УДК 004.65 DOI: 10.26102/2310-6018/2020.29.2.013

Ситуационно-ориентированные базы данных: формирование персонализированных графических документов для поддержки учебного проектирования

В.В. Миронов, А.С. Гусаренко, Г.А. Тугузбаев

Уфимский государственный авиационный технический университет,

Уфа, Россия

Резюме: Рассматривается задача создания персонализированных заготовок для проектных документов в форматах офисной графики. Под персонализированными заготовками понимаются документы, заполненные конкретными проектными данными, чтобы освободить разработчика от рутинных действий при последующем проектировании. Отмечается два уровня сложности графической персонализации: параметрический и структурный. Формирование персонализированных заготовок выполняется в два этапа: разработка шаблона; персонализация шаблона. На первом этапе в среде графического редактора вручную разрабатывается шаблон заготовки с предварительной разметкой точек персонализации. На втором этапе выполняется программная обработка шаблона, при которой в шаблоне отыскиваются точки персонализации и в них размещаются персональные данные из базы данных. Обсуждается персонализация на основе ситуационно-ориентированных баз данных — интегратора разнородных данных на основе информационного процессора, управляемого встроенной высоко абстрактной иерархической ситуационной моделью. Доступ к разнородным данным задается в ситуационной модели в виде виртуальных документов, которые отображаются на разнородные реальные данные. Рассматриваются особенности отображения виртуального документа на документы в форматах VDX и FODG, а также VSDX и ODG. Если в первом случае требуется отображение на XML-файл, то во втором — на ZIP-архив, в папках которого размещены XML-файлы. Рассматриваются фрагменты ситуационных моделей, обеспечивающие персонализацию на основе: поиска в дереве XML-документа узлов, содержащих идентификационные метки, и замены их данными базы. В сравнении с традиционным подходом это дает более простое задание процесса. Обсуждается практическое использование результатов для информационной поддержки учебного проектирования по дисциплине «Базы данных». Отмечается снижение трудоемкости выполнения рутинной части проекта, увеличение возможностей творческой деятельности в процессе учебного проектирования.

Ключевые слова: персонализированные документы, ситуационно-ориентированная база данных, иерархическая ситуационная модель, виртуальный документ, VDX, VSDX, ODG, FODG.

Для цитирования: Миронов В.В., Гусаренко А.С., Тугузбаев Г.А. Ситуационно-ориентированные базы данных: формирование персонализированных графических документов для поддержки учебного проектирования. *Моделирование, оптимизация и информационные технологии.* 2020;8(2). Доступно по: https://moit.vivt.ru/wp-content/uploads/2020/05/MironovSoavtors_2_20_1.pdf DOI: 10.26102/2310-6018/2020.29.2.013

Situation-oriented databases: the formation of personalized graphic documents for educational design support

V.V. Mironov, A.S. Gusarenko, G.A. Tuguzbaev

Ufa State Aviation Technical University, Ufa, Russia

Abstract: There is considered the task of creating personalized blanks for project documents in office graphics formats. Personalized blanks are understood as documents filled with specific design data in order to free the developer from routine actions during subsequent design. Two levels of graphic personalization complexity are noted: parametric and structural. The formation of personalized blanks is carried out in two stages: template development; template personalization. At the first stage, a blank template with preliminary marking of personalization points is manually developed in the environment of a graphical editor. At the second stage, software processing of the template is performed, in which personalization points are found in the template and personal data from the database is placed in them. Personalization based on situationally-oriented databases is discussed — an integrator of heterogeneous data based on an information processor controlled by a built-in highly abstract hierarchical situational model. Access to heterogeneous data is specified in the situational model in the form of virtual documents that are mapped onto heterogeneous real data. The features of mapping a virtual document to documents in VDX and FODG, as well as VSDX and ODG are considered. If in the first case, mapping to an XML file is required, then in the second to a ZIP archive, in the folders of which XML files are located. Fragments of situational models are considered that provide personalization based on: searching nodes containing identification tags in the XML document tree and replacing them with database data. Compared to the traditional approach, this gives a simpler task definition. The practical use of the results for information support of educational design in the discipline of "database" is discussed. There is a decrease in the complexity of the routine part of the project, an increase in the possibilities of creative activity in the process of educational design.

Keywords: personalized documents, situation-oriented database, hierarchical situational model, virtual document, VDX, VSDX, ODG, FODG.

For citation: Mironov V.V., Gusarenco A.S., Tuguzbaev G.A. Situation-oriented databases: the formation of personalized graphic documents for educational design support. *Modeling, Optimization and Information Technology.* 2020;8(2). Доступно по: https://moit.vivt.ru/wp-content/uploads/2020/05/MironovSoavtors_2_20_1.pdf DOI: 10.26102/2310-6018/2020.29.2.013 (In Russ).

ВВЕДЕНИЕ

Построение графических документов (инженерная и инфографика) является обязательным требованием в процессе учебного проектирования во многих технических и социально-экономических предметных областях [1–3]. Популярными инструментами при создании векторной графики, диаграмм, блок-схем являются офисные графические редакторы, такие как проприетарный Microsoft Office Visio и свободно распространяемые OpenOffice Draw, LibreOffice Draw и др. Они широко используются в учебном процессе как инструменты графического дизайна в самых разных учебных дисциплинах [4–7].

Для выполнения учебного проекта студентам обычно выдаются заготовки графических документов, на основе которых они создают решения с учетом своего индивидуального задания. Это сокращает трудоемкость выполнения рутинной части процесса проектирования. Заготовки, как правило, выполняются вручную в среде графического редактора.

Персонализация обучения, в том числе проектной деятельности студентов, — актуальный современный тренд [8–10]. Персонализация графических документов дает возможность освобождения студентов от рутинных действий в ходе учебного проектирования. Возможность этого связана с появлением открытых графических форматов на основе XML, допускающих программную обработку документов [9]. Персонализация заключается в том, что в заготовки проектных документов программным путем вносятся данные, хранящиеся в базе данных и отражающие конкретные особенности выполняемого проекта (Рисунок 1). Это могут быть:

- спецификации и реквизиты выполняемого варианта проекта;
- персональные данные разработчика, которые должны присутствовать в разрабатываемом документе;
- результаты предшествующих этапов выполнения проекта, которые должны присутствовать в разрабатываемом документе или могут служить основой для разработки текущего этапа.

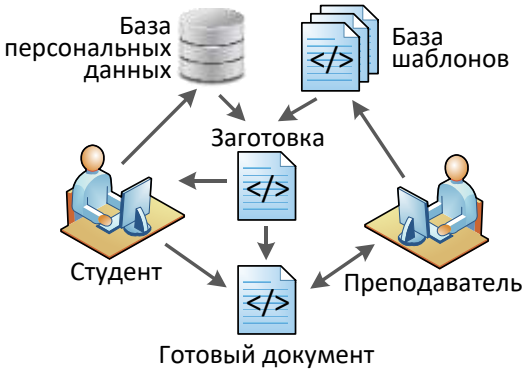

Рисунок 1 — Использование персонализированных заготовок графических документов в процессе учебного проектирования

Figure 1 — The use of personalized blanks of graphic documents in the process of educational design

Формирование персонализированных заготовок выполняется в два этапа: 1) разработка шаблона; 2) персонализация шаблона. На первом этапе в среде графического редактора разрабатывается шаблон заготовки с предварительной разметкой точек персонализации. На втором этапе выполняется программная обработка шаблона, при которой в шаблоне отыскиваются точки персонализации и в них размещаются данные из базы персональных данных.

