УДК 004.41

Valeriy Mironov, Artem Gusarenko, Nafisa Yusupova

Ufa State Aviation Technical University, Ufa, Russia

SITUATION-ORIENTED DATABASES: DOCUMENT MANAGEMENT ON THE BASE OF EMBEDDED DYNAMIC MODEL*

ABSTRACT

Present state of research situation-oriented databases (SODB) is discussed. SODB as model-driven data-processing application is considered. Using the embedded finite-state dynamic model for the specification of data processing according to the current situation is discussed. Hierarchical situational model (HSM), used as a meta-model for the dynamic modeling of specific applications, is considered. HSM elements intended for document processing based on the concept of Data Processing Objects are discussed.

KEYWORDS

Situation-oriented database; web application; dynamic model; finite state model; NoSQL; HSM; XML; DOM.

Миронов В.В., Гусаренко А.С., Юсупова Н.И.

Уфимский государственный авиационный технический университет, г. Уфа, Россия

СИТУАЦИОННО-ОРИЕНТИРОВАННЫЕ БАЗЫ ДАННЫХ: УПРАВЛЕНИЕ ДОКУМЕНТАМИ НА ОСНОВЕ ВСТРОЕННОЙ ДИНАМИЧЕСКОЙ МОДЕЛИ

АННОТАЦИЯ

В статье обсуждается текущее состояние исследований в области ситуационно-ориентированных баз данных в части обработки данных внутри приложений на основе динамической модели. Обсуждается использование встроенных динамических моделей для спецификации обработки данных в соответствии с текущей ситуацией. Ситуационные модель используется как метамодель для динамического моделирования элементов и подсистем приложения. Рассматриваются специализированные HSM-элементы, предназначенные для обработки данных и документов, лежащих в основе концепции обработки данных.

КЛЮЧЕВЫЕ СЛОВА

Ситуационно-ориентированные базы данных; веб-приложение; динамическая модель; модель конечных состояний, NoSQL; HSM; XML; DOM.

Introduction

Notations Business Process Modeling. Currently, the modeling of business processes developed notation of using diagrams which are determined based on the semantic structure of information systems. The most common of these are BPMN and UML.

BPMN (Business Process Model and Notation) - Business Process Modeling Notation is a unified representation of the formal language for the design of modern information systems. This notation can be easily transferred to the platforms of two or more independent users (platforms must support BPMN 2.0), and is part of the project documentation. This notation has the capability of business process modeling, but does not cover the area of data modeling and organizational structure, which is a disadvantage notation. Instead, BPMN modeling solves the problem of data streams and message flows. Process modeling using BPMN uses a set of graphical charts, by which the general representation of the business process, which helps the user to understand the business process logic.

UML (Unified Modeling Language) - graphical modeling language for object-oriented approach to software development of information systems. UML notation allows to create abstract models of systems designed for visualization, documentation and design of information systems. This notation is not a programming language, but code generation is possible. Despite all the advantages of UML insufficient for

* **Proceedings of the XI International scientific-practical conference «Modern information technologies and IT-education» (SITITO'2016), Moscow, Russia, November 25 - 26, 2016**

data-base design, as required to generate an executable code for the model and its interpretation during operation of the system for processing data within an information sys-tem, and external services. UML is a general-purpose language, whereas SODB database design requires a specialized language. UML can be used as a secondary tool for designing information systems. The language contains a large number of diagrams, among which there are structural diagrams and charts to describe the behavior of their analytics provide advantages in designing, as it is close to many object-oriented programming languages and using a behavior chart describes the behavioral aspects of the future system. The user can easily read charts as charts syntax is easy to learn.

Case-tools. On the basis of existing notations developed specialized Case-tools for UML has several commercial solutions, including the development of databases, for example, Oracle Designer Suite. These tools allow graphically construct relational database model and generate the SQL-code for the application.

The SODB uses a dynamic model based on XML which has its own syntax for defining data structures required in situations and conditions associated with DPO-model objects with the data processing specifications. Thus, for SODB it requires special tool Case-oriented dynamic model of language.

Currently, the document-oriented databases are actively developing within the NoSQL movement [1]. Web application services to provide clients with access to remote information resources are developed on this basis. Situation-oriented databases (SODB), the concept of which was proposed in 2010 [2] can be assigned to this area. Since that time, some aspects of SODB functionality were studied with the sup-port of Russian Fund of Basic Research:

- SODB use as the basis for web applications [3, 4];
- Processing of XML documents in SODB based on dynamic DOM objects [5, 6];
- The hierarchical organization of the user interface based on SODB [7, 8];
- OLAP-oriented organization of the user interface based on SODB [9, 10];
- Processing JSON documents and documents provided by web services [11, 12].

These results were published in Russian in Vestnik UGATU – the journal of Ufa State Aviation Technical University. During the investigation the original concept, the terminology, and the general understanding of the SODB purpose have been corrected. This article presents an attempt to consistently explain the key principles underlying the SODB as a new approach to building data-processing applications.

