Integrating medical data management and decision-making systems with common metamodel

Aliaksandr Kurachkin

***Abstract*—The paper proposes a unified declarative approach to developing and integrating medical information systems. Two main usage scenarios for data access are considered – retrieving patient-based medical history for complex diagnostic scenarios, and retrieving research-based historical data for generalized statistical analysis. The concept of metamodel is introduced as a descriptive layer used to inter-integrate data access, expert system development and evaluation, and user interface for interacting with stored data and predictive models. Proposed metamodel structure can be used in order to generalize and simplify data access and validation independently of specific database or storage solution, and provide a common inter-application communication API. It also can be used to aggregate and translate individual entity information to datasets for developing and verifying supervised machine learning models and verifying rule-based inference models, allowing to create and analyze various types of decision-making systems based on provided data. Finally, using the metamodel for medical information systems development also allows to procedurally generate corresponding form-based and list-based views to rapidly prototype user interfaces based on common controls, for any given record data structure.**

***Keywords*—decision-making systems, medical systems, metamodel.**

I. Introduction

Medical information systems are one of the most important directions of research & development in IT. Deeper informational integration in medical practice enables usage of modern coherent solutions for solving various problems. Most commonly, these problems are related to patient information processing in one way or another – storing and actualizing information on patient's medical history, diagnostic research data, therapeutic measures. Data processing also includes proposing diagnostics and treatment strategies using medical expert systems and decision support systems. Processing an entire set of data available for a specific patient simplifies diagnosis, since it allows to project complex cases based on disjointed results for various different examinations, while integrating diagnosis process with expert systems allows for additional insights and increases the overall quality of medical services [1].

One of the challenges for inter-integration of various medical information systems within a single medical establishment is the absence of a common standardized interoperability model. Medical information systems employ different types of databases and storages, specific diagnostic hardware and respective software solutions, various formats and protocols for storing and transmitting information.

One of the most well-known solutions for medical systems integration within a single medical establishment is DICOM (Digital Imaging and Communications in Medicine) standard. This standard specifies file layer for information storage and communications layer for network communication on transport and application OSI layers over TCP/IP and HTTP protocols, respectively. The biggest disadvantage of DICOM is the fact that the standard is the rigid DICOM file structure for storing information, focused primarily on working with images or series of images. In addition, the attribute model of the DICOM file standard is redundant (i.e. the same research parameters can be represented by several distinct attributes), which complicates the initial data entry and the development of graphical user interfaces for information systems using DICOM. In addition, the standard is not adapted for batch processing of several patient records for a specific type of research, which leads to additional difficulties in statistical analysis within a specific research type and complicates its usage while developing and verifying expert system and decision support system models [2].

To solve these problems, this paper considers a generalized model of a medical information system, according to which any information system can implement a common interface for simplified inter-integration. This model assumes 3 main use cases:

1) storage of general medical history information;
2) inputting and storing research-specific data on concrete diagnostic and therapeutic procedures;
3) development, testing and integration of expert systems and decision support systems based on available data.

II. Data Metamodel for Medical Information Systems

Any information system operates with a certain set of structured information based on so-called data model – a declarative description of entities and their unique characteristics that can be distinguished within the data source. Usually, a data model is understood as a logical data scheme – the organization of individual objects, subjects and their relations, expressed in a dataset, in accordance with the requirements of the domain.

Manuscript received September 30, 2020.

A. Kurachkin is with Belarusian State University, Minsk, Belarus (phone: +375 29 304 53 21; e-mail: alex.v.kurochkin@gmail.com).

There are two main problems of working with the logical data schema – the complexity of projecting the logical scheme onto the scheme of any storage or database management system (DBMS), and the variability of the scheme itself [3].

The logical schema sets the characteristics of individual entities and their relationships in the form of simple logical statements. To map these statements to the storage schema of specific database management systems, an additional conversion step is required. For example, relational DBMSs require data normalization to project a logical data schema into a relational one. In many cases, this projection (including normalization) of the logical schema can be performed in various mutually exclusive ways. At the same time, the specific data schema definition of used DBMS may differ significantly from the logical schema, since it may be necessary, in order to comply with the rules for organizing the information of the DBMS data schema, to introduce additional structures that cannot be mapped to the logical schema (for example, additional link tables are required to implement a "many-to-many" relationship in relational DBMS).

The variability of the logical schema arises naturally due to the fact that all possible characteristics of the individual components of the data model cannot be taken into account at the design stage. Because of this, it becomes necessary to modify the logical schema “on the fly”, during its use; if, at the same time, the logical schema also specifies the format of the physical data organization for the corresponding storage scheme, any modifications to the scheme require rebuilding data storage structure on the medium from scratch. In addition, individual elements of a schema (usually attributes) can be optional or added ad-hoc, and their inclusion to the schema significantly complicates aggregation and accumulative processing of data.

