THE CONCEPTION OF SIMULATION ENVIRONMENT FOR DEVELOPMENT AND TESTING OF DISTRIBUTED DIAGNOSTIC SYSTEMS[1]

Sebastian RZYDZIK

Silesian University of Technology, Institute of Fundamentals of Machinery Design
18A Konarskiego Str., 44-100 Gliwice, Poland; sebastian.rzydzik@polsl.pl

Summary

Machinery, equipment and system for monitoring and diagnostics are considered. It is assumed that the considered a set of machinery and equipment is distributed territorially and/or functionally (for example, several systems which are part of one machine). Machinery, equipment and system for monitoring and diagnostics that make up a distributed diagnostic system are called nodes of that system.

The article presents the concept of the environment to develop and testing distributed diagnostic systems. Connection with each nodes of distributed diagnostic system is implemented by the agent system and blackboard. Each agent represents the selected node of the distributed diagnostic system. Locally, the agent performs the task of communicating with the node that represents and the tasks of processing and collection of information contained in the received data from that node. Globally, the agent cooperates with other agents by performing tasks transmitting, receiving, processing and storing messages. Blackboard carries out the task of collecting the data common to the entire agent system. Communication with the blackboard is implemented through an agent who represents that.

Keywords: diagnostics of mechanical systems/machines/components, systems modeling and evaluation.

KONCEPCJA ŚRODOWISKA SYMULACYJNEGO DLA POTRZEB TWORZENIA I TESTOWANIA ROZPROSZONYCH SYSTEMÓW DIAGNOSTYCZNYCH

Streszczenie

Rozpatrywane są maszyny i urządzenia wraz z systemami monitoringu i diagnostyki tych maszyn. Zakłada się, że rozpatrywany zbiór diagnozowanych maszyn i urządzeń jest rozproszony terytorialnie i/lub funkcjonalnie (np. kilka układów wchodzących w skład jednej maszyny). Maszyny, urządzenia oraz systemy monitoringu i diagnostyki wchodzące w skład rozproszonego systemu diagnostycznego są nazywane węzłami tego systemu.

W artykule przedstawiono koncepcję budowy środowiska, w którym można tworzyć i testować rozproszone systemy diagnostyczne. Połączenie ze sobą węzłów rozproszonego systemu diagnostycznego realizowane jest przez system agentowy i tablicę ogłoszeń. Każdy agent reprezentuje wybrany węzeł rozproszonego systemu diagnostycznego. Lokalnie, agent realizuje zadanie komunikowania się z reprezentowanym węzłem rozproszonego systemu diagnostycznego oraz zadania przetwarzania i gromadzenia informacji zawartych w otrzymanych od reprezentowanego węzła danych. Globalnie, agent współpracuje z innymi agentami przez realizację zadań nadawania, odbierania, przetwarzania i gromadzenia wiadomości. Tablica ogłoszeń realizuje zadanie gromadzenia danych wspólnych dla całego systemu agentowego. Komunikacja z tablicą ogłoszeń jest realizowana za pośrednictwem reprezentującego ją agenta.

Słowa kluczowe: diagnostyka układów mechanicznych/maszyn/komponentów, modelowanie systemów.

[1] Described herein are selected results of study, supported partly from the budget of Research Task No. 4 implemented under the strategic program of The National Centre for Research and Development in Poland entitled "Advanced technologies of generating energy".

1. THE USE OF AGENT TECHNOLOGY

1.1. Introduction

According to the dictionary [5] agent (Lat. *agens*, *agentis*, "acting" comes from *agere* "work, do"), this is a person acting on behalf of an institution, company.

As a result of the development of programming methods, there are special programs that, properly configured, act on user behalf, relieving him of certain tasks or searching some information for him [6]. Thus came the term *software agent* [1], where software agent is an autonomous software entity capable of interacting in their environment (especially with other agents).

