Advances in Intelligent Systems Research, volume 174

Proceedings of the 8th Scientific Conference on Information Technologies for Intelligent Decision Making Support (ITIDS 2020)

Ontological Approach to Viable Decision Support Services Development

Valeriya Gribova
Institute of Automation and Control Processes Far Eastern Branch of Russian Academy of Sciences
Vladivostok, Russia
gribova@iacp.dvo.ru

Elena Shalfeeva
Institute of Automation and Control Processes Far Eastern Branch of Russian Academy of Sciences
Vladivostok, Russia
shalf@iacp.dvo.ru

Philip Moskalenko*
Institute of Automation and Control Processes Far Eastern Branch of Russian Academy of Sciences
Vladivostok, Russia
philipmm@iacp.dvo.ru

Vadim Timchenko
Institute of Automation and Control Processes Far Eastern Branch of Russian Academy of Sciences
Vladivostok, Russia
vadim@iacp.dvo.ru

***Abstract*—The concept and the technology for creation and maintenance of decision support services based on knowledge bases are described. The ontological approach for knowledge, data and solver formation, along with the principle of reusability and extensible tools provide the viability for such services. An important issue is a unified representation and storage for service components of all types – both information and software ones. It allows to involve experts and knowledge engineers into development process on par with programmers and to use alike tools for creation of service components. Another significant requirement is a multi-user access for tools and components, preferably a cloud one – over the Internet. This eases the control of service development and allows to store and update components with no need of redistribution afterwards. The implementation of the described technology and tools on the IACPaaS cloud platform is presented with examples of medical knowledge portal and its fractions.**

Keywords—decision support service, knowledge base, ontological approach, viable service, intelligent system

I. Introduction

Development of methods for creation of viable intelligent system that provide specialists with useful advice on the basis of extensive knowledge bases is an urgent task [1-5].

An intelligent software system (ISS) can be considered as consisting of a problem solver, user interface, knowledge base and, possibly, databases with source and resulting data. Various formalisms are used to represent these components by specialized programming languages and shells.

It is known that a declarative representation, in contrast to an imperative one, has a number of advantages: simpler software creation, easier program code understanding, and, thus, modification [6-9]. At the same time, a unified representation of heterogeneous components of ISS is aimed at reducing the complexity of its development and maintenance.

Taking into account the above considerations, it is proposed to represent all ISS components (knowledge and data bases, a problem solver and a user interface) in a single unified format, providing common principles for their formation, access and modification. Semantic networks (more precisely, hierarchical conceptual digraphs) are chosen as this format.

The article describes the model of knowledge and data representation, ISS service components, tools and technology for its development.

II. Knowledge And Data representation Model

We propose an approach based on hierarchical conceptual digraphs to provide a clear representation of knowledge for development participants. The knowledge structure is clearly represented at the initial stage of ISS creation. Necessary interested participants are involved in its discussion as the conceptual digraph is understandable to them.

Developers are offered a universal problem-independent metalanguage for the specification of ontologies – languages for representing knowledge and their contextual conditions. Ontology-based knowledge and data bases are presented as hierarchical semantic networks [10].

A metalanguage is a pair: a graph of concepts (in terms of which ontologies of domain knowledge and data are formed) and graph labeling. The latter allows one to set restrictions on the structure and rules for generating knowledge and data bases by the ontology [10, 11].

The structure of knowledge together with ontological agreements is the ontology of knowledge. It defines rules for interpreting data and knowledge in the decision-making process, it is also the basis for the flexible interface of the knowledge editor. Ontological agreements should be known to developers of problem solving algorithms.

The ontology for solving an intelligent problem includes: an ontology of knowledge about the relationship of essential terms, an ontology of reality (input data / facts and expected solutions), limitations for their interpretation and ontological agreements on the rules of correlation of facts and knowledge.

The defined syntax of the ontology can and should take into account the organization of knowledge and data of a specific domain. Due to this, the intelligent complexity of the formation and maintenance of knowledge bases is reduced to an acceptable level. Ontologies are also used to formulate declarative descriptions of ISS components.

When creating thematic knowledge portals, it is important that the information being processed is equally understood by all members of the community. The definition of a base of terms (and observations) is necessary at the initial stage. Their uniform representation (in the form of hierarchical semantic networks [10, 11]) is provided under the control of “The ontology of the terminology and observation base”. This ontology allows one to define the nomenclature of terms, observation groups, groups of signs, signs, their elements or characteristics, their meanings. The concept of “synonyms” is introduced for terms, signs and qualitative (string) values of signs.

