, however, simply resizing individual elements of an assembly is not enough, but you need to replace some of the elements with others. One option is to use the SolidWorks software API again. Suitable code sequences can replace manual insertion of elements into the assembly and creation of links between them. In Fig. 10 there is a code sample in the C # programming language for loading and inserting a selected element (rotor of a CanisDrive drive unit 40A-100-AU-H-MGS-B) into an assembly.

The first piece of code opens the featured model that is to be inserted into the assembly in a new SolidWorks software window. The second part of the code is used to insert this model into the assembly. The third ensures that the window with the model is closed. When inserting a model of another element, all you need to do is change its name and path to its location. The process of inserting individual elements into an assembly can, therefore, be automated with the help of several lines of code and a list of elements with links to their location.

In Fig. 11 there is an example of code with which it is possible to create links between models of individual elements of an assembly. It is a creation of a strong connection, using coordinate systems, between the rotor and the stator of the CanisDrive 40A-100-AU-H-MGS-B drive unit.

```
// Open the model in a new SolidWorks software window
swAssembly = ((AssemblyDoc)(swDoc));
tmpObj = swApp.OpenDoc6("C:\\Working\\Robotic_arm\\" +
    "Drive_units\\canisdrive_40a_100_au_h_mgs_b_" +
    "rotor.SLDPRT", 1, 32, "", errors, longwarnings);
swAssembly = ((AssemblyDoc)(swDoc));

// Insert the model into an assembly or subassembly
swInsertedComponent = swAssembly.AddComponent5("C:\\ +
    "Working\\Robotic_arm\\Drive_units\\canisdrive_40a_" +
    "100_au_h_mgs_b_rotor.SLDPRT", 0, "", false, "", 0.0,
    0.0, 0.0);

// Close the window with the inserted model
swApp.CloseDoc("C:\\Working\\Robotic_arm\\Drive_units\\" +
    "canisdrive_40a_100_au_h_mgs_b_rotor.SLDPRT");
swMathUtil = swApp.GetMathUtility();
```

**Figure 10.** Code sample for loading of a model into an assembly

```
// Select the drive rotor coordinate system
boolstatus = swDoc.Extension.SelectByID2("cs_rotor_cd_" +
    "40a@canisdrive_40a_100_au_h_mgs_b_rotor-1@Assembly1",
    "COORDSYS", 0.0, 0.0, 0.0, true, 1, null, 0);

// Select the drive stator coordinate system
boolstatus = swDoc.Extension.SelectByID2("cs_stator_cd_" +
    "40a@canisdrive_40a_100_au_h_mgs_b_stator-1@Assembly1",
    "COORDSYS", 0.0, 0.0, 0.0, true, 1, null, 0);

// Establish a link between models
swAssembly = ((AssemblyDoc)(swDoc));
swMate = swAssembly.AddMate5(20, -1, false, 0.0, 0.0, 0.0,
    0.0, 0.0, 0.0, 0.0, 0.0, false, false, 0, out longstatus);
```

**Figure 11.** Code sample for creating a strong link between two models using coordinate systems

The first and second part of the code is used to select the coordinate systems of the stator and rotor models with which a strong bond is to be created. The third piece of code is used for the creation of that bond. Similarly, it is possible to create relationships using planes, axes, or other predefined reference elements of selected models. In this way, it is possible to create not only fixed but also movable bonds.

Again, the code structured in this way can be relatively easily modified to form different types of links between any elements of the assembly, as long as the elements of the assembly contain suitably spaced reference elements. In the case of the first part of the code, it is sufficient to change the existing data concerning the coordinate system, i.e. its name (cs\_rotor\_cd\_40a) and the name of the first of the models (canisdrive\_40a\_100\_au\_h\_mgs\_b\_rotor), after the name of another reference element and the name of the model of which this reference element is a part. Then you need to change the type of reference element. The location of the coordinate system and its corresponding designation within the "COORDSYS" code can be specified, for example, "AXIS" in the case of an axis or "PLANE" in the case of a plane. It is possible to proceed similarly when editing the second part of the code. However, keep in mind that it is not possible to create links from all available combinations of auxiliary elements. The last part of the code is based on the type of reference elements used to create the link. By changing the first two values in parentheses, it is possible to create links between different reference elements. In the case of coordinate systems it is a pair of values (20, -1), in the case of axes a pair of values (0, 0) and in the case of planes a pair of values (0, 1). In the case of other types of reference elements, or when creating links between different types of reference links, there are other combinations of these two values.