By connecting several agents are built *multi-agent system*. It is important that under such a system, agents cooperate in order to realize a certain task. Multi-agent systems can create multiple copies of the same agent. As a result of such an operation is performed the parallel processing. Is obtained then higher probability obtain an optimal results.

With a wide range of tools to design and develop multi-agent systems, such as: ZEUS, BOND, or Grasshopper, special attention was paid to the environment called JADE [4]. JADE (*Java Agent DEvelopment Framework*) is a platform that fully complies with the specification developed by FIPA foundation [3].

1.2. Examples of application

As part of the work conducted in Institute of Fundamentals of Machinery Design, multi-agent system used in the project DiaDyn [2].

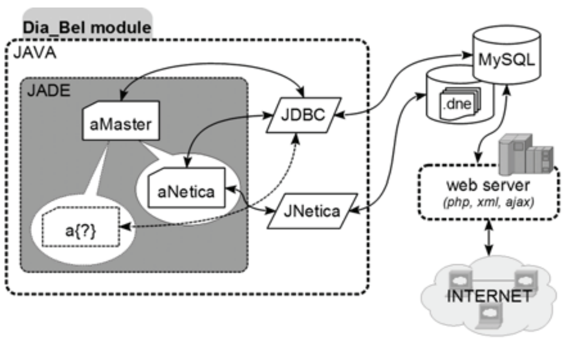

Fig. 1 The structure of the Dia_Bel module [11]

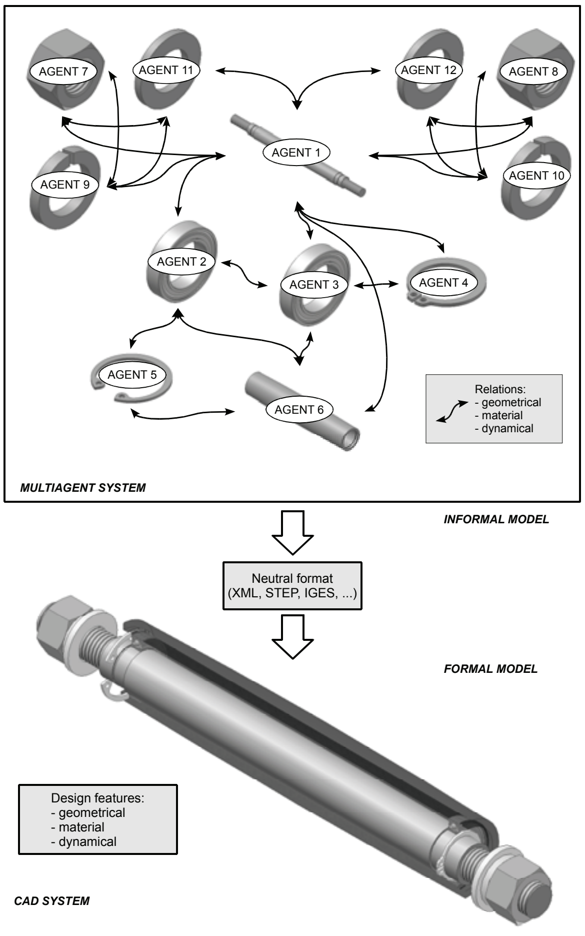

Fig. 2 Network of agents representing the actual object model as a set of elements. The transition from informal model to formal model [10]

DiaDyn is an open internet expert system. One of the modules of DiaDyn system is a module called Dia_Bel [11]. This module has been developed for the process of the approximate inference. Dia_Bel module is written in Java [7] and was to carry out various tasks (inference by different methods). For this reason it was decided to use the multi-agent system. It was assumed that each type of task (any method of inference), will be supported by a specialized agent.

The Fig. 1 shows the structure of the Dia_Bel module. *aMaster* agent creates *aNetica* agents and the other agents labeled as *a{?}* which support approximate inference tasks.