For example, “The medical terminology base” (for gastroenterology, pulmonology and other profiles) contains the necessary and sufficient sets of features, objective and instrumental examinations, laboratory tests – all divided into the corresponding observation groups (and subgroups). The group of signs “Complaints” includes a description of hundreds of signs and symptoms with a detailed representation of their various characteristics and values that are found in most diseases.

The results of the solution (with explanation) formed by the service (a list of rejected hypotheses, a list of hypotheses, which will require additional information to confirm, and, possibly, a confirmed hypothesis) can be visualized by the standard user interface. The explanation structure (which is considered to be part of the domain ontology) is created to hold the full explanation of the solution and considered hypotheses.

III. The Formation Of Declarative Components Of Decision Support Services

The proposed basic toolkit includes editors for creating ontologies and the generator of editors of service components.

When forming ontological bases, the toolkit maintains a correspondence between them and their ontology. Further, when changing the ontology, the correspondence between the knowledge and data bases and their ontologies is not broken: all the knowledge and data bases are automatically modified to the form that is consistent with the changed ontology. Any ontology is formed and modified by knowledge engineers using the same tool – the ontology editor.

For the formation and modification of knowledge and data bases, as well as declarations (specifications) of software components, a specialized editor is proposed that automatically generates a user interface by component’s ontology [12]. The editor is based on the process of generating a digraph of information based on an ontology graph, which provides a correspondence between them. Formation of a digraph of information is carried out from top to bottom. The editing process can be finished at the user request in any state, while the digraph of the information will be formally complete or incomplete in relation to the ontology digraph.

Editing a digraph of information is possible to the extent that does not break the correspondence between it and the digraph of its ontology. In addition, the editor automatically adapts to changes in the representation of knowledge and data. To represent the digraph to the user in a convenient form, additional tools of generating multimodal interfaces are offered: in the form of a graph, a text, and a table.

IV. Decision Support Service Components

The division of the system into information and software components has the following goals:

- independent development of problem solvers and information resources by relevant specialists;
- components of both types are stored in the fund of information units and are reusable – one problem solver can be associated with different knowledge bases and vice versa.

The separation of ontologies from knowledge bases and the explicit representation of ontologies induces the need to replace inference machines that interpret production rules by specialized ontology-based reasoners – ontological solvers. Such solver traverses the declarative knowledge base (KB) comparing its sentences with input information (about the object) in order to search and approve hypotheses or refute them. The number of rules for comparing the elements of a KB to elements of domain objects is rather small for most problems. For example, for diagnostics, one of the most common types of sentences is <failure-type-k, sign-j, range of values-kj of sign-j) which is matched to the pairs <sign-j, value-kj of sign-j> (or triplets – with the addition of a date). The most important result of algorithm’s work is a structured report which is formed in terms of specialists and takes into account their requirements.

Since it is mainly programmed a set of rules for comparing elements of knowledge base sentences with elements of observations (and there are few such rules), an ontology based solver based is usually not complicated. This is determined by the number and complexity of the types of axioms (sentences), the length of the chains of causal relationships between the sought-after (tested) hypotheses and the elements of domain objects’ descriptions and/or restrictions being set on them (for example, in the treatment problem, the restriction is “the object must become healthy”).

An IACPaaS platform solver is a collection of agents interacting with each other through the exchange of messages generated by some templates. Solvers have a declarative representation. Agents and message templates have declarative parts as well as procedural ones.

Descriptions (declarations) of software components are formed in a unified structure defined by the same metalanguage (one for each type of component). The solver has a list of formal parameters (ontologies) that determine the set and types of processed resources – data, knowledge, explanations.

The procedural part is a bytecode. Its source code is written in Java using the methods provided by the platform API. It contains, in particular, a set of program interfaces for accessing the content of information resources (vertices and arcs of digraphs), hiding the format of their internal representation. This allows developers to use the proposed high-level data types without thinking about the details of the internal organization and storage of information.

The construction of solvers is carried out by setting the relationships of the agents (dynamically or statically): in the code of agent production blocks or through a control graph (methods for agent interaction are also contained in the API).