Необходимо отметить два уровня персонализации и, соответственно, два уровня сложности задачи, что относится не только к графическим, но и к другим видам документов (текстовым, мультимедийным и др.).

- *Параметрическая*. Корректируются параметры фигур, которые уже размещены в шаблоне заготовки. Например, вносится текст, связанный с фигурой, или устанавливаются параметры, определяющие ее размеры и положение.
- *Структурная*. В заготовке размещают ся новые фигуры, которых изначально там не было. Например, вносятся графические фрагменты и фигуры, разработанные на предшествующих этапах и необходимые как основа для разработки графики текущего этапа. Отметим, что это более сложная задача, требующая более сложных программных манипуляций с шаблоном и извлекаемыми из базы графическими фрагментами.

Здесь рассматривается, в основном, задача параметрической персонализации. Другая особенность рассматриваемого подхода заключается в том, что решение задачи персонализации выполняется в рамках ситуационно-ориентированных баз данных — интегратора разнородных данных, над которым работают авторы.

СИТУАЦИОННО-ОРИЕНТИРОВАННЫЕ БАЗЫ ДАННЫХ И ИЕРАРХИЧЕСКАЯ СИТУАЦИОННАЯ МОДЕЛЬ

Ситуационно-ориентированные базы данных представляют собой проект, который обеспечивает интеграцию разнородных данных на основе информационного

процессора, управляемого встроенной высоко абстрактной иерархической ситуационной моделью [11].

Иерархическая ситуационная модель (HSM — Hierarchical Situation Model) задает функционирование ситуационно-ориентированной базы данных. Для этого HSM циклически обрабатывается HSM-интерпретатором, который отслеживает ее текущие состояния и сохраняет их между циклами обработки. HSM имеет две эквивалентные формы представления: графическую, задающую иерархическую структуру элементов и ориентированную на восприятие человеком, и текстовую, использующую XML-подобный синтаксис и ориентированную на программную обработку интерпретатором. Здесь используется графическая форма моделей HSM.

На Рисунке 2 приведена общая структура HSM-модели. Модель HSM в целом представляет собой совокупность субмоделей `sub`, организованных в виде иерархии. Каждая субмодель содержит множество элементов состояний `sta`, которые, в свою очередь, в качестве вложенных могут содержать субмодели, а также такие элементы как переходы `jmp` и акции `act`. Предусмотрены также специальные виды акций: виртуальные документы `doc` и объекты обработки `dpo`.

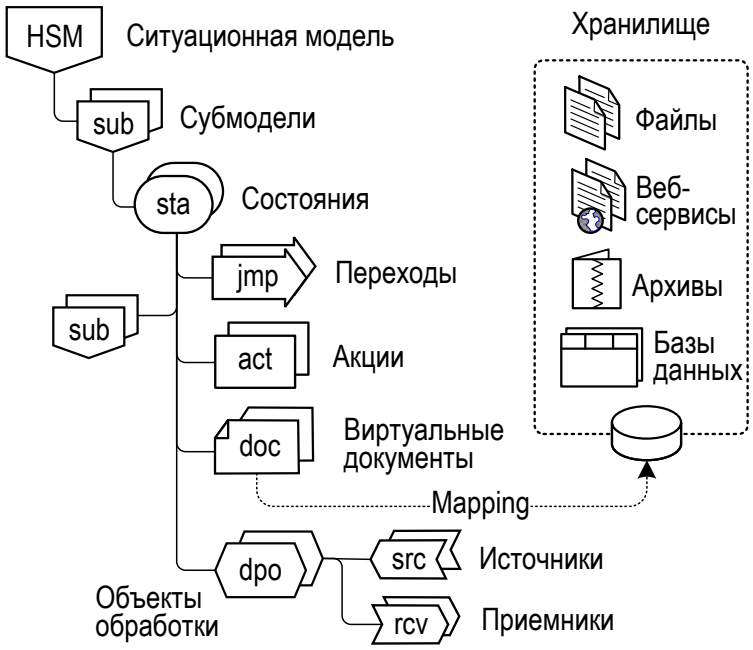

Рисунок 2 — Структура иерархической ситуационной модели ситуационно-ориентированной базы данных

Figure 2 — The structure of the hierarchical situational model of a situation-oriented database

Управление текущим состоянием. Интерпретация модели HSM выполняется циклически или по запросу в рамках сеанса. Первый цикл обработки субмодели в пределах сеанса начинается с начального состояния этой субмодели. В процессе цикла текущее состояние субмодели может изменяться с помощью элементов-переходов. Текущее состояние субмодели по завершению цикла сохраняется и используется для начала обработки на следующем цикле сеанса. При обработке текущего состояния могут выполняться действия, задаваемые акциями и связанные с обработкой внешних данных.

Концепция виртуальных документов. Доступ к внешним данным задается в ситуационной модели с помощью виртуальных документов, отображаемых

на разнородные реальные данные [12–15]. Элементы `doc` задают виртуальные документы и определяют реальные данные, на которые эти реальные документы отображаются. На Рисунке 2 показаны виды внешних данных, доступные для отображения: локальные и удаленные файлы, веб-сервисы, архивы, реляционные базы данных [16–20]. Обработка данных выполняется с помощью объектов обработки. Элементы-источники `src` обеспечивают загрузку данных в память на основании спецификаций отображения. Собственно обработка загруженных данных может задаваться в источниках `src` и приемниках `rcv`. Вывод результатов обработки осуществляется также с помощью элементов-приемников `rcv` на основе спецификаций отображения виртуальных документов. Таким образом, чтобы обработать некоторые данные, нужно в модели HSM, во-первых, задать виртуальный документ, отображаемый на соответствующие исходные данные, во-вторых, задать объект обработки, определить загрузку в него данных виртуального документа, обработку загруженных данных и выгрузку результатов обработки.

Отображение на графические документы. Применительно к рассматриваемой задаче следует учитывать, что в настоящее время для файлов графических документов на основе XML могут применяться две категории форматов:

- документ в виде единого («плоского») XML-файла. Типичным примером является формат VDX — XML-файл, задающий диаграмму Visio на языке DatadiagramML (http://m8y.org/Microsoft_Office_2003_XML_Reference_Schemas/Help/html/vixmlconSchema.htm). Этот формат применяется в версиях Microsoft Visio 2010 и более ранних, пока он сохраняет свою актуальность ввиду широкого распространения этих версий. Другой пример — формат FODG (Flat ODG), применяемый, например, в редакторе LibreOffice Draw;
- документ в виде ZIP-архива, в папках которого упакованы XML-файлы, задающие различные аспекты изображения. Примером может служить формат VSDX — диаграмма Visio в соответствии со стандартом OPC — Open Packaging Conventions (http://www.ecma-international.org/publications/standards/Ecma-376.htm). Этот формат применяется в версиях Microsoft Visio 2013 и более поздних. Другой пример — формат ODG (Open Document Graphics), соответствующий стандарту ODF — Open Document Format (http://standards.iso.org/ittf/PubliclyAvailableStandards/c066363_ISO_IEC_26300-1_2015.zip) и применяемый в редакторах Apache OpenOffice Draw и LibreOffice Draw.

