Intelligent Assessment of Structure Correctness using Antipatterns

Wojciech Kacalak, Maciej Majewski, Andrzej Tuchołka
Faculty of Mechanical Engineering
Koszalin University of Technology
Koszalin, Poland
{wojciech.kacalak, maciej.majewski, andrzej.tucholka}@tu.koszalin.pl

***Abstract*— Authors propose a method of normalizing and analyzing structures, which enables automated cross-comparison of features observed in tested structures with: predefined structures and anti-patterns. Additionally, a language, enabling symbolic representation of structural knowledge is presented, and used in the demonstration of the proposed method applied to mechanical shafts, and used for qualitative purposes with more complex calculation models.**

Keywords—artificial intelligence; machine design; automated design systems; structural feature language; anti-patterns

I. Introduction

One of the key challenges in the design of mechanical structures and structural analysis, is the ability of engineers to meaningfully evaluate the structural properties and structures [6, 9]. The quality and speed of this evaluation is a key element of the design, construction, diagnostic, and maintenance processes.

Existing methods require manual creation of models for each of the tested structures, or executing simulations which (due to their nature) either lack in speed or precision when evaluating complex models. Both of these limitations make it very difficult to apply genetic or unsupervised machine learning algorithms, where potentially vast scope of solutions has to be quickly evaluated.

An additional difficulty lies in automation of including a knowledge based model in a rich and quantitative description of the tested structure. Inclusion of such knowledge provides an opportunity for a quick and cumulative increase in baseline quality of designed structures, and also allows advanced solution-finding algorithms to enhance the collaborative intelligence of human operators and their tools. The particular use of anti-patterns (anti-patterns are in this context considered as known and incorrect examples of structures) [1, 3, 4, 7] provides the future algorithms with the ability to explore an open set of possible correct solutions, taking as the only reference already identified, incorrect ones.

Proposed method enables cross-testing of structures providing mechanical designers with an ability to check different options against each other, but also easily including known, predefined solutions in the analysis (be it correct ones or anti-patterns).

II. Interactive Systems for Designing Machine Elements and Assemblies

The presented research involves the development of intelligent interactive automated systems for designing machine elements and assemblies using descriptions of structural elements' features in a natural language. Realization of the automated design processes is in conditions of uncertainty and with non-repeatable processes. We propose a new concept [2] which consists of a novel approach to these systems, with particular emphasis on their ability to be truly flexible, adaptive, human error-tolerant, and supportive both of design engineers and data processing systems. The comparison of the proposed new automated design system with the present system of carrying out design tasks is shown in Fig. 1.

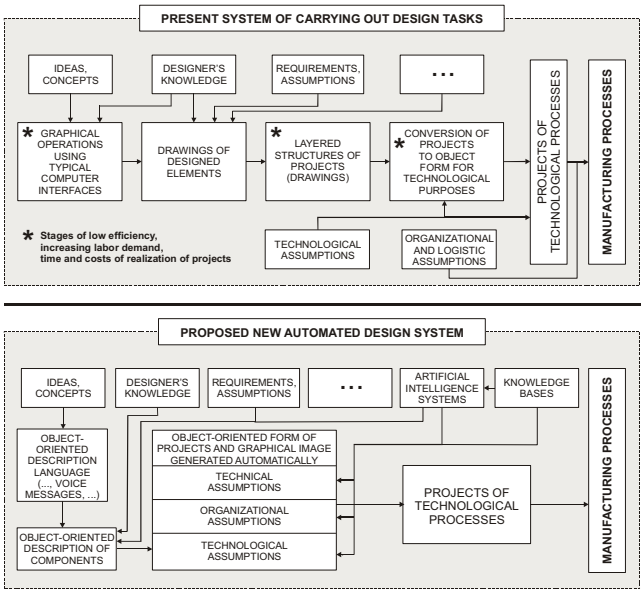


Fig. 1. Comparison of the proposed new automated design system with the present system of carrying out design tasks

The proposed interactive automated design system (Fig. 2) contains many specialized modules and it is divided into the following subsystems: subsystem for communication between designers and the intelligent CAD system, subsystem for

978-1-4673-9795-7/15 $31.00 © 2015 IEEE
DOI 10.1109/CSCI.2015.131

design engineers' voice messages content analysis, construction analysis subsystem, construction notation subsystem, construction rating subsystem, subsystem for visualization and CAD system control, design process optimization subsystem, construction decoding subsystem.