SODB and Data Processing Applications

We consider SODB as a new, more effective approach to the problem of the development of data processing applications. Let's see how this problem is solved on the basis of SODB at a higher level of abstraction.

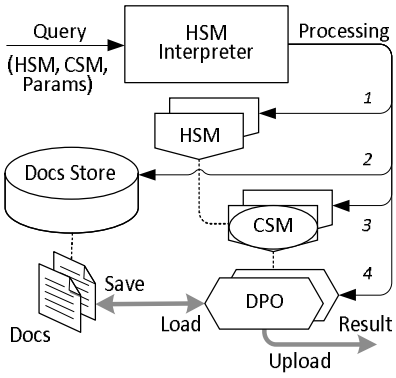

Fig.1. SODB Architecture

SODB Architecture. Fig. 1 shows an architectural model of SODB, which includes the following components:

- A storage of documents Docs Store, containing a set of electronic documents Docs;
- A set of HSM (Hierarchical Situation Models or Hierarchical State Models). Each HSM is a finite-state model corresponding to the business process and defines its states and jumps;
- A set of CSM (Current State Models). Each CSM corresponds to one parent HSM, while one HSM can match zero or more child CSM. Thus, CSM corresponds to an independent implementation of HSM and determines the current state of the implementation;
- A set of buffers DPO (Data Processing Objects), intended for documents processing. Each DPO corresponds to one parent CSM (i.e. to one HSM implementation). Several DPO can be created for a single CSM depending on the current state of the HSM;

- HSM Interpreter – an information processor that manages the database on the basis of the interpretation of HSM and CSM.

SODB functioning. The Interpreter receives a Query, which defines the hierarchical model HSM, the current state CSM and processing parameters Params. Interpreter executes processing cycle (interpretation cycle). It processes the HSM (1) indicated in the Query starting from a state specified in the CSM (3) indicated in the Query, or starting from the initial state, if the CSM is not indicated in the Query. During the processing the Interpreter monitors the current state jumps in accordance with the specifications of HSM (3). The current state changes are reflected in the CSM.

Furthermore, the Interpreter manages the Docs Store (2) and creates / manages run-time buffers DPO (4) (in accordance with HSM specification). For example:

- It loads the DPO certain documents from the repository (Load);
- Processes documents in the DPO;
- Saves the modified documents in storage (Save);
- Uploads the documents as a result of the query (Upload).

Thus, the result of the interpretation cycle may be tripartite:

1. Change the current state of the processed HSM in CSM;
2. Change the content of the documents store;
3. Creating the resulting document.

Thus, the Model Driven Approach is implemented in the SODB architecture. Using the embedded highly abstract dynamic models aims to facilitate the design of the application software. A higher level of abstraction is achieved through the use of HSM declarative form to define the rules of monitoring the current situation and the rules of data processing (and not a procedural form, as in the case with the traditional use of scripts and stored procedures). In these circumstances, the developer is required to create a highly abstract situational model, and the interpreter runs the application in accordance with this model.

Hierarchical Situational Model HSM

HSM Elements. HSM is an ordered set of elements forming a hierarchy (tree) in the sense that the model has a single root element, and each non-root element has a single parent element and may have several ordered child elements. Each element contains three components: Type (mandatory); Name (mandatory); Attributes (optional).

HSM Notations. Two equivalent notations can be used to represent HSM: graphical notation designed for developers, and text notation designed for the interpreter.

When using the graphical notation, the element type is specified as an icon, and the element name and element attributes are written to the right of the icon. The hierarchy of elements is formed by a connector according to the following rules:

• Connectors attached to the bottom of the element icon, or on the right, leading to its child;

• Connectors attached to the element icon on the left, or from above, leading to its parent.

When using the text notation, the XML syntax is used. The HSM element type is specified as XML namespace prefix. The HSM element name is specified as the name of XML element. The HSM element attributes are specified as the XML element attributes, respectively. The hierarchy is formed by embedding elements in accordance with the rules of XML.

HSM Main Elements:

sta State – the root of HSM (root state), or child of submodel element (submodel state);

sub Submodel – a child of a certain State. It used to specify a set of internal states, one of which is the current at each moment;

act Action – a child of a certain State. Used to specify certain activities to be performed when the parent State is the current state;

jmp Jump – a child of a certain State. It is used to change the current state in accordance with the specified activity predicate.

HSM Example. Fig. 2 shows an example of a two-level business process model that we use to illustrate the HSM. Fig. 3 shows an example of the HSM graphical notation. The equivalent representation in a text notation is shown in Listing 1.

The model (see Fig. 2) corresponds to a typical multi-user web application that serves both anonymous and registered users. Anonymous users are granted access to shared (Public) data. Registered users have access to additional special (Private) data. The logic of the application (Application Level) is represented as a transition graph, which nodes correspond to the possible states (situations), and the arcs correspond to the state jumps. BP symbol indicates the initial state of the model. Activity predicates which marked arcs, define the conditions of the current state jumps.