В зависимости от категории используемого формата программная обработка графических документов должна организовываться по-разному: в первом случае требуется отображение виртуального документа непосредственно на XML-файл, а во втором — на XML-файлы, упакованные в ZIP-архиве. В первом случае нужно знать внутреннюю структуру XML-файла, а во втором — внутреннюю структуру как ZIP-архива графического документа, так и интересующих XML-файлов в нем.

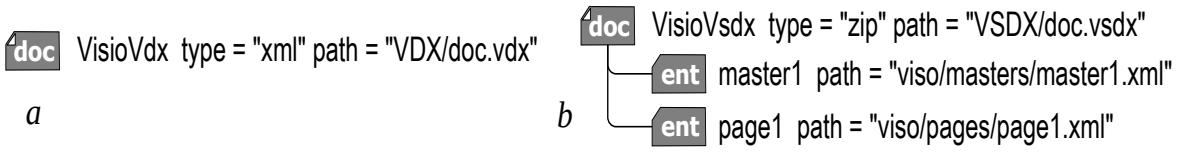

Рисунок 3 — Задание виртуального документа в модели HSM с отображением на документ Visio в форматах VDX (*a*) и VSDX (*b*)

Figure 3 — Defining a virtual document in the HSM model with mapping to a Visio document in VDX (*a*) and VSDX (*b*) formats

Рисунок 3 иллюстрирует задание виртуального документа, отображаемого на внешний графический документ, для этих двух случаев. В варианте *a* виртуальный документ `doc:VisioVdx` отображается на файл `doc.vdx`, находящийся в папке `VDX`, а в варианте *b* — документ `doc:VisioVsdx` на файл `doc.vsdx`, находящийся в папке `VSDX`. При этом во втором варианте отображение задано как отображение на ZIP-архив и в нем с помощью вложенных элементов `ent:master1` и `ent:master2` заданы два пути доступа к содержимому архива: к файлу `master1.xml`, лежащему в папке `visio/masters`, и к файлу `page1.xml`, лежащему в папке `visio/pages`.

Рисунок 4 иллюстрирует задание в HSM-модели обработки виртуального документа. Обрабатываемый виртуальный документ загружается в объект обработки с помощью элементов `dom:VisioVdx-Proc` и `dom:VisioVsdx-Proc`. В данном случае объект обработки — это объект DOM Document Object Model — «объектная модель документа» — программно-независимый интерфейс, предоставляющий доступ к дереву XML-документа и позволяющий программе изменять XML-документ (W3C. Document Object Model (DOM) Level 1 Specification. https://www.w3.org/tr/rec-dom-level-1/). Загружаемый документ указывается с помощью атрибута `srcDoc` — ссылки на виртуальный документ. В случае VDX-формата (Рисунок 4, *a*) в DOM-объект загружается целиком XML-файл документа Visio, а в случае VSDX-формата (Рисунок 4, *b*) — XML-файл из архива документа Visio (в данном случае — файл `page1.xml`). Атрибут `saveDoc`, ссылающийся на тот же виртуальный документ, предписывает по завершении обработки сохранить измененный XML-файл в ZIP-архиве на прежнем месте.

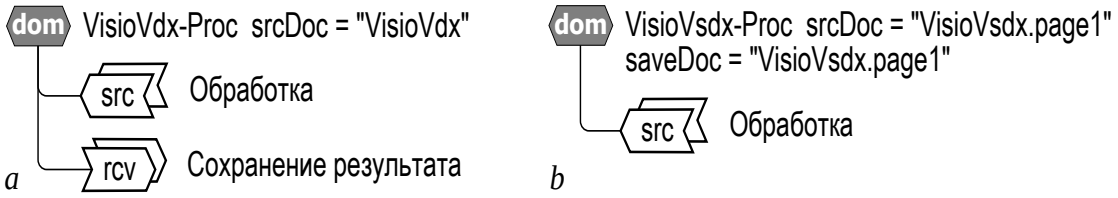

Рисунок 4 — Общая схема обработки виртуального документа в модели HSM для форматов VDX (*a*) и VSDX (*b*)

Figure 4 — General scheme for processing a virtual document in the HSM model for VDX (*a*) and VSDX (*b*) formats

Обработка графического шаблона в формате «плоского» XML-файла проще, чем в формате ZIP-архива. XML-файл, будучи целиком загружен в DOM-объект и персонализирован в нем, может быть сразу же отправлен адресату (студенту). ZIP-архив должен обрабатываться по частям, обработанные части должны быть возвращены в архив на прежнее место, чтобы не нарушить целостность документа. В этих условиях приходится делать копию исходного шаблона, подвергать ее последовательной персонализации по частям и после этого отправлять адресату. Например, затруднения могут возникнуть, если потребуется совместная обработка нескольких частей документа.

Собственно обработка на Рисунке 4 не детализирована, обозначены лишь выполняющие ее элементы — источники и приемники. Детали поясняются ниже.

ПОДГОТОВКА ШАБЛОНОВ И ИХ ОБРАБОТКА В ХОДЕ ПАРАМЕТРИЧЕСКОЙ ПЕРСОНАЛИЗАЦИИ

Формирование персонализированных заготовок выполняется в два этапа: 1) разработка шаблона; 2) персонализация шаблона. На первом этапе в среде графического редактора вручную разрабатывается шаблон заготовки с предварительной разметкой точек персонализации. На втором этапе выполняется программная обработка шаблона, при которой в шаблоне отыскиваются точки персонализации и размещаются персональные данные из базы данных.

Шаблон персонализации. Шаблон представляет собой графический документ, содержащий фрагменты результирующего изображения. Изображение строится из логических элементов — фигур. Фигуры, содержащиеся в шаблоне, могут в том или ином виде присутствовать в результирующем документе. В ходе учебного проектирования студент может их настроить в плане размеров, расположения, соединения, поясняющих надписей и т. д.

В документах Visio фигуры могут присутствовать в двух видах: в виде собственно изображения, размещенного на листах документа, и в виде фигур-образцов (stencils — трафаретов), которые прикреплены к документу и могут использоваться для построения собственно изображения. В документах OpenOffice/LibreOffice Draw прикрепленные фигуры-образцы не предусмотрены, есть только фигуры изображения.

Использование идентификационных меток

При параметрической персонализации, рассматриваемой в этой статье, необходимо вручную задать в имеющихся фигурах шаблона (в изображении и/или в прикрепленном наборе) точки персонализации в виде идентифицирующих меток, которые позволили бы однозначно отыскать фигуру и соответствующим образом скорректировать ее. С каждой фигурой может быть ассоциирован некоторый текст. Проще всего пометить фигуру изображения, вручную размещая в ней уникальный текст. Обнаружение этого текста при программной обработке позволит идентифицировать и саму фигуру.