In this system, artificial intelligence methods allow communication by speech and natural language, resulting in analyses of design engineer's messages, analyses of constructions, encoding and assessments of constructions, CAD system controlling and visualizations. The system is equipped with several adaptive intelligent layers for human biometric identification, recognition of speech and handwriting, recognition of words, analyses and recognition of messages, enabling interpretation of messages, and assessments of human reactions.

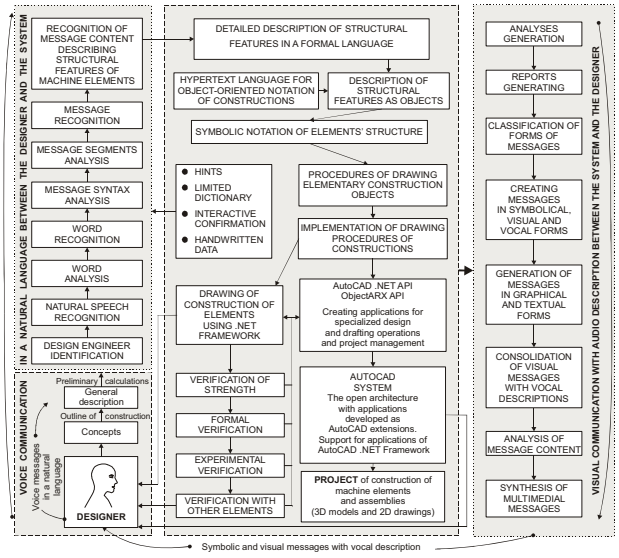


Fig. 2. Concept of design processes using interactive automated systems

III. RELATIONS BETWEEN OBJECTS' GEOMETRICAL PROPERTIES

In automated systems for designing machine components and assemblies, relations between an object, of a given class, element's geometrical properties as well as geometrical relations between those objects are key. A diagram of such relations, by the example of a single key-type sleeve coupling, is presented in the Fig. 3. The geometrical properties identification subsystem consists of the following types of geometrical relations:

- 1 relation type - is present in a particular element of a given type class. In the presented example, it is a dimensional relation determining the position of the axis of the hole in the crankpin (element 1) in relation to its facing surface. And for the object of class sleeve it is a dimension marking the distance between the hole's axis (element 1), and its facing surface.
- 2 relation type - determines the geometrical relation between geometrical properties of different elements of a given object (machine component). For a shaft and a sleeve of a sleeve coupling these are dimensions determining the distance between holes' axes. The axis of the first hole is connected with the crankpin (element 1), and the axis of the other hole belongs to the shaft's step (element 2). The situation is analogous in the case of the sleeve object.
- 3 relation type - geometrical relations between different objects in an assembly. In machine designs these are relations between geometrical properties of the 1st and 2nd type, belonging the object of a given class having such relations with other classes of objects. Third-type relations usually specify mutual position of construction's components. In a single key-type sleeve coupling it is crucial to ensure accuracy of mutual position of the shaft hole's axis (Element 1) and the sleeve hole's axis (Element 1). For this reason the third-type relation, in the context of the provided example, means that the axes of the aligned holes must be coaxial.

