

Review of Computer Engineering Research

2020 Vol. 7, No. 1, pp. 1-11

ISSN(e): 2410-9142


ISSN(p): 2412-4281

DOI: 10.18488/journal.76.2020.71.1.11

© 2020 Conscientia Beam. All Rights Reserved.



EXPLORATION OF INTELLECTUAL SOFTWARE SYSTEMS AND DEVELOPMENT OF CONCEPTUAL MODEL

 **Shafagat Mahmudova**

SE Department, Institute of Information Technology of ANAS, Baku, Azerbaijan.

Email: shafagat_57@mail.ru



ABSTRACT

Article History

Received: 22 October 2019

Revised: 25 November 2019

Accepted: 30 December 2019

Published: 13 February 2020

Keywords

Artificial intelligence

Intelligent system

Classification

Direction

Conceptual model.

This article explores the field of artificial intelligence and intellectual systems. The structure, types and classification of the intellectual system are studied. The intellectual system is a technical or software system that is capable of solving the problems which are specific and creative in a particular subject area, so that knowledge is stored in the memory of such systems. Intelligent System is a technical or software system that competently solves the problems that are relevant to a particular subject area and that knowledge is stored in such systems. Intelligent systems are studied by a team of researchers called "artificial intelligence". The article provides the differences between the ordinary system and the intellectual system. The system is intellectual when it not only changes the parameters of information access, but also changes its behavior itself depending on the system capability. Intelligent system in decision-making technologies is an intellectual information-computing system that solves the problems without human involvement. The classification of intellectual systems is analyzed in the article. A conceptual model for intelligent systems has been developed. The intellectual systems themselves may also have some gaps. In future, the development of new intelligence systems and improvement of existing ones will benefit the economy even more.

Contribution/Originality: The intelligent system structure, types of program errors, classification of programming errors, research areas of intellectual systems was studied. Based on the study, a conceptual model for intellectual systems has been developed.

1. INTRODUCTION

Artificial intelligence is a universal field of science. The application areas of artificial intelligence are diverse. They are actively adapting to and changing other sciences when solving any problem.

In 1956, American scientist John McCarthy first used the term artificial intelligence [1]. Intelligent System (IS) is a technical or software system that competently solves the problems that are relevant to a particular subject area and that knowledge is stored in such systems. Intelligent systems are studied by a team of researchers called "artificial intelligence."

Is an automated knowledge-based system or a set of software, linguistic, and logic-based tools for realizing a key problem. It is an implementation of a person's information retrieval in a natural language in a dialogue mode.

The system is called intellectual when it changes not only the parameters of access to information, but also its behavior itself depending on the capability of the system [2].

In addition, the systems are intellectual when the logical processing of information overweighs the computing when solving the complex problems. Thus, any information system that addresses the intellectual problems or uses artificial intelligence methods is called intellectual.

Intelligent Information System (IIS) is a set of software, language and logic tools to support the human activities for the key task implementation and to search the information in the natural language in a dialogue mode.

Intelligent systems in decision-making technologies are intelligent information-computing systems that are capable to solve the problems without human involvement [3, 4].

Logic programming (Prolog, List, etc.) was previously used for working with IS, nevertheless, different procedure languages are used now.

2. ABOUT THE INTELLIGENT SYSTEM STRUCTURE

The structure of the intelligent system includes three main units:

- Knowledge base.
- Solution acquisition mechanism.
- Intelligent interface.

The following aspects are used to ensure the functioning of the intelligent system:

- Mathematical.
- Linguistic.
- Information.
- Semantic.
- Software.
- Technical.
- Technological.
- Staffing.

The classification of the tasks addressed by IS includes:

Data interpretation is one of the traditional tasks for expert systems. It refers to the process of interpreting the meaning of the information needed.

Diagnostics refers to the process of detecting a malfunction in a particular system.

Monitoring. The main task of monitoring is to continuously visualize the data in real time and to ensure the specified parameters to exceed the certain limits.