### 3 EXAMPLE OF USING A KNOWLEDGE DATABASE AND THE DRIVE PICKER SOFTWARE TOOL

Demonstration of the use of a knowledge database and a software tool for the design of compact drive units Drive Picker is performed on the design of a robotic arm with 5 degrees of freedom. The design of the robotic arm depends on the application, resp. the type of technology for which it is proposed. The choice of individual parts of the proposed device is derived from this. The design procedure is indicated in Fig. 12.

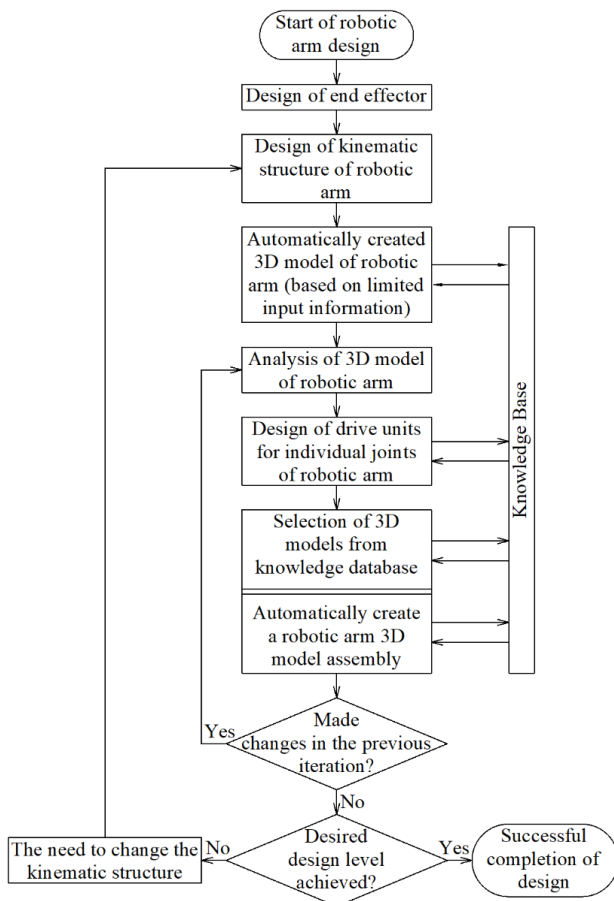


Figure 12. Robotic arm design procedure

In the initial phase of the design, the end effector is solved. Its type and properties depend on the activity for which the robotic arm is designed. In the case of manipulation tasks, these are often gripping types of effectors (e.g. jaw, vacuum or magnetic). In the case of technological tasks, the type of effector depends on the given technology (e.g. welding, paint application or laser beam cutting) [Mohyla 2014]. However, there are also combined or special end effectors. In this case, a double-jawed effector with a pneumatic drive was designed for handling a cylindrical object with a diameter of 52 mm, a length of 200 mm and a weight of 10 kg, the taken 3D model was modified in a similar way as indicated for compact drive units and added to the knowledge database.