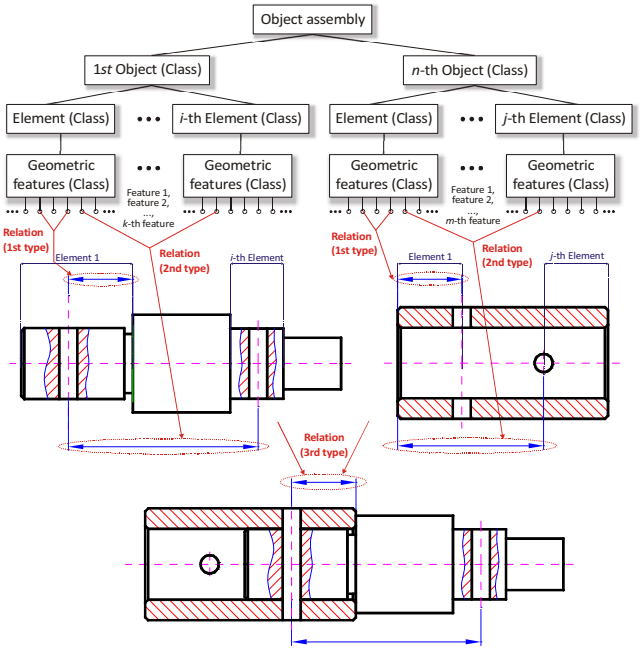


Fig. 3. Relations between an object, of a given class, element's geometrical properties and geometrical relations between those objects by the example of a single key-type sleeve coupling

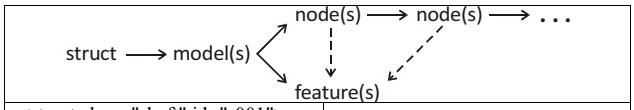
IV. KXML REPRESENTATION OF STRUCTURES

Assuming a reference classification of structures (i.e. ISO) grants us with the ability to assign relevant features and calculation methods to the model. It is obvious that the features of pipes can be reused among them, while being different from ones found in electric wires or mechanical joints. Still, for most purposes, one can use any of commonly used classification to satisfy the need for a general abstraction of structures.

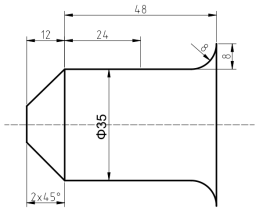
The representation of the structure is built upon a set of models, defining a set of features used in it to describe its tree of nodes defining the structural dependencies.

To enable the possibility of using a library of predefined structures, this syntax can be used in two contexts: as a pre-defined template of a structure, or its concrete description. The difference between these can be observed after verifying if all of the features required for a given class of a node in the particular model, have been provided with values. The additional flexibility built into this method is based on optional support of both: composition and inheritance between structural nodes via the class attribute. Both: the class based relations and the library mechanism, provide a common denominator that can be used for batch processing or flexibly structuring the analysis.

TABLE I. PRESENTATION OF THE BASE SYNTAX STRUCTURE




```
<struct class="shaft" id="s001">
<model class="mechanical">
  <feature id="l" unit="mm" />
  <feature id="R" unit="mm" />
  <feature id="d" unit="mm" />
  <feature id="angle" unit="deg"
vector="(0,1,0)" />
  <node class="spigot" id="p001">
    <feature id="l">48</feature>
    <feature id="d">35</feature>
    <node class="mill">
      <feature id="l">12</feature>
      <feature id="angle">45
      </feature>
    </node>
    <node class="thread">
      <feature id="l">24</feature>
    </node>
    <node class="undercut">
      <feature id="R">8</feature>
      <feature id="d">8</feature>
    </node>
  </node>
</model>
</struct>
```



Each of the keywords used in this notation provides both: a structural and a functional interpretation (dependent on the class and other attributes):

- STRUCT - a root KXML element providing a top level, singular access to the contained structure; storing multiple struct nodes in one KXML file is allowed and supported for easy and flexible creation of collections of structures. Attributes: class, id.
- MODEL - an element of the syntax used to define the model used for analysis while defining the trees of NODEs contained inside. Attributes: class, id.
- FEATURE - an element on one side defining a reused (between KXML elements), measurable property of the structure, model or a node, and also providing concrete values describing the structure. Attributes: class, id, unit, vector, value.
- NODE - the main structural element of KXML allowing the definition of the structure of the description through nesting of the elements, ability to differentiate between the types of the relations between nodes for each of the models, and to relate the nodes with each other to represent implicit relations between the nodes. Attributes: class, id.

Normally the scope of features will directly depend on the chosen classification model, but it is important to note, that the feature scopes might overlap among the chosen models. This means that the final structure is a combination of all node trees in defined models and can form a directed graph to describe the complete structure.

A. RELATIONS BETWEEN ELEMENTS OF THE SYMBOLIC LANGUAGE

The structures in the proposed language, are defined using differently classified models. This classification is considered a reference definition of the names, and the approach to number formatting, or practices in scope of provided information. This approach also ensures, that the data that has been provided for processing purposes, has a concise and concrete meaning to avoid unnecessary noise in its numerical analysis.