Design refers to the creation of the specifications for building the objects with predefined properties. The specifications include all the necessary documents (drawing, explanatory notes, etc.).

Forecasting allows for predicting the outcome of the events based on the analysis of existing data.

Planning involves the development of action plans associated with the facilities that can perform the certain functions.

Training refers to teaching any subject using a computer.

Control refers to the work of an organized system that supports a certain mode of action. Such specifications are accordingly controlled by the complex systems.

Decision Support is defined as a set of procedures that provide the necessary information and recommendations to facilitate the decision-making process.

The types of intelligent systems are provided in [Figure 1](#).

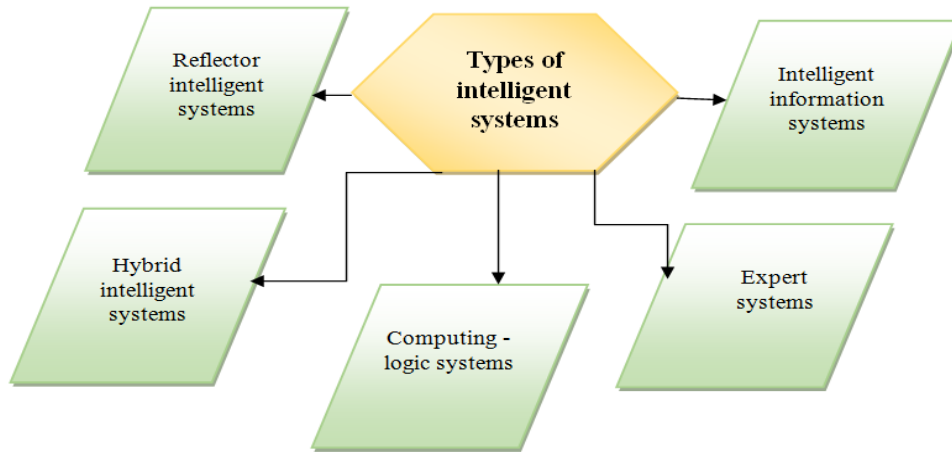


Figure-1. Types of intelligent systems.

There are certain tools that can be used to create intelligent systems. These technologies may include:

- Rule-based explanations.
- Decision trees.
- Bayesian network.
- Fuzzy logic and soft computing.
- Inductive logic programming.

The Bayesian network is an example of variability and a probability model of a graph with their probable Bayesian dependence. For example, the Bayesian network can be used to calculate the probability of a patient's illness with or without the presence of numerous symptoms based on the data about the relationship between the symptoms and disease. The mathematical apparatus of the Bayesian networks was developed by Judea Pearl, the American scholar and Turing Prize winner (2011).

Inductive logic programming is a machine learning unit that uses logical programming as a form for presenting examples. By acquiring a number of examples, which are presented as already known background descriptions and logical bases of facts, IS can provide a logical program as a hypothesis that explain all positive examples.

Application areas may include:

- Business & Management (pricing, workforce, products, strategy, etc.)
- Engineering (product quality control).
- Finance (credit and loans).
- Healthcare (medicines, treatment types, diagnostics).
- Environmental protection, etc.

Examples of artificial intelligence systems are provided in [Table 1](#).

Table-1. Examples of artificial intelligence systems.