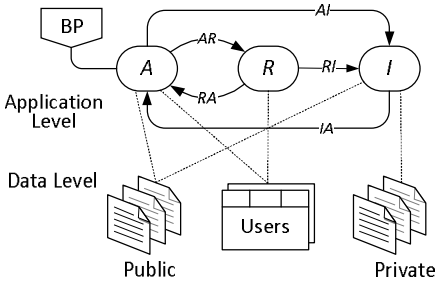

Fig.2. An example of a two-level business process model

The model contains three aggregated states: A (Anonymous) – the initial state corresponding to the anonymous user. Any user begins as an anonymous; I (Identified) – the state corresponding to the identified registered user; R (Registration) – the state corresponding to the user registration process.

Activity predicates have the following meanings: AI – an anonymous user successfully passed the authentication procedure; AR – an anonymous user wished to register; RI – a user has successfully passed the registration procedure; RA – a user has not passed the procedure of registration; IA – identified user has completed the session.

Data Level includes three types of data: Public and Private electronic documents, and a table of registered Users. Public documents are available in states A and I. Private documents are available only in state I, and the table Users is available in states A and R.

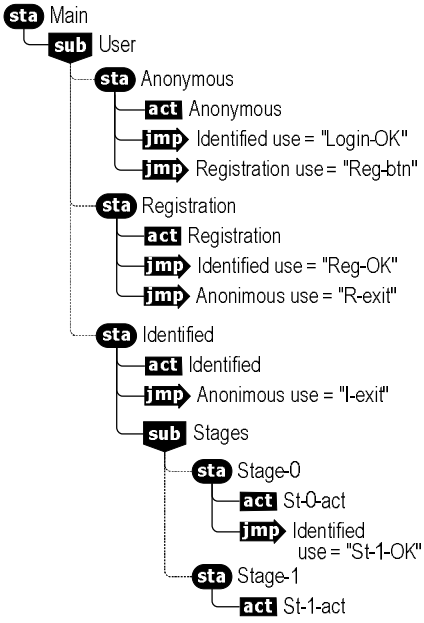

Fig.3. An example of the HSM graphical notation

Root state sta: Main (see Fig. 3) contains submodel sub:User, implements the business logic to manage users. This submodel includes three states: sta: Anony-mous is initial state of the submodel corresponding to an anonymous user; sta:Registration is a state corresponding to the registration of the user; sta:Identified is a state corresponding to the identified user.

Each state in the submodel sub: User includes actions which form the result of the query: act: Anonymous – the image formed on the basis of documents Public (see Fig. 2) and the form for entering the identification data or a command for registration are sent to anonymous users; act:Identified – the image formed on the basis of documents Private (see. Fig. 2), as well as the control to return to the anonymous user, are sent to the identified user; act: Registration – the form to enter personal data for registration or refusal of registration is sent to the user.

Changing the current states mediated by jump elements that refer to the predicates that verify user input, which came in the parameters of the query: jmp:Identified provides a jump from sta:Anonymous to sta:Identified, if a user enters the correct username and password (Login-check function is OK); jmp: Registration provides a jump to sta:Registration, if a user pressed the registration button (Reg-btn function is OK); jmp:Identified provides a jump from sta:Registration to sta:Identified, if a user correctly filled out the registration form (Reg-OK function is OK); jmp:Anonymous returns to the state sta:Anonymous from

states sta:Registration or sta:Identified, if a user clicks the Exit button (R-exit or I-exit function is OK).

State sta:Identified also contains submodel sub:Stages, which define two internal sub-states: sta:Stages-0 and sta:Stages-1.

Thus, HSM is a declarative representation of a hierarchy of states, jumpss between states, and actions associated with the states.

Listing 1. An example of the HSM text notation

```
001   <sta:Main>
002       <sub:User>
003               <sta:Anonimous>
004                       <act:Anonimous/>
005                       <jmp:Identified use = "Login-OK"/>
006                       <jmp:Registration use = "Reg-btn"/>
007               </sta:Anonimous>
008               <sta:Registration>
009                       <act:Registration/>
010                       <jmp:Identified use = "Reg-OK"/>
011                       <jmp:Anonimous use = "R-exit"/>
012               </sta:Registration>
013               <sta:Identified>
014                       <act:Identified/>
015                       <jmp:Anonimous use = "I-exit"/>
016                       <sub:Stages>
017                               <sta:Stage-0>
004                               <act:St-0-act/>
005                               <jmp:Stage-1 use = "St-1-OK"/>
007                                   </sta:Stage-0>
008                                   <sta:Stage-1>
009                                       <act:St-0-act/>
012                                   </sta:Stage-1>
006                           </sub:Stages>
007                   </sta: Identified>
008           </sub:User>
016   </sta:Main>
```

HSM Interpretation

Consider how processing is performed in the situational model interpretation cycle for query processing and results forming.