На Рисунке 5, *a* приведен пример идентификационных меток в основной надписи шаблона конструкторского документа. Слова «Шифр», «проект», «модель», «вуз», «студент», «препод», «12.12.12», «11.11.11» и др. представляют собой идентификационные метки, внесенные разработчиком в неизменяемое изображение шаблона. На этапе персонализации эти метки будут найдены в шаблоне заменены на персональные данные: шифр документа, название проекта, название вуза, фамилии и инициалы студента и преподавателя-консультанта, даты разработки и проверки и др. (Рисунок 5, *b*). Работая в среде графического редактора, разработчик может достаточно просто выбирать места внесения идентификационных меток и оформления замещаемых ими персональных данных. Устанавливая визуальные свойства меток (цвет, размер, шрифт и т. д.), разработчик тем самым задает свойства отображения соответствующих персональных данных.

a

					Шифр			
						Лит.	Масса	Масштаб
Изм.	Лист	№ докум.	Подпись	Дата	проект	л1 л2		
Разработал		студент		12.12.12				
Проверил		препод		11.11.11		Лист 1	Листов 3	
					модель	ВУЗ		

b

					2020-1.6.БД.КП.ЭАС-309.17130026			
						Лит.	Масса	Масштаб
Изм.	Лист	№ докум.	Подпись	Дата	Разработка концептуально-логических моделей базы данных бизнес-процесса	Л И		
Разработал		Яковлев И.А.		17.02.20				
Проверил		Миронов В.В.		10.03.20		Лист 1	Листов 3	
					Модель локальная иерархическая	ФГБОУ ВО «УГАТУ»		

Рисунок 5 — Пример размещения идентификационных меток в основной надписи документа (*a*) и результат персонализации (*b*)

Figure 5 — An example of the placement of identification tags in the title block of the document (*a*) and the result of personalization (*b*)

Программная обработка шаблонов с целью параметрической персонализации требует учета внутренней организации графических документов. Чтобы отыскать в шаблоне идентификационную метку и подставить вместо нее персональное значение, нужно знать внутреннюю структуру документа. Изучение доступных сведений о внутреннем строении форматов VDX/VSDX и ODG/FODG, а также непосредственное исследование XML-разметки графических документов с помощью текстовых XML-редакторов позволило выявить сведения об особенностях различных графических форматов, существенные в плане решаемой задачи.

Документы в виде «плоского» XML-файла

На Рисунке 6 приведены модели структуры графических документов в форматах VDX и FODG. Представленное дерево XML-разметки, содержит иерархию только тех XML-элементов и их атрибутов, которые имеют отношение к тексту, ассоциированному с фигурами изображения, что существенно для рассматриваемой задачи параметрической персонализации. Чтобы не загромождать картину, опущены элементы и атрибуты разметки, задающие несущественные для решаемой задачи свойства фигур, такие как положение на листе, размеры, цвета и т. п. В применяемой графической нотации имена XML-элементов заданы шрифтом с подчеркиванием, XML-атрибутов — без подчеркивания, текстовое содержимое XML-элемента обозначено символом равенства. Вложенность XML-элементов показана с помощью коннекторов «вниз-вправо», светлый треугольник указывает возможность ноля, одного или нескольких экземпляров, а светлый кружок — возможность ноля или одного экземпляра вложенного (дочернего) элемента.

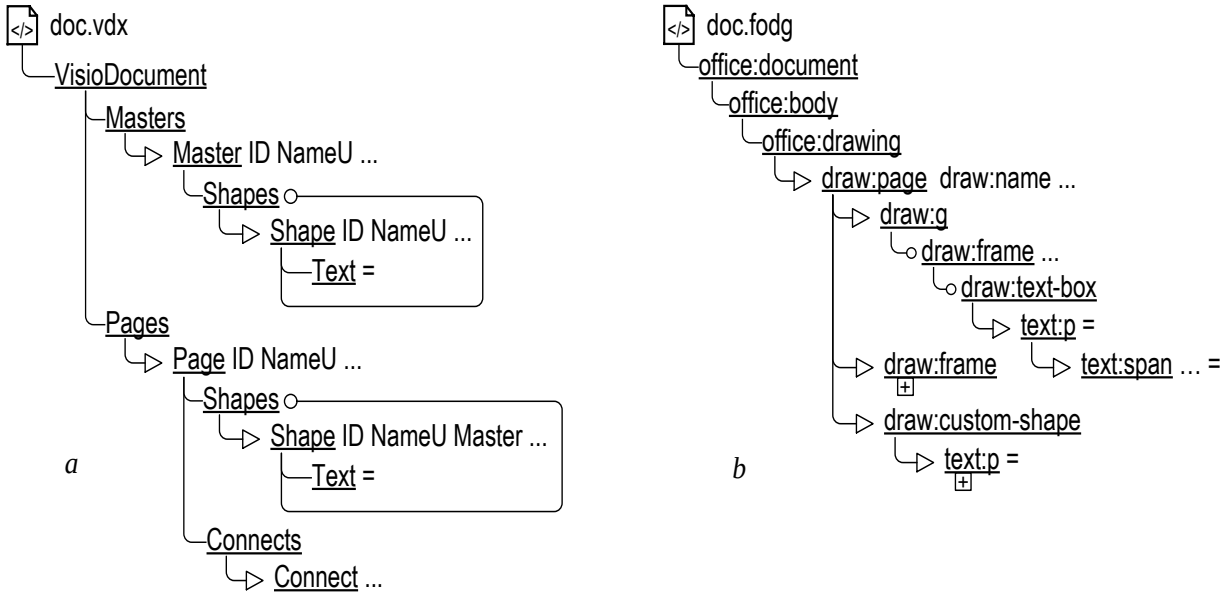

Рисунок 6 — Модель XML-разметки документов VDX (*a*) и FODG (*b*)
Figure 6 — XML markup model for VDX (*a*) and FODG (*b*) documents

Шаблон в формате VDX. Документ Visio в формате VDX (Рисунок 6, *a*) представляет собой XML-документ с корневым XML-элементом `VisioDocument`. Для рассматриваемой задачи важны два его дочерних XML-элемента: `Masters` и `Pages`. Элемент `Masters` задает набор фигур-образцов (мастер-фигур, фигур-трафаретов), а элемент `Pages` задает листы документа Visio, на которых размещается собственно изображение. Фигуры-образцы могут использоваться для построения изображения на листах документа. При изменении свойств фигур-образцов изменяются используемые по умолчанию соответствующие свойства фигур изображения на листах, основанные на этих образцах.

XML-элемент `Masters` содержит дочерние элементы `Master`, каждая из которых задает отдельную фигуру-образец. XML-атрибут `ID` содержит идентификатор образца, атрибут `NameU` — его имя (остальные атрибуты опущены). Дочерний XML-элемент Shapes задает множество фигур образца. Каждая из этих фигур задается своим XML-элементом `Shape`, вложенным в элемент `Shapes`. XML-атрибут `ID` задает идентификатор фигуры, атрибут `NameU` — имя фигуры. Остальные атрибуты, задающие многочисленные свойства фигуры, на рисунке не показаны. Не показаны также многочисленные дочерние XML-элементы, за исключением элемента `Text`, задающего текст, ассоциированный с фигурой.