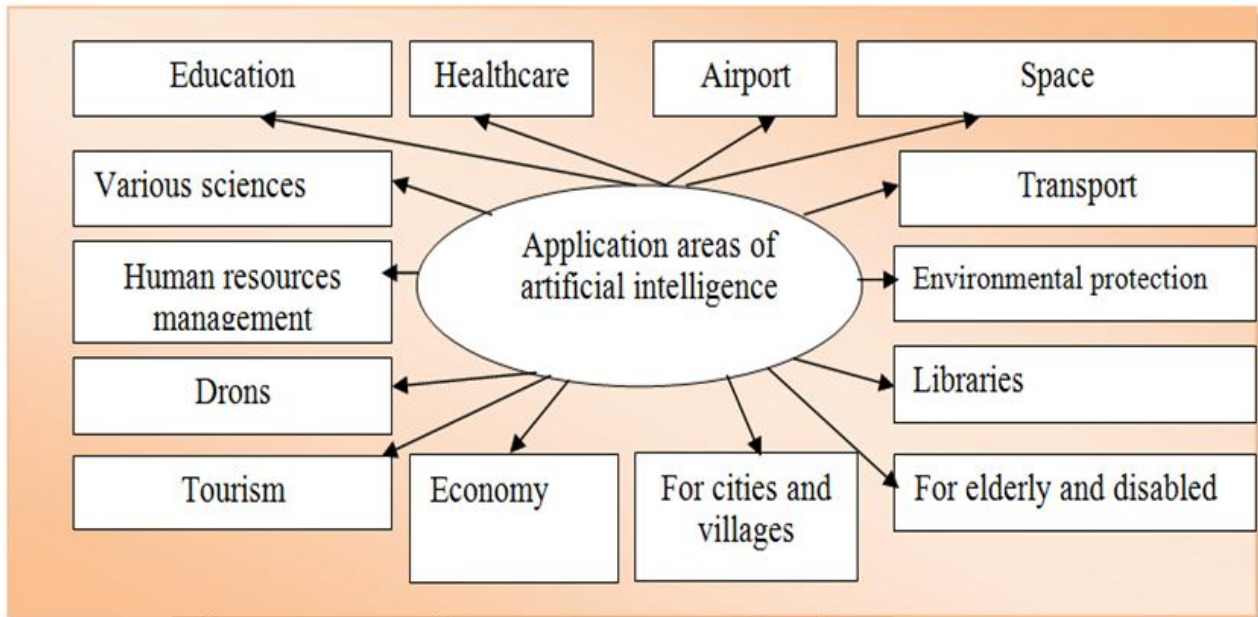
System	Artificial Intelligence field	Definition
ROBIN	Strategic Games	Intelligent programs for chess-type games
ALICE	Information retrieval engines	Search for anthology or deduction database
PEACE	Problem solving	Solving problems for synthesis and electro-scheme analysis
RITA	Knowledge Presentation	Modeling of Concepts development (Improvement)
MUSCADET	Proof of Theorems	Proof of 'Pure' Mathematics theorems
ARGOS-II	Robotics Robototexnika	Robot simulation of decision-making

The largest research centers in the field of artificial intelligence are provided in [Table 2](#).

Table-2. Largest research centers in the field of artificial intelligence.

Countries	Enterprise
USA	Massachusetts Institute of Technology Institute of Machine Intelligence
Germany	German Research Center for Artificial Intelligence
Japan	National Institute of Advanced Industrial Science and Technology
Russia	Scientific Council on Methodology of Artificial Intelligence of the Russian Academy of Sciences
India	Indian Institute of Technology in Madras

The application areas of artificial intelligence are provided in [Figure 2](#).

**Figure-2.** Application areas of artificial intelligence.

The intelligent system is capable of collecting and analyzing data and sharing them with other systems. Other criteria of the intelligent systems include the ability to train based on the personal experience, and other capabilities related to security, connectivity, adjusting to current data, and remote monitoring and management.

3. LITERATURE REVIEW

Some of the works in this area are reviewed and analyzed below [5]. Analyzes various features of the program's decision-making quality. Despite numerous studies, a comprehensive criterion for software quality control is still only available at an informal level. The quality of the program is defined to be a manageable indicator. The quality features of intellectual systems are of particular importance. The action plan is checked through the Boolean functions for which the graph of cause and effect relationships are drawn and presented in logic diagrams. A plan can be created at any stage of the software life cycle.

Soliman [6] analyzes open code software field at the terminological level and seeks to understand the ideological value of concepts. Due to Stallman and Raymond, new terms have emerged to represent the development patterns chosen by FLOSS (open source software) and to explain common principles.