Multi-pass Interpretation. HSM processing is performed by downward traversal of HSM elements according to their hierarchical order. Traversing begins with the root state; state handling involves the processing of its child elements, and so on. Submodel processing begins with processing of the current state element in accordance with a CSM. Multi-pass interpretation in which the interpreter performs several passes under one HSM interpretation cycle holds. Usually two passes are enough (the initial pass and the final pass), in special cases, additional passes (intermediate passes) are require.

The initial pass is intended for creating and updating the Current State Model (CSM). The initial states of the submodels are set as the current states in creating CSM. Updating the current state, it is performed in accordance with the activity of the jump elements. When jump activity is detected (activity predicate is true), new current state of the processed submodel is recorded in CSM.

The final pass is intended to form the query result in accordance with the reached current state. In this pass, the jump elements are ignored, and only the state elements corresponding to the current states of the CSM are processed.

Changing the Current State. Changing the state is only performed on the first pass when processing jump element. If the processing jump element shell reveals its activity, the following actions are performed:

• The parent state is processed in the epilogue mode (in epilogue mode jump elements are ignored);

• Target state element for the active jump is searched and is recorded to the CSM as the current state of the submodel;

• The new current state element is processed in the prologue mode.

Possibility to lock the HSM elements processing depending on the pass and processing mode using attributes pass and mode is provided.

Current States Model. CSM is a subtree of the HSM, where submodels are represented only by their current states. Fig. 3 shows an example of CSM, corresponding HSM, discussed above (see. Fig. 3 and Listing 1). CSM corresponds to the current state sta:Identified for submodels sub:User and the current state sta: Stage-1 for submodels sub:Stages.

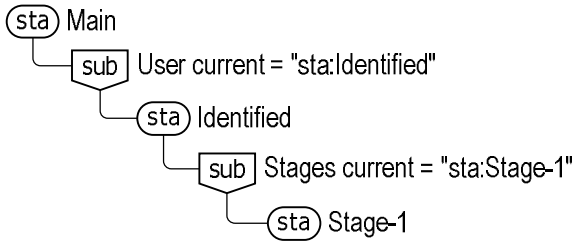

Fig.4. CSM example in graphical form

Special types of action element, called Document Processing Object elements (DPO elements) are provided for the processing of documents from the SODB document store. DPO elements enable us to explicitly specify the document processing in the HSM, not hiding it inside the implementation of actions program code. Currently, two kinds of DPO elements are designed:

• DOM element focused on the processing XML documents using XSL transformation technology [7–9];

• Smarty-element focused on the processing JSON documents using the technology of filling templates [16–25].

We explain the concept of DPO on the example of DOM elements. DOM elements are placed inside the state elements and are specified DOM objects that are created and used for loading, modifying and uploading XML-content during HSM interpretation cycle. These functions are supported by three types of HSM-elements:

dom DOM element is a child of a state element. It is used to create a DOM object, which exists during interpretation cycle;

src Source element is a child of DOM element or of another source element. It refers to the XML document placed in the document store. It is used to specify XML content loading to the DOM-object created by the parent DOM element. Nested source elements allow forming the content of the DOM object on the basis of several XML documents;

rcv Receiver element is a child of DOM element. It is used to upload data from the DOM object created by the parent DOM element to the document store or as a result of a query.

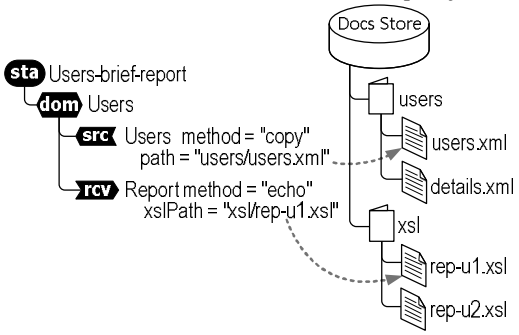

Fig.5. Single XML document processing example

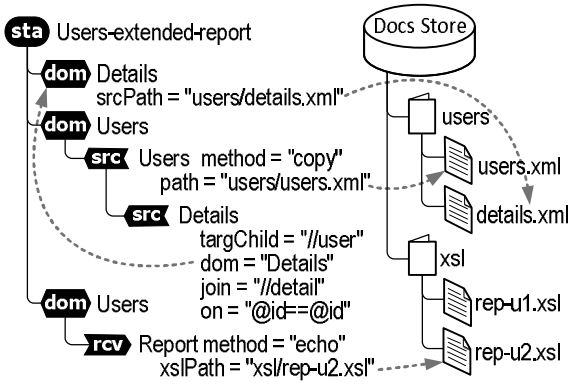

Fig.6. Two XML documents processing example

Fig. 5, 6 and 7 show the use of DOM elements for XML documents processing. Fig. 5 shows a simple case of DOM object creating on the base of a single XML document.

Fig. 6 shows a more complex case, when the content of DOM object is based on the two XML documents.