TABLE II. FOUR MODEL BASED RELATION CLASSES WITH EXAMPLES

<table>
<tr><td colspan="2">Sharing the common definition of the feature by the structure of nodes in the same model</td></tr>
<tr><td><model class="mechanical">
  <feature id="l" unit="mm" />
  <feature id="R" unit="mm" />
  <feature id="d" unit="mm" />
  <node class="spigot" id="p001">
  <feature id="l">48</feature>
  <feature id="d">35</feature>
  <node class="mill">
  <feature id="l">12</feature>
  </node>
  <node class="thread">
  <feature id="l">24</feature>
  </node>
  <node class="undercut">
  <feature id="R">8</feature>
  <feature id="d">8</feature>
  </node>
  </node>
</model></td><td>Each feature instance, regardless of the level of nesting the node, is using the same feature definition provided in the model.

Example on the left includes the feature “l” - the length measured in millimeters, and being provided a concrete value of the spigot, mill and the thread nodes.</td></tr>
<tr><td colspan="2">Sharing the information about node’s identity between different models</td></tr>
<tr><td><model class="mechanical">
  <feature id="l" unit="mm" />
  <feature id="d" unit="mm" />
  <node class="spigot" id="p001">
  <feature id="l">48</feature>
  <feature id="d">35</feature>
  </node>
</model>
<model class="electrical">
<feature id="rho"
unit="Ohm*m"/>
  <node class="conductor"
id="p001">
  <feature id="rho">10E-8
  </feature>
  </node>
</model></td><td>Each node definition, regardless of the nesting structure in a different model, can be defined with the same id relating the node with one from a different model.

Example on the left includes the node “p001” - which is described using separate features in two different models.</td></tr>
</table>

Referencing the feature values from different models

```
<struct class="shaft" id="s001">
<model class="geometrical">
  <feature id="S" unit="m^2" />
  <feature id="V" unit="m^3" />
  <node class="cylinder"
id="p001">
    <feature id="S">
      PI*(p001.d/2)^2</feature>
    <feature id="V">
      p001.S*p001.l</feature>
  </node>
</model>
<model class="mechanical">
  <feature id="l" unit="mm" />
  <feature id="d" unit="mm" />
  <node class="spigot" id="p001">
    <feature id="l">48</feature>
    <feature id="d">35</feature>
  </node>
</model>
<model class="electrical">
<feature id="rho"
unit="Ohm*m"/>
  <feature id="R" unit="Ohm" />
  <node class="conductor"
id="p001">
    <feature id="rho">
      10E-8</feature>
    <feature id="R">
      rho*p001.l/p001.S</feature>
  </node>
</model>
</struct>
```

Since the values of the features can contain functions to be evaluated, and by so are implicitly linked to values of other features, in possibly different units, and classification models, this should be done with high diligence.

The analysis performed on such defined structures is strongly suggested to be first pre-calculated and verified before applying further numerical or lexical processing methods.

Example on the left presents a more complicated, but allowed by the language, structuring of the model crossing dependencies. Such description, after parsing yields a structure with its basic mechanical, electrical, and geometrical features linked with each other in one entity "p001".

For the purposes of achieving a more complete description of relations between the nodes, it is possible to reference nodes not only within one model, but also between them. Notice, that as much as the features of the nodes are comparable only within the same model (or referencing the same feature definition in the same model classification), while the referencing relation between the nodes from separate models should be only structural (this is node the same node as...) or optionally value references (this value of the feature is calculated based on the value of ...).

The structure of nodes created for each of the models, allows for only unitary placement of each node in a tree structure built per the model. The node ids have to be unique in the model or be automatically assigned a unique id in case of no provided identifier. As mentioned above, reusing of the node id between models is interpreted as co-identity. While nodes contained within the model form trees, the relations possibly differing between models form a directed graph, with models linked at least on the root level of nodes.

Additionally, the concept of nodes, through their CLASS attribute, allows forming abstract relations that the designer wants include in the structure. The features defined directly in such abstract class are considered available for it, still the main data carrying concept is the overall structure of nodes, even if further parametrized via the composition (or other) node. This approach, in addition to the language defined, adds further flexibility into the method enabling inclusion of abstract know-how into the description.