The Intellectual Property Rights of Computer Software is the right of access to software creators that are not limited by time and space. [7] compares two population groups to determine the level of awareness and respect for intellectual property rights in software. 1) 96 software engineers who are IEEE Union members 2) 386 students are randomly selected. The results are analyzed using special software and the reliability of the results is tested.

Comparing the results, it is concluded that the first population is more aware of these cases. Subsequently, a model for the protection of intellectual property rights in software is introduced to reduce the problems.

In recent years, program project managers have compared the reports completed by the project participants to identify significant deviations in the workflow and manage the project risk. Chang, et al. [8] proposes an intellectual system for monitoring and managing software projects using natural language techniques to recognize the text reports to better evaluate the project implementation levels.

In the ever-growing competitive world, IT systems of large corporations traditionally require a lot of effort and expense to maintain and manage the system engineering called IT service management. With the development of artificial intelligence, IT service management can be more beneficial and effective in the incidents' managing. AI can not only assist in the resolution of an incident (for example, an incident creation, detection, response, resolution and completion), but also ensure the prevention of the incident repeat. Prasad and Mukkada [9] focuses on this feature. This includes the determination of the knowledge management and machine learning algorithm.

4. COMPUTER DISRUPTIONS

There are a number of computer disruptions occur when the programs run. Disruptions are a computer event that stops one or more programs or the entire operating system to respond to the user's actions. In this case, if the monitor restricts an image provided by the program, unlike the performance error, the following message will appear on the screen:

- Erroneous completion of cycles.
- Mutual blocking of processes in multi-task CS.
- Misuse of resources for programs, etc.

An infinite cycle of programming is a period when the exit condition out of it is never provided. The infinite cycle of programs is estimated to perform an endless cycle. Deadlock is a situation in a multi-task environment or DBMS. Thus, the resources are in constant swooping mode during the program execution and none of them can continue their implementation [10].

Thrashing occur when a virtual memory subsystem of a computer is in constant swooping mode, exchanging data on drive and memory which negatively affects the program performance. This may cause delays or computer operation disruptions. It can last for an extended period of time until the causes are eliminated.

4.1. Apparatus

- Drivers' errors.
- Violation of temperature mode.
- Mechanical damages (cable disconnection, cracks, etc.).
- Chemical damages.
- Incorrect voltage on computer or its components.
- Fork bombs.
- Disruption problems.
- Breakpoint.
- Death screen.
- Linux oops.

Fork-bombs are malicious or incorrectly written programs that produce own endless replications, and so on. Disruption problem (or disruption of mechanism) is one of the main problems in the theory of algorithms, as it is formally put in this way:

The procedures are described and its first input data is given, and with this data, it is required to determine when the process will be completed. An alternative to this is that it always runs continuously.

Alan Turing in 1936 proved that the problem of disruptions was not solvable in Turing's machine. In other words, there is no common algorithm to solve this problem.

The program breakpoint mechanism is a deliberate interruption of the program execution; thus, the regulator calls are executed. After switching to the regulator, a programmer can examine the program status (log, memory, processor registers, stacks, etc.), for example, whether the program runs properly. Unlike full break, the regulator determines whether the program is completed, or if so, whether it can continue its operation from the breakpoint.

In practice, a breakpoint mechanism is defined by one or more conditions, resulting in program interruptions. Based on the instruction breakpoint given in the program, the switching time and the conditional release mechanism are often used. Another condition for break mechanism is the operation of reading, writing or changing the specified field or the range of memory (data breakpoint or watchpoint) fields.

Most processors have hardware support for breakpoint mechanism (often for instruction breakpoint and watchpoint only). In the absence of such hardware support, the regulators use the program points of break mechanism.

Death screen is the slang name of an image that occurs during the errors in the operating system. In case of death screen, a computer restart is required, and unsaved (unprotected) issues are lost.

Sometimes, self-checking is enabled within the operating system kernel. This means that the kernel or drivers are running in crash mode, and the problem is solved only through reloading. The device, which runs without human intervention, is reloaded without micro-controlling queries. However, as a rule, personal computers are often equipped with a monitor and enable a relatively inexperienced person (user or system administrator) to replace the details of hardware or software. To point out the cause of the error, the OS releases any details related to it. For example: which self-control does not work, which kernel module is implemented, in which case it is stacked, etc. This information is often displayed in text mode.