The knowledge database contains sample code sequences for the design of robotic arms with an angular structure and 4, 5 and 6 degrees of freedom. Based on the sample code sequence, a 3D model of the robotic arm with 5 degrees of freedom was created (Fig. 13). Harmonic Drive AG drive units of the CanisDrive series have been designed for the individual joints. From size 14, in the case of the joint closest to the end effector, to size 40, in the case of the base of the robotic arm. Other elements of the arm model were generated using models

in the form of codes stored in the knowledge base. Modifying the sample code sequence to create this 3D model of the robotic arm to the desired shape took approximately 25 minutes. The actual generation of the model took 6.5 minutes. The total time for the preparation and creation of the robotic arm model thus slightly exceeded half an hour. The total time required to create a 3D model of this robotic arm manually was approximately 190 minutes, more than six times longer. This is provided that its creator had the necessary documents prepared in a similar way as in the case of the proposed knowledge database. Otherwise, the time required to create this 3D model manually can be many times longer, and the benefits of using the proposed knowledge database would be significantly more tangible.

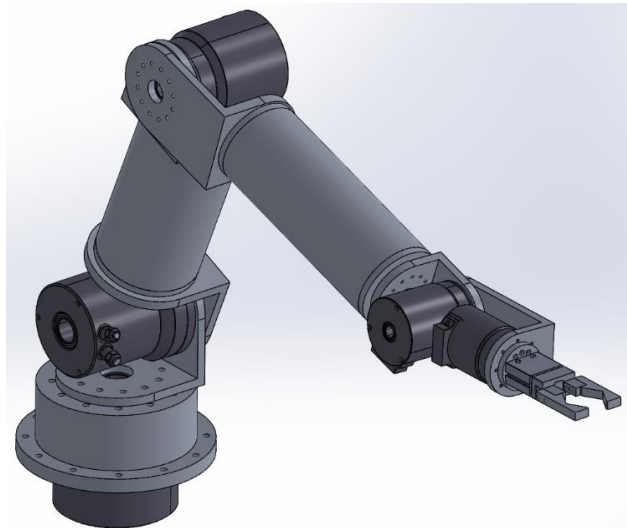


Figure 13. 3D model of a robotic arm

It turned out that if the potential of the proposed knowledge database is used appropriately, the time required to create the required 3D assembly model can be significantly reduced.

The 3D model created in this way would then be subjected to analyses in the SolidWorks software environment. Based on these analyses, it would be possible to determine the load, respectively required parameters of drive units in individual joints of the robotic arm. Based on these values, it would be possible to design suitable drive units of the individual joints subsequently. In the case of the design of robotic arms, the drive closest to the effector is always designed first. Then the assembly model of the manipulator is modified according to the newly designed drive unit, and analyses are performed again in the SolidWorks software environment. Based on their results, the newly designed drive unit is checked to see if it meets the requirements placed on it, or whether it is not possible to select a drive unit with even more suitable parameters. The same procedure is continued in the case of the drive units of the other joints of the robotic arm. It is thus an iterative task, as indicated in Fig. 12. The drive unit of each joint is therefore to be subjected to at least two design calculations. If we take into account the time required to manually perform the design calculation of the drive unit mentioned above, i.e. approximately 50 minutes, multiply it by two and multiply this value by the number of drive units of the robotic arm, i.e. 5, we get to the time value of 500 minutes. If the Drive Picker software tool is used, one design calculation would take between 21 and 31 seconds. In the case of performing ten design calculations, we get to the range of values from 210 to 310 seconds, i.e. 3.5 to 5.17 minutes. Even in the case of a comparison with the highest value of this time interval, the

time of manual design of power units would take almost 100 times longer, respectively would take approximately 494 minutes longer. It is therefore clear that by using this software tool, it is possible to reduce the overall design time of the device significantly.

The functionality of the Drive Picker software tool was tested using data obtained from the dynamic analysis, in which an object of manipulation in the form of a cylinder with a diameter of 52 mm, a length of 200 mm and a weight of 10 kg was manipulated along the marked trajectory. The trajectory has a length of 1314.16 mm, and the movement from the starting position (the manipulation object is green) to the target position (the manipulation object is blue) took 2 seconds, while the axis of the manipulation object was in a vertical position throughout the movement point.