Approach taken by this method, allows inclusion of further enhancements to the structure with the ability to define own classification models and classes of nodes providing a unique set of functions enabling interpretation of the representation of the structure. To illustrate this approach, let's take the abstract node representing the milling direction into consideration. Such node, grouping several spigots of the shaft, provides additional information to the description of the element being not only the measurement of its features. This approach also opens the possibility of defining compositions of elements or their more complex collections in form of abstract classes of nodes.

Usage of the value of the ID field in the CLASS of a KXML element (interpreted as inheritance) provides the language with a method of building a re-usable library of both features and nodes. Additionally, usage of this type of a relation for representation of the structures adds further depth to the description of implicit relations in the design.

As much as nodes or features with different values of the ID attribute represent different instances, no meta-data comes along the usage of common but abstract CLASS names. As mentioned earlier, reusing the same values of the CLASS supports comparability of features and nodes, but by using a defined ID as a CLASS name it is possible to form an inheritance tree including features or structures of nodes and features assigned to them.

B. FEATURE VALUES AS FUNCTIONS

Values of features can be either values or functions that when evaluated provide a value. If the value of the function cannot be evaluated (i.e. one of the references is incorrect) a NULL value will be returned for the given feature (same as when the value of the entity has not been provided). The functions available for usage to calculate the value will depend on their implementation in the parser, and as much as there are many mathematical libraries, they have to be selected with care, and considering that even different versions of the same library can yield different results. These vary due to using different calculation models, algorithms, assumptions, constants, and errors. This further enforces the need for a symbolic representation, with a requirement of adequate and standardized approach to analysis of the mechanical structures.

The ability to calculate the functions referencing values of other features, requires the parser to wait until the whole structure has been loaded. This approach makes it easier to write the symbolic representation, not having to take into consideration the ordering of the nodes and features.

C. PARSING AND VALIDATION

The main requirements for the implementation of parsing mechanisms for this symbolic structure is maintaining the structure of nodes and leaving the calculation of feature values as a last and optional step. Feature values, when represented as functions, allow analysis of the implicit relations between the nodes, while the concrete value lacks that information. Depending on the type of the analysis, either or both of the states of the value can be useful and if possible, both should be maintained.

To adequately address the requirements of the proposed symbolical description, the parsing process should be executed as follows:

- Locating and loading the file with either an XML or JSON parser. During this step it is assumed that a notation based validation will be performed by the library in use.
- Traversing loaded data (structure containing multiple models defining features and a structure of nodes with concrete feature values) to build an adequate (for the analysis) in-memory representation of the description, independent of the notation used to store it.
- Integrity of the loaded structure should be validated against the rules for relations defined earlier. This phase should be further adjusted to the input requirements of the analysis, but the integrity of the structure can be tested with:

  a) verifying that the minimum data is provided for processing;

  b) checking if used feature ids are defined in all cases;

  c) detection of non-unique (in model) IDs in the node structure, not-looped node's class trees;

  d) successful parsing and loading into memory each of the functions along with the detection of stack loops;

  e) verification of references to other feature values, the existence of data that is referenced, and verification/normalization of units.
- As the last step of the process, all of the defined models for the structure should be integrated into one, in-memory, directed, graph, data structure. The integration of the models, consists of merging them taking node IDs as the reference identity points. For the merging to be successful, there cannot be any nodes disconnected from the structure (i.e. when there is no node IDs overlap between the models).

V. Calculation Models

Having defined a concrete description of tested elements we can serialize the data contained in it to a form adequate for the qualitative algorithm we want to apply. Authors have not yet fully explored the capabilities of inclusion of the node graph in the representation, but even simple models (1) provide quick and relatively detailed representation of the structure's quality.

$$\sum_{n=1}^{k} \frac{w_n}{\Delta d_n};\ w_n \geq 1;\ \Delta d > 1 \qquad (1)$$

for k compared features, where: w-weight, Δd-distance from reference feature.