For the devices that are not available to the user (players, mobile phones, gaming consoles), the "death screen" can be an image occurred when the broken device is enabled. Oops is a kernel function of Linux operating system when Linux deviates. In case of Oops, the kernel generates a writing to describe an error. After frequent oops calls, a kernel panic is called to restart the system.

Kernel panic is data about a critical error of the operating system, since the operating system cannot continue to function after this error. This term is often used in the UNIX operating system. After Oops, some system resources may be unavailable. Kernel panic mainly occurs when the systems try to use unavailable resources.

Examining the contents of the System.map file can help to identify the causes of Oops. System.map is a symbolic table of features and procedures used by the Linux operating system. This table lists the names and addresses of variables and functions in computer memory. This table is very useful in case of Kernel panic or Linux oops. System.map appears when compiling the kernel [11].

5. TYPES OF PROGRAM ERRORS

Software errors often occur due to unexpected behavior in software or system. Many program errors, which may be in the source code or design, are caused by programmers when developing the programs. In addition, some errors are caused by poor performance of the tools, such as the compiler's error code. A program with a large number of errors is called unstable [12].

There are various types of software errors that can occur during the software development phase. Each programmer should be aware of them. The most common errors may include the followings:

- Logical errors.
- Syntax errors.
- Compilation errors.
- Execution errors.

- Arithmetic error.
- Interaction errors and so on.

Logical error is probably the most serious error. Accordingly, any software written in any language is compiled and operates properly, however it provides the wrong result, since the main flaw is in the programming logic. This is related to the fault in the program's algorithm. The logic structure of the entire program is flawed. To find a solution to these errors, the main program algorithm is required to be changed.

Syntax error. Each programming language, such as C, Java, Perl, and Python, has its own syntax for writing code. If the programmer ignores the "grammatical" specifications of the programming language, then the syntax errors will occur. Such errors are easily eliminated during compilation.

Compilation error. Compilation is a process in which high-level software is translated into machine language. Many errors, including syntax errors may occur at this stage. Sometimes the word sequence of the source code can be flawless, however, in this case, an error may occur. This may depend on the compiler's own problems. These errors are corrected during the development phase of the program.

Execution errors. The program code is successfully compiled and an execution file is created. A program can be enabled to check the performance. Errors can occur as a result of an accident or a lack of resources. The programmer has to preview the actual conditions of the program being executed. It can be adjusted going back to the coding stage.

Arithmetic errors. Many programs use numeric variables, and the program's algorithm may perform several mathematical calculations. Arithmetic errors may occur when a computer is unable to cope with the problems, such as a "division by zero" or infinite result. This is a logical error which can only be corrected by changing the program algorithm.

Resource errors. If the value of a variable exceeds the maximum acceptable value, it will cause a resource error. It may include board overflow, use of non-initialized variables, violation of access rights, and other common mistakes.

Interaction errors can be caused by a mismatch between the hardware interface and the application interface. The interface error in web applications may be as the result of misuse of the web protocol.

The errors may also vary depending on the programming languages. The errors reduce the reliability and security of programs.

6. CLASSIFICATION OF PROGRAMMING ERRORS

Software errors for microprocessors can be categorized as follows [13]:

1. **Movement of operands or their parts.** Typical errors include the placement of the operators pointing the commands to the source and destination, and the specification of the format in storing 16-bit values, and the change of directions during subtractions and comparisons.
2. **Improper use of flags.** Typical errors include:
 - Use of the wrong flags to be checked in this case (for example, a pointing flag instead of a portable flag).
 - Conditional switch after the commands that do not affect the flag.
 - Inversion of switching conditions (especially when using a zero flag).
 - Proper conditional switching in case of flag equation and accidental change before conditional switch.
1. **Confusing the registrations and registration pair.** A typical error includes working with the registration (B, D or H) instead of pair registration with the same name.
2. **Confusing the addresses and information.** Typical errors include the use of direct addressing instead of addressing, or vice versa, and confusing the registration with the memory records specified by the registration pair.
3. **Using the wrong formats.** Typical errors include the use of binary or BCD format and the use of binary and hexadecimal codes instead of ASCII.