JADE multi-agent system was also used in the work related to aided process of the mechanical design [8][9][10]. It is proposed that each software agent is a model of a single part (Fig. 2). In order to obtain assembly of parts are combined agents, usually give a network structure. Such a connections structure should be regarded as a system of mutual design relations between the parts. The aim is to achieve a stable state of equilibrium between different types of design features and their values. The result of multi-agent system is a geometric model of the future product.

1.3. General assumptions

Possible situations:

1) At the output of the simulator of the object, data are not standardized. Need to develop a programs that will convert the output data format of the simulator of the object in the input data format of the blackboard. In addition, each program will communicate with the blackboard.
2) At the output of the simulator of the object, data are standardized. Just develop a one program that will communicate with the blackboard.

The best situation is the second situation. Imposition of the output data format of simulator of the object will reduce the number of needed to develop applications. On the other hand, most of the work required to adapt to the imposed format of the output data rests on the authors of the object simulators.

The basic definition of multi-agent system assumes that the some number of agents working together to solve defined problem. In the proposed concept of a simulation environment for development and testing of distributed diagnostic systems, used agent system has to solve the problem of efficient communication between nodes of the distributed system. Therefore, the role of agents is limited to tasks related to the process of communication – as the patterns of interaction between agents, are used mainly cooperation and coordination of communication tasks.

Assumed that the agent's name will consist of a small letter *a* and agent's type designation starting with a capital letter *a{?}* (*{?}* this is the type agent). Agent's type designation is written without spaces, small and capital letters in Latin characters and Arabic numerals.

2. THE CONCEPTION OF SIMULATION ENVIRONMENT

2.1. Structure of the agent system

On the Fig. 3 shows a diagram of overall structure of the agent system. It is assumed that it will be one host with blackboard application and several hosts with simulators of the objects, which will be write or read calculation results to/from the blackboard – a global database of the distributed system.

On the Fig. 4 shows a diagram of the structure of agent system from the perspective of the connection between the simulator of the object and blackboard. Hosts are connected together via a computer network (LAN or WLAN). In the case of the host with the simulator, next to the application to simulate the behavior of the selected real object, is running agent system that has access to certain directories on local disk.

For a host with blackboards, next to the software performing the functions of data collection, environmental agents are running, which acts as an intermediary between multiple simulators of the objects. Thanks to this access the database from the "outside" is limited.