Отметим рекурсивный характер структуры фигур: сложная фигура может включать в себя несколько вложенных фигур, которые, в свою очередь, могут содержать другие вложенные фигуры. Это отражено в модели с помощью коннектора, прикрепляющего вышестоящий элемент `Shapes` в качестве (необязательного) ребенка к нижестоящему элементу `Shape`.

XML-элемент `Pages` содержит дочерние элементы `Page`, каждая из которых задает отдельный лист документа. XML-атрибут ID содержит идентификатор листа, атрибут `NameU` — его имя (остальные атрибуты опущены). Дочерний XML-элемент `Shapes` задает

множество фигур, размещенных на листе. Элемент `Shapes` имеет то же внутреннее строение, что и в элементе `Masters`.

Отметим, что у этих XML-элементов `Shape` может быть дополнительный атрибут-ссылка `Master`. Он содержит значение идентификатора `ID`, указывающего на элемент `Master`, если данная фигура на листе была получена из соответствующей фигуры-образца. Следует учитывать, что в этом случае элемент `Shape` задает только те свойства фигуры на листе, которые отличаются от свойств фигуры-образца.

Элемент `Page` содержит также вложенный элемент `Connections`, который задает множество соединений фигур изображения на листе. Каждое соединение описывается своим элементом `Connect`.

Таким образом, идентификационная метка представляет собой текстовое содержимое некоторого XML-элемента `Text`, вложенного в некоторый XML-элемент `Shape`, который, в свою очередь, вложен в родительский элемент `Shapes`. Этот элемент является потомком элемента `Master`, если метка размещена в фигуре-образце, или потомком элемента `Page`, если метка размещена в фигуре на листе.

Шаблон в формате FODG. Документ в формате FODG (LibreOffice Draw и др., Рисунок 6, *b*) представляет собой XML-документ с корневым XML-элементом `office:document`. Для рассматриваемой задачи важен его элемент-потомок `office:drawing` со вложенными элементами `draw:page`, соответствующими отдельным листам документа (имя листа задается атрибутом `draw:name`). Фрагменты текста, присутствующие на листе, представляются в XML-разметке как текстовое содержимое элемента из пространства имен с префиксом «`text`»: элемента-абзаца `text:p` или элемента-интервала `text:span`, вложенного в элемент-абзац. Элемент-абзац, в свою очередь, может быть вложен в такие элементы, как `draw:text-box`, `draw:frame`, `draw:custom-shape`.

Таким образом, идентификационная метка в случае формата FODG, как и в случае формата VDX, представляет собой текстовое содержимое некоторого XML-элемента. Отметим, что в формате FODG отсутствует понятие прикрепленных к документу фигур-образцов. Фигуры, исполняющие эту роль, приходится размещать в заготовках на самих листах документа.

Документы в виде ZIP-архива

Распределенная структура графических документов обусловливает более сложную процедуру их персонализации. На Рисунке 7 представлена структура основных компонентов ZIP-архива документов в форматах VSDX и ODG.

Особенности шаблона в формате VSDX. На Рисунке 7, *a* представлена структура основных компонентов ZIP-архива документа в формате VSDX. Содержательная информация об изображении документа собрана в папке `visio`, где во вложенной папке `pages` размещены сведения об изображении на листах, а в папке `masters` — о прикрепленных к документу фигурах-образцах. Каждому листу документа соответствует свой XML-файл: `page1.xml`, `page2.xml`, …, кроме того, файл `pages.xml` содержит сведения, относящиеся ко всем листам. Аналогичным образом, каждому образцу документа соответствует свой XML-файл: `master1.xml`, `master2.xml`, …, кроме того, файл `masters.xml` содержит сведения, относящиеся ко всей совокупности образцов.

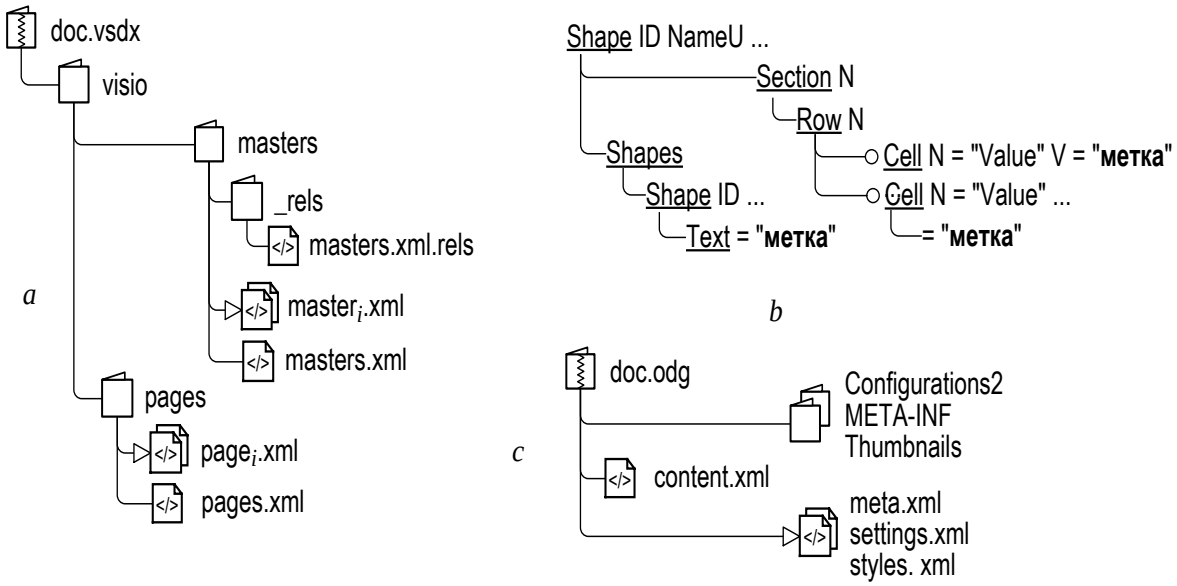

Рисунок 7 — Структура основных компонентов ZIP-архива документа в форматах VSDX (*a*, *b*) и ODG (c)

Figure 7 — Structure of the main components of a ZIP archive of a document in VSDX (*a*, *b*) and ODG (*c*) formats

Отметим, что если в фале `masters.xml` содержатся сведения общего характера по каждому образцу, то в файлах `master1.xml`, `master2.xml`, … размещены детальные сведения фигур-образцов. Так, именно в этих файлах будут размещены идентификационные метки фигур-образцов. Связь общих и детальных сведений фигур-образцов отражена в файле `masters.xml.rels`, который лежит в папке `visio/masters/_rels`.

Хотя имеются различия в XML-разметке форматов VSDX и VDX, внутренняя XML-разметка файлов `master1.xml`, `master2.xml`, … в целом сходна с XML-разметкой элемента `master`, а файлов `page1.xml`, `page2.xml`, … — на XML-разметку элемента `page` в формате VDX. Важной особенностью разметки в формате VSDX, которую нужно иметь в виду, является то, что текст идентификационной метки может представляться в XML-разметке документа как в виде текстового содержимого XML-элемента, так и в виде значения XML-атрибута. На Рисунке 7, *b* показано, как в элементе `Shape` связанный с фигурой текст «`метка`» может быть задан как значение атрибута `V` элемента `Cell`, как текстовое содержимое этого элемента, а также как текстовое содержимое элемента `Text`.