Models of initial XML documents are presented in Fig. 7a and 7b. Model of resulting XML document formed in the DOM object on the basis of two initial XML documents is presented in Fig. 7c.

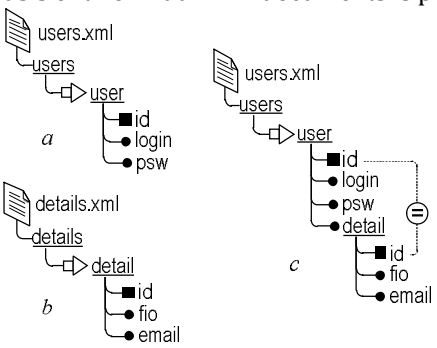

Fig.7. Models of XML documents

DOM Objects Creating and Loading. In the state sta:Users-brief-report (see Fig. 5) the element dom:Users create DOM object containing a summary of the registered users. File users.xml is loaded into this DOM object from the data store directory users by using the source src:Users.

In the state sta:Users-extended-report (see Fig. 6) the element dom:Users create the DOM object containing detailed information in addition to the summary of registered users. For this the auxiliary object dom:Details is created and file details.xml containing detailed information [26-31] users is loaded. Next, the interpreter creates and loads the object dom:Users by merging (equijoin) summary and detailed user information. Each XML element user of the users.xml document is attached as a child to XML element detail of the object dom:Details such that their XML attributes id have the same value (see Fig. 7c). For this the source element src:Details enclosed in the source element src:Users is used (see Fig. 6).

Source src:Details contains the following attributes:

• targChild contains XPath expression that specifies a set of target (parent) nodes (the set of all elements user from the document users.xml), to which child nodes of the source must be attached;

• dom is a reference to the previously created DOM object containing the nodes of the source;

• join contains XPath expression that specifies the set of attached (child) nodes (the set of all elements detail from the document details.xml), which must be attached to the target node;

• on defines join condition (requires that the id attributes of joined nodes have the same value).

This example, in particular, demonstrates the use of multiple DOM objects handed in the same state. Noting that the XML documents merging can be performed without auxiliary DOM object dom:Details and apply the sources pointing directly to the files in the documents store.

Formation of the Results. Fig. 8 explains the formation of the query result via receiver rcv:Report (see Fig. 6). The result is formed as a HTML-document.

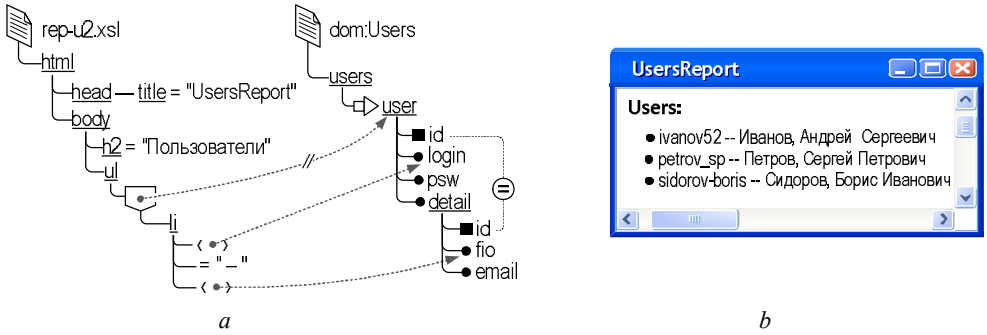

Fig.7. An example of the formation of the resulting HTML-document by XSL transformation of the contents of DOM-object

This receiver defines the XSL transformation of content of parent DOM object in accordance with the style sheet, placed in a documents store in the directory xsl. Transformation model, whereby the interpreter generates the detailed report is shown in Fig. 8a. The resulting screen form is shown in Fig. 8b.

Note that in Fig. 6 receiver, forming a detailed report, is placed in duplicate element dom:Users. Generally speaking, said receiver can be placed in the same element dom:Users, which placed sources src:Users and src:Details. Separation is done to illustrate reusability of DOM elements having the same name to refer to the corresponding DOM object (in both single and several states).

Conclusion

1. SODB is a new approach to building data-processing applications based on the principles of Model Driven Approach.

2. SODB contains an embedded dynamic finite states model in the form of transition states graphs, reflecting the logic of the corresponding business process.

3. Data management is performed by interpreting the embedded dynamic model tracking the current states and access to the data associated with the current states.

4. To construct embedded dynamic models the declarative language HSM is proposed. It allows you to describe in graphic or text form a hierarchy of submodels containing several states, which, in turn, can be specified state jumps, actions and other submodels.

5. HSM interpreter performs several downstream passes of the dynamic model for each interpretation cycle. Current state model CSM is formed as a HSM subtree, wherein for the submodels provides their current states.

6. To specify document processing the document processing elements DPO, generating DOM and Smarty objects, are provided in HSM. Source elements provide loading documents into objects.