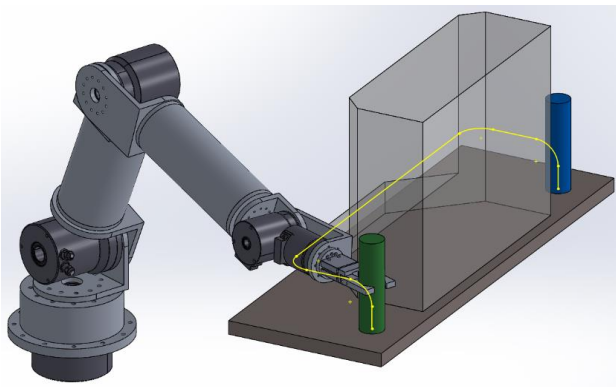


Figure 14. Trajectory of the object of manipulation in dynamic analysis

In Fig. 15 are graphs with course curves of torques, output speeds, radial and axial forces acting on individual drive units of the robotic arm. The drive units are numbered in order from the location in the joint closest to the end effector (Drive unit – joint 1) to the base of the robotic arm (Drive unit – joint 5).

Table 2 shows the maximum, average and minimum values of torques, output speeds, radial and axial forces and maximum values of tilting torques relating to the individual drive units. The values of the moments of inertia, which were obtained from the CAD software SolidWorks in the initial position of the robotic arm, i.e. at the beginning of the given trajectory, is also given here. The last row of the table contains types, respectively the size, of the Canis Drive series drives, which with their parameters would suit the load determined during the dynamic analysis. The design of the drive units was also performed manually according to the manufacturer's catalogue (Harmonic Drive LLC, 2020). The same result was obtained as in the case of using the Drive Picket software tool, which again confirmed its functionality.

Currently, the Drive Picker software tool is being developed with maximum emphasis on the reliability of drive design. In the next phases of development, it is planned not only to expand its functionality but also to optimize the speed of the power unit design process.