Особенности шаблона в формате ODG. На Рисунке 7, *c* представлена структура основных компонентов ZIP-архива документа в формате ODG. Интересующие нас сведения, задающие изображение, содержатся в файле `content.xml`, размещенном в архиве на верхнем уровне. Корневым XML-элементом в этом файле является элемент `office:document-content`, который содержит вложенный элемент `office:body` (см. Рисунок 6, *b*), идентичный одноименному элементу в формате FODG. Таким образом, в этом случае, как и в случае формата FODG, идентификационная метка соответствует текстовому содержимому соответствующего XML-элемента.

Адресация меток с помощью XPath-выражений

В процессе персонализации необходимо найти в документе XML-элемент (или XML-атрибут в случае формата VSDX), значение которого совпадает со значением идентификационной метки, и заменить это значение на новое. Первую часть этой задачи — задачу поиска — можно решить с помощью языка XPath (https:/ /web.archive.org/web/ 20121209085946/http://www.w3.org/TR/xpath/). XPath позволяет задавать выражения, адресующие определенные узлы в дереве XML-документа. Так, XPath-выражение

```
//* [text () = 'значение']
```

адресует все XML-элементы с заданным «значением» текстового содержимого. Для однозначной идентификации необходимо, чтобы значение идентификационной метки было уникальным в пределах всего документа. Выражение

```
//Master//Text [text () = 'значение']
```

адресует все XML-элементы `Text` с заданным текстовым содержимым, которые являются потомками элементов `Master`. То есть это выражение выполняет поиск среди фигур-образцов документа Visio. Для однозначной идентификации необходимо, чтобы значение идентификационной метки было уникальным в пределах всех фигур-образцов, прикрепленных к документу. Выражение

```
//Page [NameU = 'имя']//* [text () = 'значение']
```

адресует все XML-элементы с заданным «значением» текстового содержимого, которые являются потомком элемента Page с заданным «именем». То есть это выражение выполняет поиск среди фигур на заданном листе документа. Для однозначной идентификации необходимо, чтобы значение идентификационной метки было уникальным в пределах этого листа. Наконец, выражение

```
//* [text () = 'значение'] | //@* [. = 'значение']
```

адресует все XML-элементы и все XML-атрибуты с заданным значением. Этот вариант удобен для случая, когда заранее не известно, как представлена метка в документе — в виде элемента или в виде атрибута.

Требования к идентификационным меткам. При задании идентификационных меток в среде графического редактора необходимо обеспечить их уникальность в определенном диапазоне, а также атомарность, т. е. однородность формата текста метки. Последнее требование необходимо, чтобы в XML-дереве документа метка была представлена одним узлом, доступным для адресации достаточно простым XPath-выражением, а не была бы расчленена на несколько узлов с различным форматированием. Поэтому целесообразно задавать метки короткими словами без пробелов и управляющих символов.

Персонализация идентификационных меток

Итак, поиск идентификационных меток можно задавать с помощью XPath-выражений. Следующая задача — замещение найденных меток соответствующими персональными данными. Для реализации этой функциональности в HSM в составе DOM-элементов обработки, ориентированных на работу с DOM-объектами, были предусмотрены `src`-источники `updateNode`, обеспечивающие поиск в дереве XML-документа узлов, имеющих заданное значение, и установку нового значения.

На Рисунке 8 приведена HSM-модель DOM-объекта, обеспечивающая персонализацию идентификационных меток в документе в виде «плоского» XML-файла

в рамках общей схемы, представленной на Рисунке 4, *a*. В DOM-объект `dom:VisioVdx-Proc` загружен документ Visio в формате VDX в соответствии с Рисунком 4, *a*.

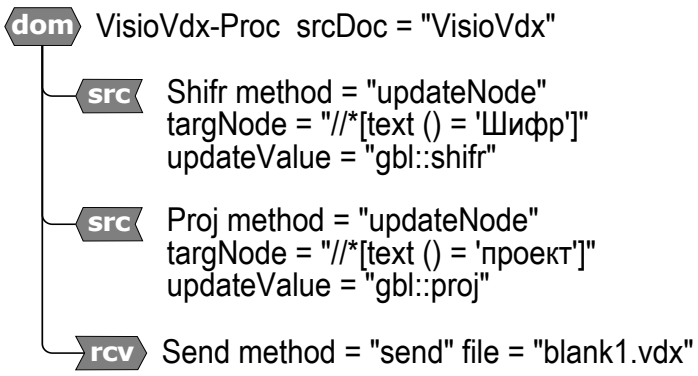

Рисунок 8 — Фрагмент HSM-модели персонализации документа в формате VDX
Figure 8 — A fragment of the HSM model for personalizing a document in VDX format

С помощью вложенных элементов-источников `src:Shifr` и `src:Proj` выполняется поиск в документе идентификационных меток и замена их на персональные данные. Затем с помощью элемента-приемника `rcv:Send` персонализированный документ отправляется пользователю.

На рисунке представлены два элемента-источника, обеспечивающие персонализацию двух идентификационных меток. Источник `src:Shifr` задает поиск и персонализацию метки "Шифр" (см. Рисунок 5). Атрибут `method = "updateNode"` предписывает обновление узла в дереве загруженного XML-документа. Атрибут `targNode` содержит XPath-выражение, адресующее обновляемый узел. Выражение `"//*[text () = 'Шифр']"` предписывает отыскать XML-элемент, у которого текстовое содержимое равно значению "Шифр". Атрибут `updateValue = "glb::shifr"` задает новое значение обновляемого узла как значение глобальной переменной `shifr`. Источник `src:Proj`, устанавливающий значение метки "проект", и другие источники, не показанные на рисунке, построены аналогично.

Аналогичным образом выполняется обработка документа в виде ZIP-архива в рамках общей схемы, представленной на Рисунке 4, *b*.

Таким образом, предусмотренные возможность использования XPath-выражений для адресации XML-узлов и возможность замены текстовых значений узлов позволяет достаточно просто задавать в HSM-модели параметрическую персонализацию на основе идентификационных меток, единообразно для различных форматов графических документов, базирующихся на XML.

ПРИМЕНЕНИЕ В УЧЕБНОМ ПРОЦЕССЕ

Предложенный подход был реализован и испытан на исследовательском прототипе интерпретатора моделей HSM, выполненном на платформе PHP и размещенном на веб-сервере Уфимского государственного авиационного технического университета (УГАТУ). Практическое применение реализовано для информационной поддержки процесса проектирования на сайте курсового проектирования по дисциплине «Базы данных» (http://hsm.ugatu.su/artem/dbproj/).