7. To specify loaded documents processing and results unloading the receiver elements are provided in the HSM. XML documents are processed by the XSL transformation, and JSON-documents are processed by templates compiler processing.

Литература

1. Sadalage, P. J., Fowler, M. NoSQL Distilled: a Brief Guide to the Emerging World of Polyglot Persistence. Addison-Wesley (2012).
2. Миронов, В.В., Юсупова, Н.И., Шакирова, Г.Р.: Ситуационно-ориентированные базы данных: концепция, архитектура, XML-реализация. // Вестник УГАТУ. 2010. Т. 14, №. 2 (37), 233–244 с.
3. Миронов, В.В., Маликова, К.Э. Интернет приложения на основе встроенных динамических моделей: Архитектура, структура данных, интерпретация. // Вестник УГАТУ. 2010. Т. 14, №. 1 (36), 154–163 с.
4. Миронов, В.В., Маликова, К.Э. Интернет приложения на основе встроенных динамических моделей: Элементы управления пользовательского интерфейса. // Вестник УГАТУ. 2010. Т. 14, №. 5 (40), 170–175 с.
5. Миронов В.В., Гусаренко А.С. Ситуационно-ориентированные базы данных: концепция управления XML-данными на основе динамических DOM-объектов // Вестник УГАТУ. 2012. Т. 16. № 3. С. 159–172.
6. Миронов В.В., Гусаренко А.С. Динамические DOM-объекты в ситуационно-ориентированных базах данных: лингвистическое и алгоритмическое обеспечение источников данных // Вестник УГАТУ. 2012. Т. 16. № 3. С. 167–176.
7. Миронов, В.В., Канашин, В.В. Иерархические виджеты: организация пользовательского интерфейса в приложениях на основе ситуационно-ориентированных базах данных. // Вестник УГАТУ, 2013. Т. 17, №. 2 (55), 138–149 с.
8. Миронов, В.В., Канашин, В.В. Иерархические виджеты: алгоритмы контроля данных пользователя в веб-приложениях на основе ситуационно-ориентированных баз данных. // Вестник УГАТУ. 2014. Т. 18, №. 1 (62), 204–213 с.
9. Макарова, Е.С., Миронов, В.В. Проектирование концептуальной модели данных для задач Web OLAP на основе ситуационно-ориентированной базы данных. // Вестник УГАТУ. 2012. Т. 16, №. 6 (51), 177–188 с.
10. Макарова, Е.С., Миронов, В.В. Функции аналитики в веб-приложениях на основе ситуационно-ориентированных баз данных. // Вестник УГАТУ. 2013. Т. 17, №. 5 (58), 150–165 с.
11. Гусаренко А.С., Миронов В.В. Smarty-объекты: вариант использования гетерогенных источников в ситуационно-ориентированных базах данных // Вестник УГАТУ. 2014. Т. 18. № 3(64). С. 242–252.
12. Миронов В.В., Гусаренко А.С. Использование RESTful-сервисов в ситуационно-ориентированных базах данных // Вестник УГАТУ. 2015. Т. 19. № 1 (67). С. 232–239.
13. Aguilar J. A. et al. An Analysis of Techniques and Tools for Requirements Elicitation in Model-Driven Web Engineering Methods // Computational Science and Its Applications--ICCSA 2015. – Springer International Publishing, 2015. pp. 518-527.
14. Rumpe B. Executable Modeling with UML. A Vision or a Nightmare? // arXiv preprint arXiv:1409.6597. – 2014.
15. Agustin J. L. H., Del Barco P. C. A model-driven approach to develop high performance web applications //Journal of Systems and Software. – 2013. – Vol. 86. – No. 12. pp. 3013–3023.
16. Delgado A., Marotta A., González L. Towards the construction of quality-aware Web Warehouses with BPMN 2.0 Business Processes //Research Challenges in Information Science (RCIS), 2014 IEEE Eighth International Conference on. – IEEE, 2014. pp. 1–6.
17. Delgado A., Marotta A. Automating the process of building flexible Web Warehouses with BPM Systems //Computing Conference (CLEI), 2015 Latin American. – IEEE, 2015. pp. 1–11.
18. Aguilar J. A. et al. An Analysis of Techniques and Tools for Requirements Elicitation in Model-Driven Web Engineering Methods //Computational Science and Its Applications--ICCSA 2015. – Springer International Publishing, 2015. pp. 518–527.
19. Kumar B., Singh K. Testing UML Designs Using Class, Sequence and Activity Diagrams // International Journal for Innovative Research in Science and Technology. – 2015. – Vol. 2. – No. 3. pp. 71–81.
20. Karamjit Kaur, Rinkle Rani. Modeling and Querying Data in NoSQL Databases. Big Data, 2013 IEEE International Conference. pp. 1–7. DOI:10.1109/BigData.2013.6691765
21. Pinheiro P. V. P., Endo A. T., Simao A. Model-Based Testing of RESTful Web Services Using UML Protocol State Machines // Brazilian Workshop on Systematic and Automated Software Testing. – 2013.
22. Pokorny J. NoSQL databases: a step to database scalability in web environment // International Journal of Web Information Systems. – 2013. – Vol. 9. – No. 1. pp. 69–82.