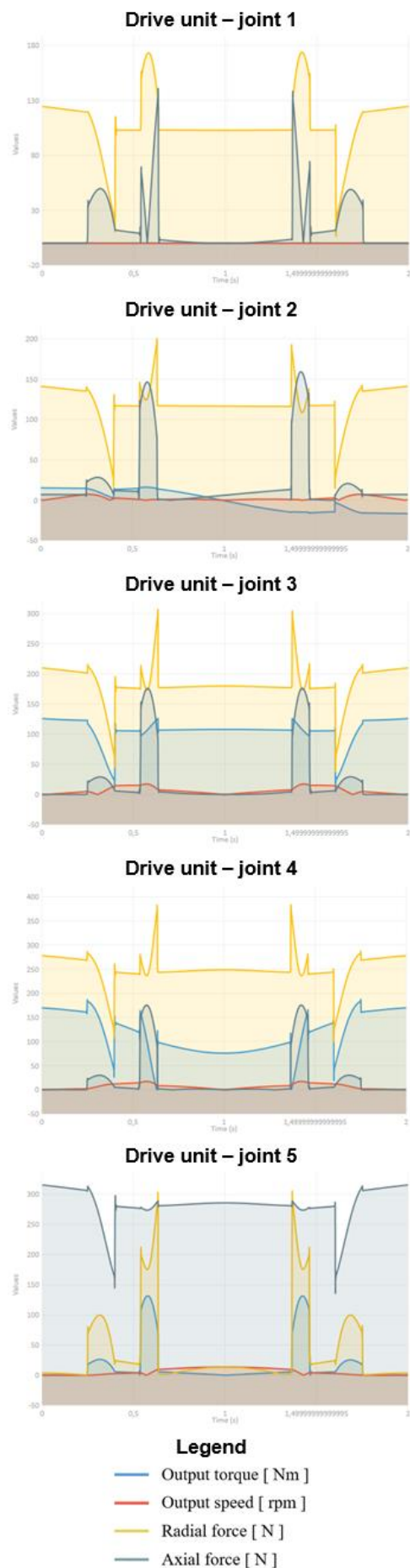


Figure 15. Load course curves of drive units



Quantity		Joint 1	Joint 2	Joint 3	Joint 4	Joint 5
Torque (Nm)	Max.	6,49	16,72	126,6	187,5	131,6
	Avg.	0,001	13,85	105,7	117,4	44,61
	Min.	0,00	0,02	5,25	0,12	0,00
Output speed (rpm)	Max.	0,08	7,23	17,32	17,31	13,69
	Avg.	0,02	2,15	6,53	6,61	5,51
	Min.	0,00	0,00	0,00	0,00	0,00
Radial force (N)	Max.	173,7	200,3	307,3	383,5	305,7
	Avg.	113,7	127,3	185,6	249,1	107,6
	Min.	6,97	15,22	38,64	100,5	0,04
Axial force (N)	Max.	140,8	159,1	176,4	175,6	315,5
	Avg.	35,43	43,14	101,7	101,1	281,6
	Min.	0,00	0,00	0,00	0,00	136,5
Tipping moment (Nm)	Max.	33,64	45,65	132,7	145,8	209,8
Moment of inertia (kg-m <sup>2</sup> )	-	0,04	0,10	8,15	14,11	12,23
Drive unit	Type	HD Canis Drive 14A (ratio 50)	HD Canis Drive 14A (ratio 80)	HD Canis Drive 25A (ratio 100)	HD Canis Drive 25A (ratio 100)	HD Canis Drive 25A (ratio 100)

Table 2. Values obtained from dynamic analysis

#### 4 CONCLUSIONS

The previous chapters described the Drive Picker software tool, how to prepare 3D models for a knowledge base, the ability to automate the creation of assembly models using the SolidWorks software API, and the knowledge base, which stores the necessary data and algorithms, and serves as a basis. Knowledge design system for designing selected elements of mechatronic devices. Verification of the functionality and benefits of the proposed solutions were demonstrated on a model of a robotic arm with five degrees of freedom. Although the preparation of materials for a knowledge database is not a trivial matter and is also time-consuming, the return on energy input can be very fast, as shown in the examples. Currently, the ability to make full use of all knowledge base resources is tied to the need for at least a basic knowledge of working with the SolidWorks software API and programming in C#. However, further, the development includes not only the further expansion of several software tools for designing standardized and non-standardized elements used in the construction of mechatronic devices and data stored in the knowledge database but also a graphical interface that would not only simplify data access but also eliminated the need for API and C# programming skills.

#### ACKNOWLEDGMENTS

This article was developed with the support of the project Research Centre of Advanced Mechatronic Systems, reg. no. CZ.02.1.01/0.0/0.0/16\_019/0000867 in the frame of the Operational Program Research, Development and Education.