To solve this problem and form a generalized structure for working with heterogeneous types of data, a metamodel concept is proposed. A metamodel is a data model that is used to describe the format and structure of information that is operated on by a particular medical system or storage. The metamodel, in its essence, is close to schema description languages, such as DDL (Data Definition Language) in relational DBMS based on SQL; however, in addition, metamodel can be used to generate necessary views for displaying and editing data, as well as for organizing them in a structure suitable for building expert systems and decision support systems.

The proposed structural diagram of various modules of the medical information system, as well as the links of these modules with the metamodel, is shown in Fig. 1. As can be seen from the presented scheme, the metamodel of the medical information system is used to solve 4 types of tasks: formalization and generalization of the data access interface, generation of the user interface, communication of the user interface with the data access layer, as well as working with expert systems and decision support systems.

III. Using the Data Metamodel to Generalize the Data Access Interface

Like schema definition language in relational DBMS, metamodel must specify a descriptive representation of entities, attributes and their types, as well as their inter-relationships. To provide a sufficient model structure representation within the metamodel, it is proposed to leverage JSON Schema – a well-known open standard for describing entities expressed in JavaScript Object Notation (JSON) with their attributes and validation constraints [4].

Describing a metamodel using JSON Schema solves two main tasks – building data access interface based on provided type information, and model validation.

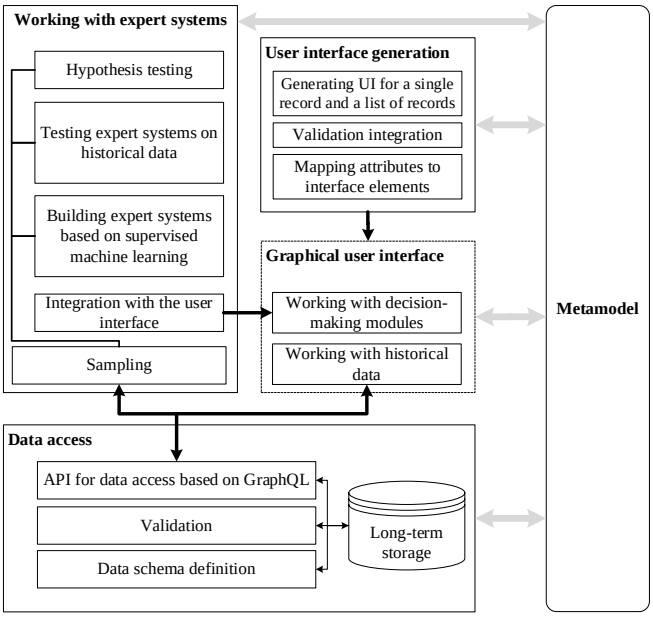

Figure 1. Structural diagram of main medical information system modules and their relationship with the metamodel

When designing any kind of interaction with one or more medical information systems, information about the schema, available as part of the metamodel, can be used directly when accessing these systems for any CRUD (Create, Read, Update, Delete) operation – any authorized client application can work directly with any data source available based on the description provided by the metamodel. The metamodel itself also allows for strong typing and automatic generation of client interfaces, since it provides all the data about entities, attributes, and relationships. Based on the schema described in the metamodel, a server-side interface can be generated for implementing any type of API that can be used to access the described entities. It is proposed to use the GraphQL standard as a universal solution for providing an external API for data access [5].

In addition to explicit type annotations for the data access interface, metamodel schema also allows to declaratively specify validation logic for program and user input, when related entities are added or modified. So, when external APIs are used to communicate via auto-generated interface to create a new entity with a specified set of attributes, this set can be automatically checked for correctness relative to the schema specified in the metamodel immediately before executing actual storage-backed modification. The JSON Schema standard defines a wide set of keywords for

validating scalar and composite attributes by type and value, and also allows for more fine-grained validation of numeric values (minimum, maximum, multipliers, etc.), string values (length, adherence to a certain formats like date or email address, or matching against regular expression), as well as nested objects and arrays.

At the level of the information system itself, the metamodel and description of data types can be used to generate a data access middleware to describe the data schema of the underlying storage, for example, when using ORM frameworks, Repository pattern or Active Object pattern. In addition, some DBMSs allow generating queries in the appropriate schema description language using a JSON Schema.

Thus, the proposed metamodel schema definition allows it to be used as an entry point for describing and typing the entities, as well as for more complex validation of individual attributes of these entities. The use of JSON Schema, a well-known open standard, greatly simplifies the integration of such a description with other existing solutions.