Обслуживаемый курсовой проект предусматривает разработку концептуально-логических моделей базы данных для информационного обеспечения бизнес-процесса

в соответствии с индивидуальными заданиями. Проектирование включает 12 последовательных этапов, на 8 из которых предусмотрена разработка графических моделей разного уровня детализации: от локальных иерархических до глобальных реляционных. Для каждого этапа моделирования были разработаны шаблоны, размеченные в соответствии с изложенным выше подходом. В начале очередного этапа проектирования студенту предоставляется возможность загрузить с веб-сервера заготовки в формате VDX, персонализированные «на лету» с учетом исполнителя, варианта задания и выполняемого этапа. Персонализация включает, во-первых, заполнение персональными данными основной надписи конструкторских документов в соответствии со стандартом ЕСКД (см. рис. 5), во-вторых, размещение сведений с предыдущих этапов, которые могут быть полезными на данном этапе, такие как набор фигур-образцов, необходимых для построения модели. Получив персонализированную заготовку, студент с помощью графического редактора строит на ее основе требуемую модель и загружает на сервер. Там документ проходит проверку корректности программным путем и становится доступным для проверки преподавателем-консультантом.

Практическое применение системы в течение нескольких учебных семестров показало работоспособность и эффективность предложенного подхода. За счет персонализации заготовок графических документов у студентов сократилось время, затрачиваемое на техническую сторону подготовки проектной документации, и, соответственно, увеличилось время для собственно проектирования.

Дальнейшее развитие подхода. В текущей реализации системы обработка предусмотрена только для формата VDX, что диктует использование редактора Visio 2010. Планируется обеспечить возможность одновременной работы системы с различными графическими форматами, что даст возможность персонального выбора студентами применяемых графических редакторов.

В дальнейшем предполагается распространить предложенный подход на структурную персонализацию. При разработке моделей баз данных очередная модель основывается на предшествующих, что требует копирования ранее построенных фрагментов и фигур. Планируется проработать вопросы вычленения из графических документов содержательных сведений, необходимых для следующих этапов проектирования, и формирования персонализированных заготовок с размещением в них необходимых элементов изображения.

ЗАКЛЮЧЕНИЕ

Таким образом, в статье рассмотрено решение задачи персонализации графических документов на основе ситуационно-ориентированных баз данных, имеющее следующие особенности:

- графические документы представлены в XML-ориентированных форматах, как в виде «плоского» XML-файла (VDX, FODG), так и в виде ZIP-архива XML-файлов (VSDX, ODG);
- предварительная подготовка графического шаблона выполняется вручную в среде графического редактора с помощью текстовых идентификационных меток;
- программная обработка шаблона выполняется в DOM-объектах с помощью виртуальных документов, отображаемых в зависимости от формата на графический документ целиком или на его части в ZIP-архиве;
- адресация идентификационных меток в шаблоне задается XPath-выражениями, учитывающими особенности XML-разметки графических документов в различных форматах;

- работоспособность и эффективность предложенного подхода подтверждены путем практического использования при информационном сопровождении учебного проектирования концептуально-логических моделей баз данных.

БЛАГОДАРНОСТИ

Работа выполнена в рамках проекта развития ситуационно-ориентированных баз данных, поддержанного Российским фондом фундаментальных исследований, грант 19-07-00682.

ЛИТЕРАТУРА

1. Baswara S.Y., Widhiastuti R., Dewi L.C. Learning Model Based on Information Technology in an Accounting Education Courses Based on Technology at Faculty of Economics in Universitas Negeri Semarang. *KnE Social Sciences*. 2020;4(6):1280-1285. Available at: https://doi.org/10.18502/kss.v4i6.6678. (accessed 20.01.2020).
2. Yu Z., Xiong Z. Comparative Analyses for the Performance of Rational Rose and Visio in Software Engineering Teaching. *Journal of Physics: Conference Series. IOP Publishing.* 2018;1087(6):062041. DOI:10.1088/1742-6596/1087/6/062041
3. Medoh C., Telukdarie A. Business Process Modelling Tool Selection: A review. *2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM). IEEE*. 2017:524-528.
4. Sarvepalli A., Godin J. Business Process Management in the Classroom. *Journal of Cases on Information Technology (JCIT).* 2017;19(2):17-28.
5. He L., Lian J. Instructional Design of Practice Course of Logistics System Planning and Design Based on Visio. *2018 9th International Conference on Information Technology in Medicine and Education (ITME). IEEE*, 2018:526-530.
6. David J Parker. Mastering Data Visualization with Microsoft Visio Professional 2016. *Packt Publishing Ltd.* 2016.
7. Черных К.В., Кожухова А.В., Толмачёва Л.В. Использование программного пакета OpenOffice.org в профессиональной деятельности инженера. *Прикладные информационные системы в технологиях наземного транспорта (машиностроение)*. 2019;5-8.
8. Xu Y., Zhang M., Gao Z. The Construction of Distance Education Personalized Learning Platform Based on Educational Data Mining. *Advances in Intelligent Systems and Computing*. 2020;(1017):1076-1085.
9. Liu D. Y.-T., Atif A., Froissard J.-C., Richards D. An Enhanced Learning Analytics Plugin for Moodle: Student Engagement and Personalised Intervention. *ASCILITE 2015 — Australasian Society for Computers in Learning and Tertiary Education, Conference Proceedings.* 2019;180-189.
10. Lee D.H., Cho S.H., Kim Y. A A Design and Development of the Learning Contents Management Based on the Personalized Online Learning. *International Journal on Advanced Science, Engineering and Information Technology.* 2018;8(4):1321-1326.
11. Mironov V.V., Gusarenko A.S., Yusupova N.I. Situation-oriented Databases: Current State and Prospects for Research. *Vestnik UGATU*. 2015;19(2):188-199.
12. Mironov V.V., Gusarenko A.S., Yusupova N.I. The Invariance of The Virtual Data in the Situationally Oriented Database when Displayed on Heterogeneous Data Storages. *Herald of Computer and Information Technologies*. 2017;1(151):29-36.
13. Mironov V.V., Gusarenko A.S., Yusupova N.I. Structuring Virtual Multi-documents in Situationally-oriented Databases by Means of Entry-elements. *SPIIRAS Proceedings.* 2017;4(53):225-242.

14. Mironov V.V., Gusarenko A.S., Yusupova N.I. Displaying Virtual XML-documents on MySQL Tables in the Situation-oriented Databases, “Distributed Approach”. *Journal of Information Technologies and Computing Systems*. 2017;1:77-89.
15. Mironov V.V., Gusarenko A.S., Yusupova N.I. Integration of Virtual Multidocument Mappings into Real Data Sources in Situational-Oriented Databases. *Applied Informatics*. 2018;13(3):47-60.
16. Gusarenko A.S. Improvement of Situation-Oriented Database Model for Interaction with MySQL. *Journal of Instrument Egineering.* 2016:59(5):355-363.
17. Mironov V.V., Gusarenko A.S., Yusupova N.I. Situation-oriented Databases: Document Management on the Base of Embedded Dynamic Model. *in CEUR Workshop Proceedings (CEUR-WS.org): Selected Papers of the XI International Scientific-Practical Conference Modern Information Technologies and IT-Education (SITITO 2016)*. 2016;1761:238-247.
18. Mironov V.V., Gusarenko A.S., Yusupova N.I. Stream Handling Large Volume Documents in Situationally-oriented Databases. *International Scientific Journal INDUSTRY 4.0. Scientific Technical Union of Mechanical Engineering “INDUSTRY 4.0”.* 2018;3(5):240-244.
19. Миронов В.В., Гусаренко А.С., Юсупова Н.И. Ситуационно-ориентированные базы данных: polyglot persistence на основе REST-микросервисов. *Прикладная информатика*. 2019;14(5):86-97.
20. Миронов В.В., Гусаренко А.С., Юсупова Н.И. Применение веб-сервисов на основе ситуационно-ориентированной базы данных для мониторинга просмотра учебного видеоконтента. *Моделирование, оптимизация и информационные технологии*. 2019;7(3):20. Доступно по: https://moit.vivt.ru/wp-content/uploads/2019/09/MironovSoavtori_3_19_1.pdf DOI: 10.26102/2310-6018/2019.26.3.031. (дата обращения: 20.01.2020).