Incorrect work with arrays. The overall error goes beyond the array.

4. Ignoring the sensitive effects. Typical errors include the use of a single battery, multiple registers, a stack of pointers, flags, or memory cells not considering the effect of commands. Many errors are based on the commands that give unexpected, covered or indirect results.

5. Errors occurred through the definition of the prerequisites for a particular application or computer as a whole. Many programs require enabling the addresses, notes, flags and counters for temporary storage, and so forth. The micro-computer requires enabling all cells distributed in RAM (addresses and counters in particular) as a whole.

6. Inappropriate program organization. Typical errors include throwing or repeating start sections, and loss of the addresses of incorrectly changing registers in counters, and loss of intermediate or final results.

The source of the common errors is a conflict between a user and system software. A simple example of such a conflict is the attempt to store the user program data in a system program memory. In this case, the data required for the user program changes while system program is running.

More complex sources of the conflict are linked to the disruption systems. System programs are required to use the same resources as the user programs. At the same time, the system programs often maintain and restore the application environment in which the user programs are enabled; however this often results in very difficult or unexpected consequences.

7. RESEARCH AREAS OF INTELLECTUAL SYSTEMS

Research areas of the intelligent systems mainly include three directions. The first research area focuses on the structure and mechanisms of the human brain, and the ultimate goal is to uncover the ambiguities of thinking. The necessary stages of research in this area include the creation of intellectual activity models based on psychophysiological data.

The second research area focuses on the artificial intelligence system. It comprises modeling of intellectual activity using computers. The goal of this research is to create software that allows some intellectual problems to be solved in the same way as a human.

The third research area focuses on the creation of human-machine or interactive and intellectual systems. The most important problem in these studies is the organization of a flawless semantic dialogue between a human being and a system.

8. CLASSIFICATION OF INTELLECTUAL SYSTEMS

The classification of intellectual systems includes the following sections:

- Development of communication skills.
- Communication skills of IS.
- Ability to solve complex poorly compiled problems.
- Self-learning ability.
- Knowledge of solving the specific situations based on the acquired experience.
- Adaptation - the system ability to develop in accordance with the objective changes in the problem area model.
- Advanced communication skills.

Intelligent databases differ from ordinary databases for the selection options of the data that cannot be stored publicly but is available in the database.

Intellectual information technology (IIT) is an information technology that helps a person accelerate the analysis of political, economic, social and technical situations, as well as the synthesis of managerial decisions [14].

In addition, the methods used are not required to be logically consistent or replicate the human thinking processes. The use of ITI in practice implies the consideration of the peculiarities of the problem area which can be characterized by the following set of features:

- Quality and prompt decision-making.
- Fuzzy goals and institutional boundaries.
- Multiple subjects involved in problem solving.
- Randomness, variability and quantity of environmental behavior.
- Multiple interactive factors.
- Poorly formulated, unique, non-stereotypical situations.
- Confidentiality, confidential data.
- Failure to realize plans, importance of minor actions.
- Paradoxical logic of solutions, etc.

9. DEVELOPMENT OF A CONCEPTUAL MODEL FOR INTELLECTUAL SYSTEMS

A conceptual model is a model represented by different concepts and the relationships between them that define the semantic structure of any field and its specific objects.

The conceptual model is a structure of the modeled system, the characteristics of its elements and the cause and effect relationship and is essential for the achievement of the modeling goal.

The conceptual model of the intellectual information system includes the following objects:

1. Structure.
2. Aspects.
3. Areas.
4. Types.
5. Tools.
6. Classification.
7. Research areas.
8. Errors.

The conceptual model of the intellectual information system is provided in [Figure 3](#).