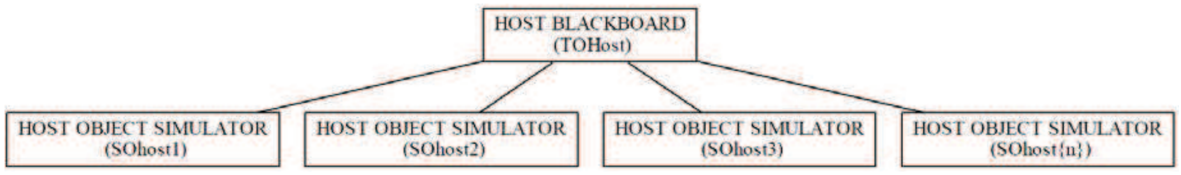

Fig. 3 Diagram of overall structure of the agent system

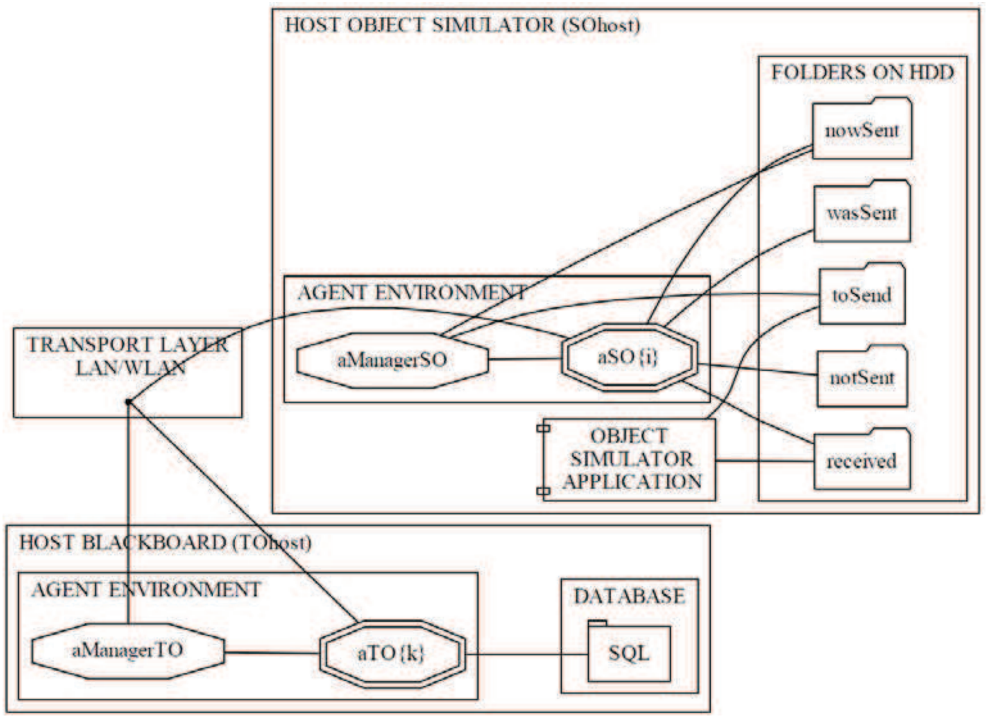

Fig. 4 Diagram of the structure of agent system

2.2. The list of types of agents

Below is a list of agent's types:

- *aManagerSO* – agent installed on the object simulator host, this agent is primarily responsible for:
  - to check are there any new data generated by the simulator of the object;
  - to verification the formal structure of the XML file;
  - to upgrade the status of the XML file, by modifying the value of the indicated timestamp;
  - to create an agent *aSO{i}*;
  - to delete an agent *aSO{i}*, when it finishes its task.
- *aSO{i}* – agent run by agent *aManagerSO* on object simulator host, *{i}* variable is set to name of the XML file that contains the data generated by the SO, this agent is primarily responsible for:
  - to connect to the agent *aManagerTO* on the side of the blackboard host;
  - to read the XML file and send it to the agent *aTO{k}* on the side of the blackboard host;
  - to upgrade the status of the XML file, by modifying the value of the indicated timestamp;
  - to save in a specified folder on local disk receiving from an agent *aTO{k}* a message that contains the XML structure.
- *aManagerTO* – agent on the side of the blackboard host, this agent is primarily responsible for:
  - to wait for a message from an agent *aSO{i}* on the side of the object simulator host;
  - to create an agent *aTO{k}*;
  - to delete an agent *aTO{k}*, when it finishes its task.
- *aTO{k}* – agent run by agent *aManagerTO* on the blackboard host, *{k}* variable is set to name of the XML file that contains the data generated by the SO, this agent is primarily responsible for:
  - to connect to the agent *aSO{i}* on the side of the object simulator host;
  - to receive a message sent by the agent *aSO{i}*,that contains the XML structure;
  - to process the information contained in the structure of XML and to generate and execute SQL queries;
  - to send SQL query results to the agent *aSO{i}*.