#### REFERENCES

- [Farhan 2012] Farhan, U. H., Tolouei-Rad, M. and O'Brien, S. An Automated Approach for Assembling Modular Fixtures Using SolidWorks. *International Scholarly and Scientific Research & Innovation*, 2012, Vol.6, No.12., pp 365-368. ISSN 2682-2685.
- [Harmonic Drive 2021] Harmonic Drive SE. 301 [online]. Copyright © 2021 [cit. 5.10.2021]. Available from: <https://harmonicdrive.de/en/home>
- [Chen 2013] Chen, T., Yan, X. and Zhonghai, Y. The Research and Development of VB and Solidworks-Based 3D Fixture Component Library. *Applied Mechanics and Materials*, February 2013, Vol. 300-301, pp 301-305. ISSN 1662-7482.
- [Isermann 2002] Isermann, R., *Mechatronic design approach*, in Bishop, R.H. (Ed.): *The Mechatronics Handbook*. Boca Raton: CRC Press, 2002.
- [Jiang 2011] Jiang, Y. J. The Modeling of Thread-Rolling Die-Plates Based on the "Design Table" Functions in CAD/CAE/CAM. *Applied Mechanics and Materials*, October 2011, Vol. 130-134, pp 499-503. ISSN 1662-7482.
- [KISSsoft 2021] KISSsoftAG. [online]. Copyright ©2021 KISSsoft AG [cit. 15.05.2021]. Available from: <https://www.kisssoft.com/en>
- [Kyura 1996] Kyura, N. and Oho, H. Mechatronics – an industrial perspective. *IEEE/ASME Transactions on Mechatronics*, March 1996, Vol. 1, Issue.1, pp 10-15. ISSN 1941-014X.
- [Lad 2014] Lad A.C., Rao A.S. Design and Drawing Automation Using Solid Works Application Programming Interface. *International Journal of Emerging Engineering Research and Technology*, October 2014, Vol.2, No.7., pp 157-167. ISSN 2349-4395
- [Lindemann 2009] Lindemann U. *Methodische Entwicklung technischer Produkte: Methoden flexibel und situationsgerecht anwenden*. Berlin: Springer-Verlag, 2009. ISBN 978-3-642-01423-9
- [Mohyla 2014] Mohyla P., Kubon Z., Cep R., Samardzic, I. Evaluation of Creep Properties of Steel P92 and Its Welded Joint. *Metalurgija*, Vol. 53, No. 2, pp. 175-178. ISSN 0543-5846.
- [Reddy 2016] Reddy E.J., Sridhar C.N.V., Pandurangadu V. Research and development of knowledge based intelligent design system for bearings library construction using SolidWorks API. *Advances in Intelligent Systems and Computing*. India: Springer, 2016. ISBN 978-3-319-23257-7
- [Reddy 2018a] Reddy E.J., Rangadu V.P. Development of knowledge based parametric CAD modelling system for spur gear: An approach. *Alexandria Engineering Journal*, December 2018, Vol.57, No.4., pp 3139-3149. ISSN 2249-6890
- [Reddy 2018b] Reddy E.J., Venkatachalapathi N., Rangadu V.P. Development of an approach for Knowledge-Based System for CAD modelling. *Materials Today: Proceedings*, January 2018, Vol.5, No.5., pp 13375-13382. ISSN 2214-7853
- [Sun 2011] Sun, B., Qin, G. and Fang, Y. Research of standard parts library construction for SolidWorks by Visual Basic. *International Conference on Electronic & Mechanical Engineering and Information Technology*, August 2011, pp 2651-2654. ISBN 978-1-61284-088-8

[VDI 2004] VDI-Fachbereich Produktentwicklung und Mechatronik. Entwicklungsmethodik für mechatronische Systeme. Berlin: Beuth Verlag, 2004.

[Yong 2011] Yong J. J. The Modeling of Thread-Rolling Die-Plates Based on the "Design Table" Functions in CAD/CAE/CAM. Applied Mechanics and Materials, October 2011, Vol. 130-134, pp 499-503. ISSN 1662-7482

[Zhang 2011] Zhang, Z. W., Xin, Y. G., Zhang, W. and Song, J. C. Three-Dimensional Manifold Checking Based on VB.NET. Advanced Materials Research, September 2011, Vol. 338, pp 263-266. ISSN 1662-8985

#### **CONTACTS:**

doc. Ing. Milan Mihola, Ph.D.  
VSB - Technical University of Ostrava, Department of Robotics  
17. listopadu 2172/15, Ostrava-Poruba, 708 00, Czech Republic  
+420 596 995 445, milan.mihola@vsb.cz,  
<https://www.fs.vsb.cz/354/en>

Ing. Zdenek Zeman  
VSB - Technical University of Ostrava, Department of Robotics  
17. listopadu 2172/15, Ostrava-Poruba, 708 00, Czech Republic  
+420 596 991 209, zdenek.zeman@vsb.cz  
<https://www.fs.vsb.cz/354/en>

Ing. David Fojtik, Ph.D.  
VSB - Technical University of Ostrava, Department of Control  
Systems and Instrumentation  
17. listopadu 2172/15, Ostrava-Poruba, 708 00, Czech Republic  
+420 596 994 193, david.fojtik@vsb.cz  
<https://www.fs.vsb.cz/352/en/>