10. CONCLUSION

When reviewing the intellectual information systems in terms of problem solving, they may include the management systems and auxiliary systems, computer linguistic systems, recognition systems, game systems and intellectual information systems. Each of them is important. The intellectual systems themselves may also have some gaps. Strong intellectual systems are required to overcome these gaps. Intellectual information systems in the economy perform effective processing of big data sets providing information support to the managers of organizations for decision-making [15]. In future, the development of new effective systems and improvement of existing ones will benefit the economy even more.

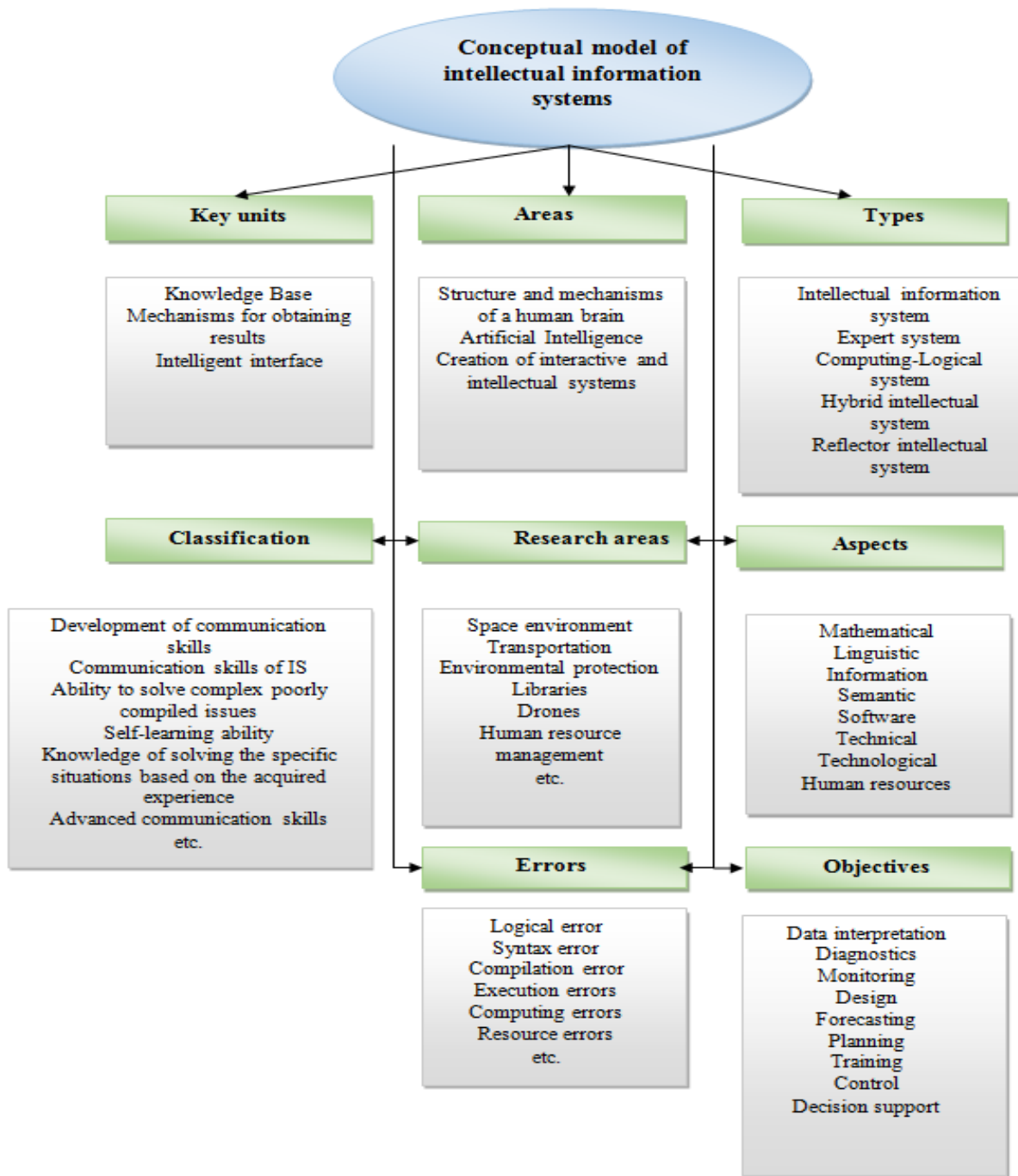


Figure-3. Conceptual model of the intellectual information system.