2.3. Data format

It is assumed that the data generated by the simulator of the object will be transferred in the form of files stored in XML format. The XML format allows for a unification of all files, regardless of the stored information and allows it to check the formal correctness of the stored content. XML file name must be unique across the distributed system and should be an identification. To generate the file name is proposed to use the MD5 algorithm

(*Message-Digest algorithm 5*). As an input parameter to a MD5 function we can specify a string which is the sum of the following components:

```
             XML_FILE_NAME =
  file creation time stamp (Unix time)
      + random string of 12 digits
```

A specific XML structure should include information about: XML file, performed simulations, simulated object and also variables and the vectors of values of these variables. Below is shown a general form of a specific XML structure:

```
<?xml version="1.0"
      encoding="ISO-8859-2"?>
<!DOCTYPE objSimData SYSTEM
      "_objSimData.dtd">
<file>
  ...
</file>
<simulation>
  ...
</simulation>
<object>
  ...
</object>
<quantities>
  <quantity>
    ...
  </quantity>
  <quantity>
    ...
  </quantity>
  ...
</quantities>
<vectors>
  <vector>
    ...
  </vector>
  <vector>
    ...
  </vector>
  ...
</vectors>
```

Below describes in detail what information should be included in the presented XML structure.

As information about the XML file must be included:

- version number of the XML structure;
- id of a data file (the same as the XML file name);
- information about who generated the file (e.g. author name);
- description of the data file (additional information about the XML file);
- timestamp (Unix time) to generate the file and save it in *toSend* folder;
- timestamp (Unix time) transfer the file to the *nowSent* folder (start the process to send the file content to the blackboard);
- timestamp (Unix time) transfer the file to the *wasSent* folder or to the *notSent* folder (complete the process of transfer the file to the blackboard – success or failure);
- timestamp (Unix time) queries the database (blackboard);
- timestamp (Unix time) to save in a *received* folder the file that contains data generated by the blackboard;
- status of data file, e.g.: 0 - waiting to send; read data from blackboard, 1 - waiting to send; save the data in blackboard, 2 – busy; sending process is still on, 3 - sent (saved in blackboard), 4 - unsent (unsaved in blackboard), 5 - another error;
- description of the status (additional information about the error, such as error code).

The proposed XML structure is as follows:

```
<file>
  <verXML></verXML>
  <file_id></file_id>
  <file_author></file_author>
  <file_desc></file_desc>
  <timestamp_create>
           </timestamp_create>
  <timestamp_sendstart>
           </timestamp_sendstart>
  <timestamp_sendstop>
           </timestamp_sendstop>
  <timestamp_query>
           </timestamp_query>
  <timestamp_receiv>
           </timestamp_receiv>
  <status></status>
  <ststus_desc></status_desc>
</file>
```

As information about the simulation process should be considered:

- id of the simulation;
- name of the simulation;
- description of the simulation (for more information about the simulation).

The proposed XML structure is as follows:

```
<simulation>
  <sim_id></sim_id>
  <sim_name></sim_name>
  <sim_desc></sim_desc>
</simulation>
```

As information about the simulated object should be considered:

- id of the object;
- name of the object;
- description of the object (for more information about the object);
- the manufacturer's name.

The proposed XML structure is as follows:

```
<object>
  <obj_id></obj_id>
  <obj_name></obj_name>
  <obj_desc></obj_desc>
  <obj_ manufacturer>
      </obj_ manufacturer>
</object>
```

As information about the values of variables should be considered:

- identifier of the variable;
- name of the volume;
- physical unit (if appointed size);
- description of the variable (for additional information about the variable);
- size (single value or a vector of values);
- id of the related variable (e.g. time series requires a two variables, such as time and speed).

The proposed XML structure is as follows:

```
<variables>
  <variable>
    <var_id></var_id>
    <var_name></var_name>
    <var_unit></var_unit>
    <var_desc></var_desc>
    <var_dim></var_dim>
    <var_related_id>
            </var_related_id>
  </variable>
  ...
</variables>
```

As information about the vector of variable values should be considered:

- id of the vector of values;
- identifier of the variable;
- value or values separated by semicolons (values can be either numeric values or the values of quality);
- description of the vector (for more information about the vector values).