V. Decision Support Service Development Technology

The technology for creating decision support services in a broad sense includes system analysis of domain and system design stages. System analysis precedes the start of automation: all intelligent tasks requiring information support are being identified. At system design stage intelligent (intellectual) tasks and their sub-tasks are distributed among components of particular types.

The key principle in designing viable decision support services is the inclusion of declarative knowledge bases in its architecture. A solver being produced is an ontology-oriented one. It has to be designed in accordance with domain ontology (take into account structural and causal relationships between domain concepts and limitations for interpretation). The most important output of service work is a structured report formed in expert terms and taking into account specialists' requirements. Future users are involved in the development of its structure, they affect how results are being grouped (depending on their importance). This structure being a part of the domain ontology is almost as important as the structure of stored information resources.

Intelligent software services are built as follows:

- A software problem solver suitable for identified task is selected.
- In accordance with the formal parameters of the solver, a set of actual service parameters is formed. (An actual parameter is an information resource based on some particular ontology and intended for processing.)
- The problem solver is linked to the selected actual parameters (in declarative part of the service).

Development technology in a narrow sense is a toolbox used to create and improve components of knowledge-based decision support services:

- a knowledge base editor;
- a database editor;
- an editor of data structure for solution results and explanation;
- a library of software solvers for task classes (and their components);
- a tools for search of ready-made components and for their integration;
- a specialized editor for GUI.

These requirements are met by the IACPaaS platform.

VI. The Development Of Decision Support Services

A system analysis, preceding the automation, identifies the tasks that need knowledge-based information support. Examples of a tasks are: formation of hypotheses about the diagnosis that do not contradict with the description of an object; requesting additional information (to confirm hypotheses and to reduce their number).

An ontology-oriented solver is created on the basis of the specified structural and causal relationships between concepts, limitations for their interpretation and ontological agreements.

Using the integration tool with knowledge base and solver, a service is constructed. To fill in an input data, a data editor is provided (generated by ontology) that allows to select correct terms and names for observations.

The structure of the result (the explanation of hypotheses or solutions) is created before the solver development and should be problem-oriented – must take into account general needs of the person who solves the corresponding problem. The format of the structured report on the results is consistent with the experts notions, it groups the results according to their importance. Each group can be viewed with various degrees of detail with a standard user interface. To meet needs of specific users, a special GUI can be created.

For example, on the medical portal of the IACPaaS platform a software solution to the medical diagnostics task has been developed – a special decision support service. For the gastroenterology domain the following has been created: "The base of terms of gastroenterology", "The knowledge base about diseases of the digestive system", "The diagnostic problem solver" and “The tool for filling in case records". The results report provides a link to medical case record, set of confirmed hypotheses about diseases for it (one or more), set of rejected hypotheses [13].

VII. Extensibility Of Toolkit For Building Knowledge Base Systems

Implementation of the toolkit requirements regarding viability support is ensured by providing expansion of its functionality by both developers and users. Users of the toolkit can improve it if the improvement tools (instrumental services) are as similar as possible to tools for applied service development. Then the complexity of this improvement process is comparable to the complexity of developing service components.

Considering that the developers of ISS are not only programmers but also knowledge engineers and experts, development tools based on a unified, single declarative presentation of information and software components with subsequent automatic generation of editors for their formation are set as a basis for this toolkit.

The development of the toolkit could be carried out both by the main team of toolkit developers, and by its users participating in the ISS development. Such participation can be expressed in the creation of libraries of reusable components, new technologies, and tools for supporting the component development.

The important requirement is the presence of a single environment and instrumental support for the development of tools and for the development and maintenance of services. It is important, firstly, because as a result of the development of the ISS, some of its elements can be directly used as an extension of the toolkit (the most obvious example is the library of reusable components). Secondly, service developers can create a new “add-in” to solve their problems, which can be useful for others, for example, a new technology and its tool support, a new solver (as a shell for many services), a new set of development tools, etc.

Since all components are formed according to their ontologies, this toolkit includes editors for forming those components. Such editors are formed by combining the stencil editor with the ontology of the corresponding component. In addition to the above elements, this toolkit contains external software tools. They are used to create service GUI, procedural part (bytecode) of software components (where declarative representation is not enough for operation), as well as a control system. The maintenance of the basic toolkit elements is carried out by its developers.

Expandable toolkit may include new technologies for developing services and their instrumental support. These include collective development and management tools that ensure the separation of competencies among developers with their sequential or parallel work on the formation of models and components. Such technologies and tools should ensure the architectural integrity of entire portals, ISSs, solvers, development tools for adaptive multimodal interfaces, libraries of reusable components (information and software). Expansion can be carried out using basic tools, as well as mechanisms of expandable tools.