The normalization of feature values in this case, due to the nature of the summation function used, is two-fold: abstract for the differences in scale between same features in different elements, and for differences in magnitude between different features of one element. We suggest to use bounded non-linear functions for fine-tuned normalization. Regardless of the approach, the normalization requirements have to be correlated with the method applied by chosen algorithm.

For the purposes of a specific analysis, the values of a normalized, symbolic description have to also be serialized in a consistent manner, so that the algorithms using it can make assumptions about the data's structure. In addition to the simple arrays and binary forms, we can present input data as a binary image (Fig. 4.) where rows list all processed features and the assignment of the active bit to one of the columns represent its value. Activated bit will depend on the magnitude of the normalized feature value. This approach can lead to precision loss that is dependent on the number of columns allowing differentiation between feature values, however a gradable level of precision provides potential advantages in speed of calculation without significant quality loss.

Having in mind the significance of the calculation models enabled by neural networks and machine learning algorithms, we propose usage of such normalized, symbolic representation of structural features (among other neural computing models) with a Hamming net [5] (Fig. 4) and a probabilistic network [8] (Fig. 5).

Applying a Hamming net to the feature data of the tested mechanical elements, the normalized input values (in range: 0..1) provide difference in signal only for a different states in each of b different features. The net is built upon the Hamming distance using the difference in the number of bits. Using this algorithm, the comparison of the feature values against each other doesn't make much sense, as the result will always be the same (one bit lit, just in different locations will always yield 0 or 2 as the Hamming distance). Instead, each of the feature values is considered a bit, providing a value directly related to the differences in between the structures. This means that as much as there is no direct valuation or grading between tested elements, the method provides a quick way of locating the most similar anti-patterns and making a quality decision based on the similarity to one of the anti-patterns (possibly also weighed using Hamming's weight function).

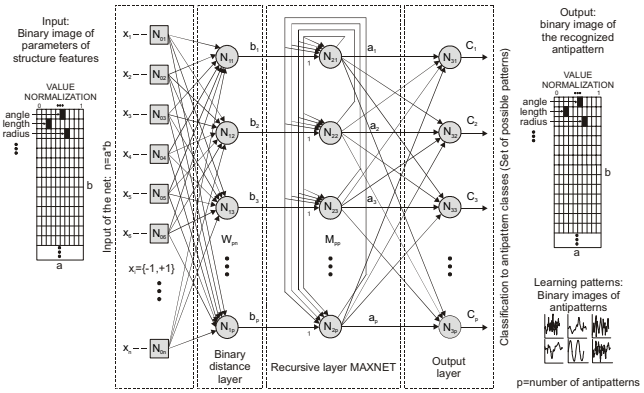


Fig. 4. A schema of applying a Hamming net to normalized feature values

Both approaches (Fig. 4., Fig. 5) enable classification of structures basing on their features. In the Hamming net, the distance between two compared structures (being the bias of the neuron function) is calculated as a Hamming distance

(number of differing bits, in our case rows) between two serialized values. The precision of this approach can also be further enhanced by taking into consideration the positive and negative interpretation of the distance, as the reference structure can be compared against an anti-pattern or a correct structure.

In case of a probabilistic neural network (Fig. 5.), the model is calculating the probability to achieve the same goal of general classification of structures. The benefits of applying this class of neural networks arise largely from being able to handle much larger variations in input data, resulting in a more flexible but possibly less precise qualitative calculation model.