The proposed XML structure is as follows:

```
<vectors>
  <vector>
    <vect_id></vect_id>
    <vect_var_id></vect_var_id>
    <vect_value></vect_value>
    <vect_desc></vect_desc>
  </vector>
  ...
</vectors>
```

2.4. The message encoding format

The exchange of messages between software agents, JADE uses a special language called *ACL* (*Agent Communication Language*, FIPA-ACL standard). As part of the language defines different types of messages sent, e.g. *inform*, *request*, *query* and *propose* [3][4]. The structure of the sample message is shown below (sign ":" precedes the message parameters).

```
(inform
  :sender agent1
  :receiver agent2
  :content (some information
         content)
  :language English
)
```

This message is a message sent by the sender *agent1* to the recipient *agent2* with the content of *some information content* stored in *English*.

Each message has: id of the sender (*sender*), id of the receiver (*receiver*), message content (*content*), language ID of the message encoding (*language*). The process of transmitting and receiving messages is asynchronous. Each agent, as part of its resources, has its own "mailbox". It was concluded that the use of language ACL facilitate transfer of XML files with data which are generated by the simulator of the object. For this purpose, set the message content as an XML file, and the language defined as XML, as follows:

```
ACLMessage msg =
 new ACLMessage(ACLMessage.INFORM);
msg.addReceiver(<id_recipient >);
msg.setLanguage("XML");
msg.setContent("
  <?xml version="1.0"
      encoding = "ISO-8859-2"?>
  <!DOCTYPE objSimData
      SYSTEM "_objSimData.dtd">
  <file> ... </file>
  <simulation> ... </simulation>
  <object> ... </object>
  <feature> ... </feature>
  <vector> ... </vector>");
send(msg);
```

2.5. The communications process

Fig. 5 shows a diagram of process of communication between the three agents: *aSO{i}*, *aManagerTO*, *aTO{k}*. According to previous assumptions, agents to communicate using the ACL language. Computer network (LAN/WLAN) is the medium of transmission between *Object Simulator (SO)* and *Blackboard (TO)*.

Sending XML data generated by the simulator of the object (SOHost) proceeds in five steps:

1) **REQUEST** After creating, the *aSO{i}* agent is ready to send the XML file contents to the blackboard. For this purpose sends to the *aManagerTO* agent request to create a transmission channel.
2) **<<create>> NewAgent()** The task of creating a new *aTO{k}* agent. This task is performed by the *aManagerTO* agent without the use of language ACL and has an indirect, but a key influence on the communication process.
3) **AGREE/NOT-UNDERSTOOD/REFUSE** After creating, *aTO{k}* agent sends to *aSO{i}* agent a response message: permission to establish a transmission channel, the information that the request is unclear or lack of consent to establish transmission.
4) **REQUEST(XML File)** After receiving permission to the transmission of data, the *aSO{i}* agent sends to the *aTO{k}* agent a message containing data stored as XML, otherwise it returns an error to an *aManagerSO* agent and ends the task.
5) **AGREE/REFUSE** An *aTO{k}* agent tries queried database (*Blackboard*). Database query results has an influence on the content of messages sent to an *aSO{i}* agent; possible answers are: `ok` or `error`. In the case when answer is `ok`, *aTO{k}* agent sent to *aSO{i}* agent structure of XML.