REFERENCES

1. Baswara S.Y., Widhiastuti R., Dewi L.C. Learning Model Based on Information Technology in an Accounting Education Courses Based on Technology at Faculty of Economics in Universitas Negeri Semarang. *KnE Social Sciences*. 2020;4(6):1280-1285. Available at: https://doi.org/10.18502/kss.v4i6.6678. (accessed 20.01.2020).
2. Yu Z., Xiong Z. Comparative Analyses for the Performance of Rational Rose and Visio in Software Engineering Teaching. *Journal of Physics: Conference Series. IOP Publishing.* 2018;1087(6):062041. DOI:10.1088/1742-6596/1087/6/062041
3. Medoh C., Telukdarie A. Business Process Modelling Tool Selection: A review. *2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM). IEEE*. 2017:524-528.
4. Sarvepalli A., Godin J. Business Process Management in the Classroom. *Journal of Cases on Information Technology (JCIT).* 2017;19(2):17-28.
5. He L., Lian J. Instructional Design of Practice Course of Logistics System Planning and Design Based on Visio. *2018 9th International Conference on Information Technology in Medicine and Education (ITME). IEEE*, 2018:526-530.
6. David J Parker. Mastering Data Visualization with Microsoft Visio Professional 2016. *Packt Publishing Ltd.* 2016.
7. Chernykh K.V., Kozhukhova A.V., Tomacheva L.V. Using the Software Package Openoffice.org in the Professional Work of an Engineer. *Prikladnyye Informatsionnyye Sistemy v Tekhnologii Telefonnogo Transporta (Mashinostroyeniye)*. 2019:5-8.

8. Xu Y., Zhang M., Gao Z. The Construction of Distance Education Personalized Learning Platform Based on Educational Data Mining. *Advances in Intelligent Systems and Computing*. 2020;(1017):1076-1085.
9. Liu D. Y.-T., Atif A., Froissard J.-C., Richards D. An Enhanced Learning Analytics Plugin for Moodle: Student Engagement and Personalised Intervention. *ASCILITE 2015 — Australasian Society for Computers in Learning and Tertiary Education, Conference Proceedings.* 2019;180-189.
10. Lee D.H., Cho S.H., Kim Y. A Design and Development of the Learning Contents Management Based on the Personalized Online Learning. *International Journal on Advanced Science, Engineering and Information Technology*. 2018;8(4):1321-1326.
11. Mironov V.V., Gusarenko A.S., Yusupova N.I. situation-oriented Databases: Current State and Prospects for Research. *Vestnik UGATU*. 2015;19(2):188-199
12. Mironov V.V., Gusarenko A.S., Yusupova N.I. The Invariance of the Virtual Data in the Situationally Oriented Database when Displayed on Heterogeneous Data Storages. *Herald of Computer and Information Technologies*. 2017;1(151):29-36.
13. Mironov V.V., Gusarenko A.S., Yusupova N.I. Structuring Virtual Multi-documents in Situationally-oriented Databases by Means of Entry-elements. *SPIIRAS Proceedings*. 2017;4(53):225-242.
14. Mironov V.V., Gusarenko A.S., Yusupova N.I. Displaying Virtual XML-documents on MySQL Tables in the Situation-oriented Databases, “Distributed Approach”. *Journal of Information Technologies and Computing Systems.* 2017;1:77-89.
15. Mironov V.V., Gusarenko A.S., Yusupova N.I. Integration of Virtual Multidocument Mappings into Real Data Sources in Situational-oriented Databases. *Applied Informatics.* 2018;13(3):47-60.
16. Gusarenko A. S. Improvement of Situation-oriented Database Model for Interaction with MySQL. *Journal of Instrument Egineering*. 2016:59(5):355-363.
17. Mironov V.V., Gusarenko A.S., Yusupova N.I. Situation-oriented Databases: Document Management on the Base of Embedded Dynamic Model. *CEUR Workshop Proceedings (CEUR-WS.org): Selected Papers of the XI International Scientific-Practical Conference Modern Information Technologies and IT-Education (SITITO 2016).* 2016;1761:238-247.
18. Mironov V.V., Gusarenko A.S., Yusupova N.I. Stream Handling Large Volume Documents in Situationally-oriented Databases, International Scientific Journal INDUSTRY 4.0. *Scientific Technical Union of Mechanical Engineering “INDUSTRY 4.0”.* 2018;3(5):240-244.
19. Mironov V.V., Gusarenko A.S., Yusupova N.I. Situation-oriented Databases: Polyglot Persistence Based on Rest Microservices. *Applied Informatics*. 2019;14(5):86-97.
20. Mironov V.V., Gusarenko A.S., Yusupova N.I. Application of Web Services Based on Situation-oriented Database for Monitoring the Viewing of the Educational Video-content. *Modeling, Optimization and Information Technology*. 2019;7(3):20. Available at: https://moit.vivt.ru/wp-content/uploads/2019/09/MironovSoavtori_3_19_1.pdf (In Russ) DOI: 10.26102/2310-6018/2019.26.3.031. (accessed 20.01.2020).

ИНФОРМАЦИЯ ОБ АВТОРАХ / INFORMATION ABOUT THE AUTHORS

Миронов Валерий Викторович, д-р техн. наук, профессор кафедры автоматизированных систем управления, Уфимский государственный авиационный технический университет, Уфа, Российская Федерация.
e-mail: mironov@ugatu.su
ORCID: 0000-0002-0550-4676

Valeriy V. Mironov, Doctor Of Technical Sciences, Professor Of Department Of Automated Control Systems, Ufa State Aviation Technical University, Ufa, Russia

Гусаренко Артем Сергеевич, канд. техн. наук, доцент кафедры автоматизированных систем управления, Уфимский государственный авиационный технический университет, Уфа, Российская Федерация.
e-mail: gusarenko@ugatu.su
ORCID: 0000-0003-4132-6106

Artem S. Gusarenko, Candidate Of Technical Sciences, Associate Professor Of Department Of Automated Control Systems, Ufa State Aviation Technical University, Ufa, Russia

Тугузбаев Гаяз Ахтямович, аспирант кафедры автоматизированных систем управления, Уфимский государственный авиационный технический университет, Уфа, Российская Федерация.
e-mail: tuguzbaev.g@ugatu.su
ORCID: 0000-0003-2036-6416

Gayaz A. Tuguzbaev, Graduate Student Of The Department Of Automated Control Systems, Ufa State Aviation Technical University, Ufa, Russia