VIII. Conclusion

The ontology-oriented approach is the basis for formation of understandable knowledge and evolving knowledge-based systems. On the IACPaaS cloud platform, which offers the infrastructure and technology for developing intelligent decision support systems [12], cloud services are being constructed of information and software components. The general principles of development and main features of the created services are:

- implementation as a system with a knowledge base that has a declarative representation (semantic network);
- formation of ontology-oriented solvers;
- using an ontology-driven structural editor allows to include domain experts (usually several ones) in the development process;
- services and tools are cloud-based, which makes them accessible to the entire interested community over the Internet;
- possibility of continuous improvement of knowledge by domain experts for ensuring the viability of ISS.

Acknowledgment

This work was partially supported by the Russian Foundation for Basic Research (projects 18-07-01079 and 19-07-00244).

References

[1] C. Izurieta, and J.M. Bieman, “A multiple case study of design pattern decay, grime, and rot in evolving software systems,” Software Quality Journal. vol. 21, (no.2). pp. 289-323, 2013.

[2] H.P. Breivold, I. Crnkovic, and P.J. Eriksson, “Analyzing software evolvability,” Computer Software and Applications, COMPSAC'08. 32nd Annual IEEE International, Turku, Finland, July - August,. pp. 327-330, 2008.

[3] S.M.H. Dehaghani, and N. Hajrahimi, “Which factors affect software projects maintenance cost more?,” Acta Informatica Medica. no. 21(1), pp. 63-66, 2013.

[4] O.A. Krjazhich, “Ensuring viability of information in time at its processing in DSS“, Matematichni mashini i sistemi, 2015. no. 2. pp. 170-176. (in Russian).

[5] A.G. Dodonov, and D.V. Landje, “Viability of information systems”, K.: Naukova dumkab, 256 p, 2011. (in Russian).

[6] J.W. Lloyd, “Practical Advantages of Declarative Programming, Proc. of Joint Conference on Declarative Programming,” GULD-PRODE`94. Peniscola, Spain, September, pp. 3-17, 1994.

[7] I.A. Dekhtyarenko, “Declarative programming”, SoftCraft diverse programming, 2003. (in Russian) URL: -http://www.softcraft.ru/paradigm/dp/ (accessed: 02.16.2018).

[8] S. Tu, J.R. Campbell, J. Glasgow, M.A. Nyman, R. McClure, J. McClay, C. Parker, K.M. Hrabak, D. Berg, T. Weida, J.G. Mansfield, M.A. Musen, and R.M. Abarbanel, “Use of Declarative Statements in Creating and Maintaining Computer-Interpretable Knowledge Bases for Guideline-Based Care,” AMIA Annual Symposium proceedings, pp. 784–788, 2006.

[9] M. Martinelli, D. Moroni, O Salvetti, and M. Tampucci, “Knowledge-based Infrastructure for the Management of Diagnostic Imaging Procedures in the Heart Failure Domain,” Trans. Mass-Data Analysis of Images and Signals? no 2, pp. 3-18, 2010.

[10] V. Gribova, A. Kleschev, Ph. Moskalenko, and V. Timchenko, “Implementation of a Model of a Metainformation-Controlled Editor of Information Units with a Complex Structure”, Automatic Documentation and Mathematical Linguistics, vol. 50, no.1, pp. 14-25, 2016.

[11] V. Gribova, A. Kleschev, Ph. Moskalenko, and V. Timchenko, “A Two-Level Model of Information Units with Complex Structure that Correspond to the Questioning Metaphor,” Automatic Documentation and Mathematical Linguistics, vol. 49, no 5, pp. 172-181, 2015.

[12] V. Gribova, A. Kleschev, Ph. Moskalenko, V. Timchenko, L.Fedorischev, and E. Shalfeeva, “The methods and the IACPaaS Platform tools for semantic representation of knowledge and development of declarative components for intelligent systems.” Open semantic technologies for designing intelligent systems. Issue 3, pp. 21-24, 2019.

[13] V. Gribova, M. Petryaeva, and E. Shalfeeva, “Cloud decision support service for diagnosis in gastroenterology,” Physician and Information Technology, no 3, pp. 65-71, 2019. (in Russian).