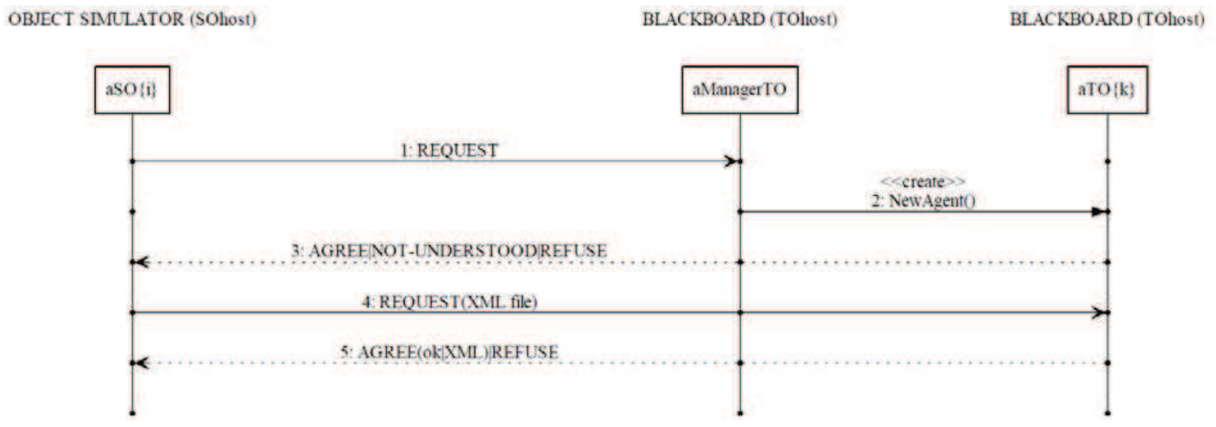

Fig. 5 Diagram of communication between agents using the ACL language

3. SUMMARY

Ultimately, the described system will be installed on virtual machines (virtualized computing resources). An analysis of the structure of the system shows that the most sensitive element is a blackboard. In order to increase the reliability of this system is planned to use the cloning mechanism for host and database.

The proposed solution allows the combination of simulators of objects developed in different research groups, without having to send the applications. Among the many benefits include: no necessity distribute new versions of the simulator and remote start of the calculation without involving their IT resources. The exchange of messages between hosts is carried out according to fixed rules in a heterogeneous environment.

Using XML allows a simple extension of the proposed structure with new elements, and thus provide additional information.

REFERENCES

[1] Adamczewski P.: *Słownik informatycz*ny, Helion, Gliwice, 2005.
[2] Cholewa W. (Eds.): *Szkieletowy system doradczy DIADYN*, vol. 137 serii Zeszyty, Silesian University of Technology, Department of Fundamentals of Machinery, Gliwice, 2008.
[3] FIPA - The Foundation for Intelligent Physical Agents, http://www.fipa.org
[4] JADE – Java Agent DEvelopment Framework, http://jade.tilab.com
[5] Kamińska-Szmaj I. (Eds.): *Milenijny Słownik Wyrazów Obcych*, Europa, Wrocław, 2002.
[6] Nwana H., Ndumu D.: *A perspective on software agents research*. The Knowledge Engineering Review, 14(0):125–142, 1999.
[7] Oracle Corporation: Java Platform, Standard Edition (Java SE). http://www.oracle.com/technetwork/java/javase/downloads/index.html

[8] Rzydzik S.: *Application of multi-agent system in the design process*, in: M. Wasilczuk: Podstawy Konstrukcji Maszyn – kierunki badań i rozwoju. Tom I. Gdańsk, 2011, s. 163-171.
[9] Rzydzik S., Skarka W.: *Formal model for the purpose of generative modeling based on multi-agent system*. Symposium on Methods of Artificial Intelligence, Gliwice, November 2009.
[10] Rzydzik S., Skarka W.: *Multiagent system for aiding designing process*. New World Situation:New Directions In Concurrent Engeneering, vol. Proceedings of the 17th ISPE International Conference on Councurrent Engeneering, Kraków, 6-10 September 2010, Springer-Verlag, 2010.
[11] Rzydzik S.: *Wnioskowanie w sieci stwierdzeń zorganizowanej jako sieć bayesowska*, in: W. Cholewa (Eds.), Szkieletowy system doradczy DIADYN, vol. 137 serii Zeszyty, Silesian University of Technology, Department of Fundamentals of Machinery, Gliwice, 2008, pp. 67-82.

Sebastian RZYDZIK is an Assistant Professor at the Silesian Technical University. He is interested in artificial intelligence techniques and their applications to designing of machinery (e.g. CAx systems) and to technical diagnostics.