Funding: This study received no specific financial support.

Competing Interests: The author declares that there are no conflicts of interests regarding the publication of this paper.

REFERENCES

- [1] W. Lawless, R. Mittu, and S. Russell, *Artificial intelligence for the internet of everything*. Amsterdam: Elsevier Inc, 2019.
- [2] A. N. Averkin, M. G. Haase-Rapoport, and D. A. Pospelov, *Explanatory dictionary of artificial intelligence*. Moscow: Radio and Communications, 1992.
- [3] G. A. Lisiev, *Decision support technologies*. Moscow: FLINT, 2011.
- [4] P. Zaraté and S. Liu, "A new trend for knowledge-based decision support systems design," *International Journal of Information and Decision Sciences*, vol. 8, pp. 305-324, 2016. Available at: <https://doi.org/10.1504/ijids.2016.078586>.

- [5] O. Dolinina, V. Kushnikov, V. Pechenkin, and A. Rezhikov, "The way of quality management of the decision making software systems development," *Software Engineering and Algorithms in Intelligent Systems*, vol. 763, pp. 99-98, 2019. Available at: https://doi.org/10.1007/978-3-319-91186-1_11.
- [6] L. T. Soliman, "Ideological terminology: Between software and intellectual property," *Studies University Babes-Bolyai-Philolog*, vol. 62, pp. 235-244, 2017.
- [7] E. Sargolzaei and F. Fateme, "Examining software intellectual property rights," *International Journal of Advanced Computer Science and Applications*, vol. 8, pp. 594-600, 2017.
- [8] Y. C. Chang, C. W. Shih, and W. L. Hsu, "Entailment-based intelligent system for software project monitoring and control," *IEEE Systems Journal*, vol. 12, pp. 216-227, 2018. Available at: <https://doi.org/10.1109/JSYST.2016.2563463>.
- [9] R. T. Prasad and J. J. Mukkada, "Intelligent autonomous systems for software engineering - an example," presented at the 2018 International Conference on Intelligent Autonomous Systems (ICOIAS), IEEE, 2018.
- [10] J. D. Dumas, *Computer architecture: Fundamentals and principles of computer design*. Florida: CRC Press, 2005.
- [11] A. Vasudevan, *The Linux Kernel HOWTO, system.map*. Washington: Amazon, 2003.
- [12] R. G. Baldwin, "Programming fundamentals: Types of errors," pp. 120. Retrieved from: <http://www.dickbaldwin.com/Cosc1315/Pf00120.htm>, 2007.
- [13] A. Y. Gerasimov, L. V. Kruglov, M. Ermakov, and S. P. Vartanov, "An approach to reachability determination for static analysis defects with the help of dynamic symbolic execution," *Programming and Computer Software*, vol. 44, pp. 467-475, 2018. Available at: <https://doi.org/10.1134/s0361768818060051>.
- [14] P. M. Norling, J. P. Herring, W. A. Rosenkrans Jr, M. Stellpflug, and S. B. Kaufman, "Putting competitive technology intelligence to work," *Research-Technology Management*, vol. 43, pp. 23-28, 2000. Available at: <https://doi.org/10.1080/08956308.2000.11671377>.
- [15] S. I. Makarenko, *Intelligent information systems: A training manual*. Stavropol: Siberian State University, 2009.

Views and opinions expressed in this article are the views and opinions of the author(s), Review of Computer Engineering Research shall not be responsible or answerable for any loss, damage or liability etc. caused in relation to/arising out of the use of the content.