23. Wu C. S., Huang C. H. The web services composition testing based on extended finite state machine and UML model // Service Science and Innovation (ICSSI), 2013 Fifth International Conference on. – IEEE, 2013. pp. 215–222.
24. Zhang S. Application of document-oriented NoSQL database technology in web-based software project documents management system // Information Science and Technology (ICIST), 2013 International Conference on. – IEEE, 2013. pp. 504–507.
25. Bugiotti, F., Cabibbo, L., Atzeni, P., & Torlone, R. (2014). Database design for NoSQL systems. In Conceptual Modeling (pp. 223-231). Springer International Publishing.
26. Schram, A., & Anderson, K. M. (2012). MySQL to NoSQL. In Proceedings of the 3rd annual conference on Systems, programming, and applications: software for humanity-SPLASH (Vol. 12).
27. Nagni, M., Ventouras, S., & Parton, G. A. (2012). Implementation of UML Schema to RDBM.
28. Tramontana, P., Amalfitano, D., & Fasolino, A. R. (2013, September). Reverse engineering techniques: From web applications to rich Internet applications. In Web Systems Evolution (WSE), 2013 15th IEEE International Symposium on (pp. 83-86). IEEE.
29. Daniel, F., & Matera, M. (2014). Model-Driven Software Development. In Mashups (pp. 71-93). Springer Berlin Heidelberg.
30. de Lara, J., Guerra, E., & Cuadrado, J. S. (2015). Model-driven engineering with domain-specific meta-modelling languages. Software & Systems Modeling, 14(1), 429-459.
31. Martins, B. F., & Souza, V. E. S. (2015, October). A Model-Driven Approach for the Design of Web Information Systems based on Frameworks. In Proceedings of the 21st Brazilian Symposium on Multimedia and the Web (pp. 41-48). ACM.

References

1. Sadalage, P. J., Fowler, M. NoSQL Distilled: a Brief Guide to the Emerging World of Polyglot Persistence. Addison-Wesley (2012).
2. Mironov, V.V., Yusupova, N.I., Shakirova, G.R.: Situation-Oriented Databases: Concept, Architecture, XML Realization. Vestnik UGATU, vol. 14, no. 2 (37), pp. 233–244 (2010) (in Russian)
3. Mironov, V.V., Malikova, K.E.: Internet Applications Based on Embedded Dynamic Models: Architecture, Data structure, Interpretation. Vestnik UGATU, vol. 14, no. 1 (36), pp. 154–163 (2010) (in Russian)
4. Mironov, V. V., Malikova, K.E.: Internet Applications Based on Embedded Dynamic Models: User Interface Controls. Vestnik UGATU, vol. 14, no. 5 (40), pp. 170–175 (2010) (in Russian)
5. Mironov, V.V., Gusarenko, A.S.: Situation-Oriented Databases: the Concept of Managing XML Data Based on Dynamic DOM Objects. Vestnik UGATU, vol. 16, no. 3 (48), pp. 159–172 (2012) (in Russian)
6. Mironov, V.V., Gusarenko, A.S.: Dynamic DOM Objects in Situation-Oriented Databases: Lingware and Knoware of Data Sources. Vestnik UGATU, vol. 16, no. 6 (51), pp. 167–176 (2012) (in Russian)
7. Mironov, V.V., Kanashin, V.V.: Hierarchical Widgets: User Interface Organization in Web Applications on the Basis of Situation-Oriented Databases. Vestnik UGATU, vol. 17, no. 2 (55), pp. 138–149 (2013) (in Russian)
8. Mironov, V.V., Kanashin V.V.: Hierarchical Widgets: User Data Control Algorithms in Web Applications on the Basis of Situation-Oriented Databases. Vestnik UGATU, vol. 18, no. 1 (62), pp. 204–213 (2014) (in Russian)
9. Makarova, E.S., Mironov, V.V.: Web OLAP Conceptual Data Model Design on the Basis of Situation-Oriented Database. Vestnik UGATU, vol. 16, no. 6 (51), pp. 177–188 (2012) (in Russian)
10. Makarova, E.S., Mironov, V.V.: Analytical Functions in Web Applications Based on Situation-Oriented Databases. Vestnik UGATU, vol. 17, no. 5 (58), pp. 150–165 (2013) (in Russian)
11. Gusarenko, A.S., Mironov, V.V.: Smarty-Objects: Use Case of Heterogeneous Sources in Situation-Oriented Databases. Vestnik UGATU, vol. 18, no. 3 (63), pp. 242–252 (2014) (in Russian)
12. Gusarenko, A.S., Mironov, V.V.: Using of RESTful Services in Situation-Oriented Databases. Vestnik UGATU, vol. 19, no. 1 (67), pp. 204–211 (2015) (in Russian)
13. Aguilar J. A. et al. An Analysis of Techniques and Tools for Requirements Elicitation in Model-Driven Web Engineering Methods // Computational Science and Its Applications--ICCSA 2015. – Springer International Publishing, 2015. pp. 518-527.
14. Rumpe B. Executable Modeling with UML. A Vision or a Nightmare? // arXiv preprint arXiv:1409.6597. – 2014.
15. Agustin J. L. H., Del Barco P. C. A model-driven approach to develop high performance web applications //Journal of Systems and Software. – 2013. – Vol. 86. – No. 12. pp. 3013–3023.
16. Delgado A., Marotta A., González L. Towards the construction of quality-aware Web Warehouses with BPMN 2.0 Business Processes //Research Challenges in Information Science (RCIS), 2014 IEEE Eighth International Conference on. – IEEE, 2014. pp. 1–6.
17. Delgado A., Marotta A. Automating the process of building flexible Web Warehouses with BPM Systems //Computing Conference (CLEI), 2015 Latin American. – IEEE, 2015. pp. 1–11.
18. Aguilar J. A. et al. An Analysis of Techniques and Tools for Requirements Elicitation in Model-Driven Web Engineering Methods //Computational Science and Its Applications--ICCSA 2015. – Springer International Publishing, 2015. pp. 518–527.
19. Kumar B., Singh K. Testing UML Designs Using Class, Sequence and Activity Diagrams // International Journal for Innovative Research in Science and Technology. – 2015. – Vol. 2. – No. 3. pp. 71–81.
20. Karamjit Kaur, Rinkle Rani. Modeling and Querying Data in NoSQL Databases. Big Data, 2013 IEEE International Conference. pp. 1–7. DOI:10.1109/BigData.2013.6691765
21. Pinheiro P. V. P., Endo A. T., Simao A. Model-Based Testing of RESTful Web Services Using UML Protocol State Machines // Brazilian Workshop on Systematic and Automated Software Testing. – 2013.
22. Pokorny J. NoSQL databases: a step to database scalability in web environment // International Journal of Web Information Systems. – 2013. – Vol. 9. – No. 1. pp. 69–82.
23. Wu C. S., Huang C. H. The web services composition testing based on extended finite state machine and UML model // Service Science and Innovation (ICSSI), 2013 Fifth International Conference on. – IEEE, 2013. pp. 215–222.
24. Zhang S. Application of document-oriented NoSQL database technology in web-based software project documents management system // Information Science and Technology (ICIST), 2013 International Conference on. – IEEE, 2013. pp. 504–507.
25. Bugiotti, F., Cabibbo, L., Atzeni, P., & Torlone, R. (2014). Database design for NoSQL systems. In Conceptual Modeling (pp. 223-231). Springer International Publishing.