IV. Using the Data Metamodel to Generate the User Interface

The basis of user interaction with any information system is the graphical user interface. Traditionally, a user interface is a static collection of controls that are manually associated with attributes of a specific entity or view model. Typical operations that a graphical user interface should support when working with freeform entities are:

1) viewing multiple entities in a list view, with support for sorting, searching and filtering by individual attributes;
2) viewing detailed information on a single entity;
3) adding a new entity;
4) modification of an existing entity;
5) deleting one or more entities.

For most information systems, the described operations are implemented in a fairly uniform manner and differ only in the type and format of the input data. For example, an individual patient or a separate medical examination protocol can act as an entity, and in both cases the graphical user interface to support the described operations will be similar. Thus, it is proposed to generalize the mechanism for generating the user interface itself. In general, the metamodel contains sufficient data to generate a graphical user interface describing the entity fields that are used within a specific medical system.

For example, if a medical information system operates with examination protocols, a specific protocol may be represented by a number of research results mapped to appropriate entity attributes. To form the user interface, it is sufficient to compare the description of a specific entity attribute corresponding to the examination protocol field with a specific user interface control element. In the proposed model, the following strategies for generating the user interface are implemented:

1) text fields: single line or multiline text input fields,
2) numeric fields: single-line text input fields with a restriction on numeric values,
3) date and/or time: a special form of text input field with date picker (calendar), time picker and format restrictions,
4) binary categorical fields: checkboxes,
5) categorical fields with three or more categories: drop-down lists,
6) categorical fields with multiple selection: multiple choice dropdowns,
7) multiple criteria aggregate score: a set of checkboxes for each of the criteria,
8) single reference fields: entity selection field with additional lookup UI and the ability to unbind,
9) multiple reference fields: an inner list with additional lookup UI to add elements and ability to unbind.

Processing with text, numeric, and categorical fields is implemented using standard user interface elements. Multiple criteria aggregate scores are formed on the basis of a set of characteristics that are not mutually exclusive, and their inclusion or exclusion is used to calculate the resulting score (for example, the ASPECTS scale). In this case, the formation of the final score by the values of the selected characteristics occurs automatically, i.e. metadata description contains base scale value and score increments for each of the options. When working with reference fields, a special interface for displaying a single link or a nested list is implemented, while creating or replacing a link to another entity is implemented through a separate dialog box with the ability to search and filter; unlinking (removing the reference between entities) is also supported.

The automated generation of the user interface allows using the metamodel as a single point of description of both data formats and the user interface itself. In addition, individual validation functions, described as validation constraints within the metamodel, can be automatically included within the client user interface, which simplifies data entry and allows the system to provide feedback to end users directly upon entry without waiting for service-side validation. For example, if a certain regular expression is specified as an additional validation attribute for a certain text field within the metamodel, when generating the user interface, such validation can be additionally performed directly on the interface application itself, either as a reaction to user input or before sending serialized data to the service, and if there is an incorrect input, the user will receive an error message immediately, before submitting the form, providing a more responsive user experience. It should be noted that the presence of client validation does not eliminate the need for additional validation on the side of the application itself, since a potentially incorrect request can be generated not only by a client application with a user interface, but also by any other application with access to the external API of a medical information system.

In addition to directly adding and editing information about individual entities of the information system, based on the metamodel data, a user interface can be generated in a similar way for viewing the list of records, searching and filtering by individual fields. It is proposed to use a tabular view as the main view for a set of entities. In this view, each record is displayed as a row in the table, and the columns correspond to the attributes of the records. For greater flexibility, individual columns can be reorganized or hidden,

and to simplify user access, each metamodel can additionally specify a standard set and order of columns for the default table view. In addition to directly displaying the list, the table view allows sorting by any column, as well as searching by any column, which is translated into the corresponding requests for data within the target system, that can, in turn, translate those requests into concrete data storage queries, based on the selected storage mechanism. The metamodel can also contain markers about whether any particular field should be excluded from filtering (search) and sorting, since such operations may not make sense for some attributes.

V. Using the Metamodel to Work with Expert Systems and Decision Support Systems

One of the key tasks of the metamodel is to simplify access to data to develop and verify expert systems and decision support systems.

When working with expert systems based on data, it is possible to use the to generate a request can to the information system itself to form a training dataset for the model. At the same time, the request for historical data can list the fields that need to be used as input features, and also indicate the target fields that must be determined by a particular model when forming systems based on supervised machine learning methods.