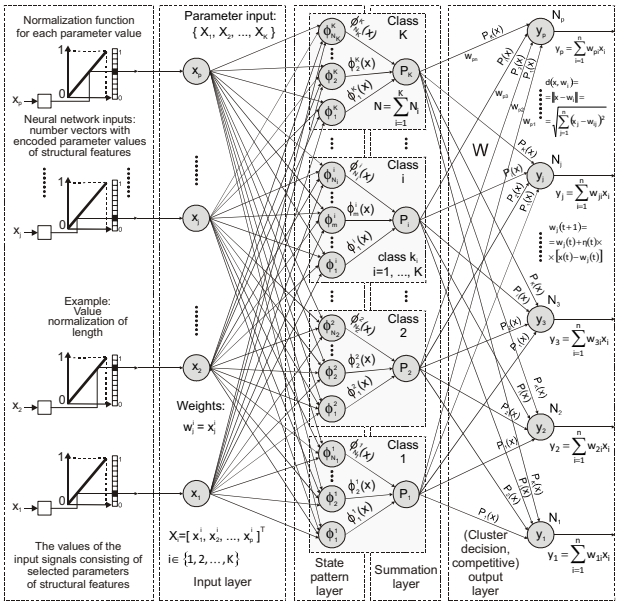


Fig. 5. Probabilistic neural networks for normalization of structures

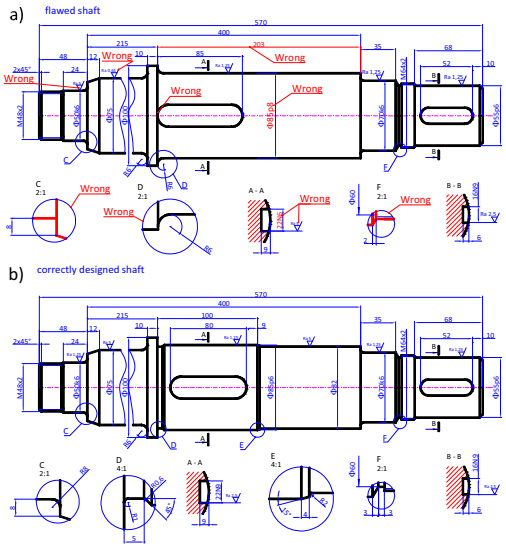


Fig. 6. A machine shaft with errors: a) an antipattern, b) correct design

The benefits of using anti-patterns are lack of constraints when designing new solutions and lesser likelihood of making errors typical of anti-patterns. An example of applied methodology for design solution and anti-pattern correspondence evaluation are presented in Fig. 6.

VI. Conclusion and Summary

Proposed symbolic notation and its processing methods, provide a low cost, flexible, and universal way of benchmarking the structural quality of mechanical elements. This approach enables mechanical structure designers to easily incorporate common design knowledge mixed with additional design quality requirements into their tool-set. Flexibility and low cost of calculations, enable usage of this method for verifications of automatically produced designs by genetic and machine learning algorithms. Furthermore, the symbolic object notation allows quick verification of completeness of the input data, as it can be quickly compared in an automated manner with the required scope of input data for each class of used elements. Another side effect of replacing the graphical representation with a symbolic one, is the avoidance of errors induced by overlapping or unreadable elements on the graphic. Additionally, the universal character of the symbolic representation of structures and their and anti-patterns, hints that a similar approach can be applied in other disciplines of science. Other potential uses of this method include the ability to easily translate structural descriptions between different metric systems, visual representation models and techniques.

Acknowledgment

This project was financed from the funds of the National Science Centre (Poland) allocated on the basis of the decision number DEC-2012/05/B/ST8/02802.

References

[1] W.J. Brown, R.C. Malveau, H.W. McCormick, T.J. Mowbray, AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis, Wiley 1998.

[2] W. Kacalak, M. Majewski, "New Intelligent Interactive Automated Systems for Design of Machine Elements and Assemblies," Lecture Notes in Computer Science, vol. 7666, Part IV, Springer, pp. 115-122.

[3] A. Koenig, "Patterns and Antipatterns," Journal of Object-Oriented Programming, vol. 8/1, pp. 46-48. 1995.

[4] J. Long, "Software reuse antipatterns," ACM SIGSOFT Software Engineering Notes, vol. 26/4, pp. 68-76. 2001.

[5] M. Majewski, J.M. Zurada, "Sentence Recognition Using Artificial Neural Networks," Elsevier Knowledge-Based Systems, vol. 21/7, pp. 629-635. 2008.

[6] L.A. Piegl, "Ten challenges in computer-aided design," Computer-Aided Design, vol. 37/4, pp. 461-470. 2005.

[7] A.J. Riel, Object-Oriented Design Heuristics. Addison-Wesley, Reading, MA 1996.

[8] D.F. Specht, "Probabilistic neural networks," Neural Networks, vol. 3/1, pp. 109-118. 1990.

[9] Y. Zeng, I. Horvath, "Fundamentals of next generation CAD/E systems," Computer-Aided Design, vol. 44/10, pp. 875-878. 2012.