26. Schram, A., & Anderson, K. M. (2012). MySQL to NoSQL. In Proceedings of the 3rd annual conference on Systems, programming, and applications: software for humanity-SPLASH (Vol. 12).
27. Nagni, M., Ventouras, S., & Parton, G. A. (2012). Implementation of UML Schema to RDBM.
28. Tramontana, P., Amalfitano, D., & Fasolino, A. R. (2013, September). Reverse engineering techniques: From web applications to rich Internet applications. In Web Systems Evolution (WSE), 2013 15th IEEE International Symposium on (pp. 83-86). IEEE.
29. Daniel, F., & Matera, M. (2014). Model-Driven Software Development. In Mashups (pp. 71-93). Springer Berlin Heidelberg.
30. de Lara, J., Guerra, E., & Cuadrado, J. S. (2015). Model-driven engineering with domain-specific meta-modelling languages. Software & Systems Modeling, 14(1), 429-459.
31. Martins, B. F., & Souza, V. E. S. (2015, October). A Model-Driven Approach for the Design of Web Information Systems based on Frameworks. In Proceedings of the 21st Brazilian Symposium on Multimedia and the Web (pp. 41-48). ACM.

Submitted 06.10.2016

Об авторах:

Миронов Валерий Викторович, проф. каф. автоматизированных систем управления. Дипл. радиофизик (Воронежск. гос. ун-т, 1975). Д-р техн. наук по упр. в техн. системах (УГАТУ, 1995). Иссл. в обл. иерархических моделей и ситуационного управления;

Гусаренко Артем Сергеевич, Старший преподаватель каф. автоматизированных систем управления. Дипл. информатик-экономист (УГАТУ, 2010). Канд. техн. наук по мат. и прогр. обеспечению выч. машин, комплексов и компьютерных сетей (УГАТУ, 2013). Иссл. в обл. ситуационно-ориентированных баз данных;

Юсупова Нафиса Исламовна, декан фак. информатики и робототехники. Дипл. радиофизик (Воронежск. гос. ун-т, 1975). Д-р техн. наук по упр. в техн. системах (УГАТУ, 1997). Иссл. в обл. иерархических моделей, ситуационного управления в техн. и соц. системах.