Individual queries for data can be complex. Since the input data vectors in supervised machine learning models must be of a fixed length, complex nested attributes must be appropriately transformed when sampling, depending on the logic of a particular expert system. Thus, in general, data fetching requires support for arbitrary queries with aggregations, filtering, and other types of transformations. The use of GraphQL as a base API allows, in addition to direct access to data, to implement arbitrary query support in a unified and uniform manner [5]

In a similar way, cross-validation samples can be formed for testing existing expert systems and verifying both statistical (supervised machine learning) models and formal rule-based expert systems.

Using a single point of access to obtain initial data allows to organize a pipeline for rapid prototyping, testing and comparison of various types of decision support models, as well as tuning their hyperparameters.

In addition to providing a single point of access for the formation of samples, the developed decision support systems can subsequently be integrated into the generated user interfaces of the medical information system. In this case, metamodel allows describing a set of rules that allow the conversion of attributes values filled in by the user into an input vector for a specific decision support model, after which a control element can be added to the user interface to display the prediction of the support system based on the entered information. Interface unification allows making predictions for both existing entries and new entries. In addition, the user interface form of creating a new examination protocol can be used as an interface for direct access to the decision support system, with the prediction of the expert system displayed directly when entering protocol fields. The specialist can see that prediction immediately and rapidly make certain diagnostic decisions.

VI. Conclusion

The paper presents a generalized architecture of a medical information system based on a concept of metamodel. It is shown that the descriptive definition and typing of entities that are used within the system make it possible to optimize the development of such a system by automated generation of methods for data validation, working with long-term storage, forming samples based on queries of varying complexity, building a graphical user interface and providing the ability to develop, test and integrate various predictive models based on expert systems and decision support systems.

On the basis of the proposed generalized approach, 3 systems have been implemented for the integration of decision support systems in various fields of medicine: an expert system for predicting the choriality of multiple pregnancies based on the results of prenatal diagnostics integrated in Republican Scientific and Practical Center “Mother and Child” [6], an expert system for determining degenerative optic neuropathies based on the results of optical coherent tomography and scanning laser polarimetry integrated in Ophthalmological Consultative and Diagnostic Center of the 3rd City Clinical Hospital in Minsk [7], as well as an expert system for predicting the outcome of thrombolytic therapy on the basis of City Clinical Emergency Hospital of Minsk [8]. The generalization of the approach to the formation of samples for the provided storages in these medical systems made it possible to generate the necessary data for developing and verifying expert models, and the generation of the user interface allowed the specialists to use the developed models in clinical practice. Even though the problems considered cover different areas of medical expertise, the similarity of the requirements confirms that the proposed model of the medical expert system and described metamodel definition can be used to generalize the most typical stages of developing such systems and significantly reduce the time for their deployment.

Acknowledgments

Author expresses gratitude to medical establishments that provided necessary data and allowed to test and integrate proposed solutions: Republican Scientific and Practical Center “Mother and Child”, Ophthalmological Consultative and Diagnostic Center of the 3rd City Clinical Hospital in Minsk and City Clinical Emergency Hospital of Minsk

References

[1] M. F. Collen, W. E. Hammond. *The History of Medical Informatics in the United States (Health Informatics).* New York: Springer, 2nd ed, 2015, 777 p.

[2] O. S. Pianykh. *Digital Imaging and Communications in Medicine (DICOM): A Practical Introduction and Survival Guide.* New York: Springer, 2nd ed, 2009, 602 p.

[3] R. Elmasri, S. Navathe. *Fundamentals of Database Systems.* New York: Pearson, 7th edition, 2015, 1280 p.

[4] T. Marrs. *JSON at Work: Practical Data Integration for the Web*. Cambridge, MA: O’Reilly Media, 2017, 376 p.

[5] E. Porcello, A. Banks. *Learning GraphQL: Declarative Data Fetching for Modern Web Apps*. Cambridge, MA: O'Reilly Media, 2018, 198 p.

[6] O. V. Pribushenya, A. V. Kurochkin. "Assessment of choriality in multiple pregnancies by modern expert and decision-making systems" *Modern perinatal medical technologies in solving problems of demographic security*, vol. 10, 2017, pp. 106-111.

[7] T. V. Kachan, A. V. Kurochkin, et. al. "Role of artificial neural networks in detecting early ganglyonar retinal cell death in patients with degenerative optic neuropaties", *Ophtalmology. Eastern Europe*, vol. 9, iss. 4, 2019, pp.445-458.

[8] K. V. Senko, A. S. Fedulov, A. V. Kurochkin, E. A. Halavataya "Prognosing the outcome of thrombolytic therapy in patients with ischemic stroke based on neural network analysis", *Neurology and neurosurgery. Eastern Europe.*, submitted for publication.