УДК 004.89

А.А. Филиппов

ФОРМИРОВАНИЕ БАЗЫ ЗНАНИЙ ПРОЕКТНОЙ ОРГАНИЗАЦИИ ДЛЯ АВТОМАТИЗАЦИИ ПРОЦЕССА АРХИТЕКТУРИЗАЦИИ ПРОГРАММНЫХ СИСТЕМ¹

Филиппов Алексей Александрович, кандидат технических наук, окончил факультет информационных систем и технологий Ульяновского государственного технического университета, доцент кафедры «Информационные системы» УлГТУ. Имеет статьи в области онтологического моделирования, интеллектуальных систем хранения и обработки информации. [e-mail: al.filippov@ulstu.ru].

Аннотация

В данной статье рассмотрен подход к автоматизации процесса архитектуризации программных систем (ПС) на основе опыта предыдущих проектов. Под архитектуризацией ПС понимается осмысление и представление ПС в форме описания архитектуры, состоящей из множества артефактов проектирования. Артефакты проектирования формируются в процессе всего жизненного цикла системы.

При разработке новой ПС возможно повысить качество и снизить время процессов проектирования и конструирования ПС за счет привлечения опыта предыдущих проектов через извлечения из них единиц проектного опыта (ЕПО). Под ЕПО понимаются удачные проектные и архитектурные решения, использованные в предыдущих проектах.

В статье описана логическая модель базы знаний (БЗ) проектной организации, позволяющая осуществлять работу по хранению и поиску ЕПО. Также модель БЗ позволяет связывать ЕПО с проектом, в рамках которого она была получена, со стадией модели жизненного цикла, на которой она была сформирована, понятиями предметной области, в рамках которой производилась работа, а также с артефактом проектирования, из которого была извлечена данная ЕПО.

Ключевые слова: архитектуризация, онтология, программная система, проектирование, конструирование, база знаний, проектный опыт.

doi: 10.35752/1991-2927-2020-1-5-86-96

THE CREATION OF THE KNOWLEDGE BASE OF DESIGN ORGANIZATION FOR AUTOMATING THE ARCHITECTING PROCESS OF SOFTWARE SYSTEMS

Aleksei Aleksandrovich Filippov, Candidate of Sciences in Engineering; graduated from the Faculty of Information Systems and Technologies of Ulyanovsk State Technical University; Associate Professor of the Department of Information Systems of UISTU; an author of articles in the field of intelligent data storage and processing systems. e-mail: al.filippov@ulstu.ru.

Abstract

This article deals with an approach to automating the architecting process of software systems based on the experience of previous projects. The architecture of the software system is the understanding and presentation of the software system in the form of the architecture description. The architecture description consists of many design artifacts. Design artifacts are formed during the whole life cycle of the system.

It is possible to improve the quality and reduce the time of design and implementation stages in the process of developing a new software system based on the experience of previous projects by extracting design experience units from previous projects. The design experience unit is a successful design and architectural solutions used in previous projects.

The article describes the logical model of the knowledge base of the design organization that allows storing and searching for design experience units. The knowledge base model also allows link the design experience unit with the project where

¹ Исследование выполнено при финансовой поддержке РФФИ и Ульяновской области в рамках научных проектов № 19-47-730005, № 19-47-730003 и № 18-47-730019.

it was obtained, with the stage of the life cycle model where it was formed, with the concepts of the subject area, and with the design artifact from which it was extracted.

Key words: architecting, ontology, software system, design, implementation, knowledge base, design experience.

ВВЕДЕНИЕ

Разработка любой программной системы (ПС) состоит из нескольких этапов, качество выполнения которых влияет на качество итоговой системы [1]. Основным этапом разработки новой ПС является стадия проектирования, в рамках которой формируется описание архитектуры и модель разрабатываемой системы. Качественный проект новой ПС важен, так как от него зависит качество функционирования всего комплекса возлагаемых на новую систему задач.

Далее выполняется этап кодирования – непосредственная разработка новой ПС. Успешное конструирование ПС возможно только при использовании удачных проектных решений, полученных на этапе проектирования. Только правильно составленный проект позволит добиться нужного результата и осуществить разработку действительно качественной и конкурентоспособной ПС.

Чем на более позднем этапе разработки ПС обнаружены ошибки проектирования, тем больше ресурсов требуется для их исправления [1]. Следовательно, удачные проектные решения, полученные при работе над предыдущими проектами, должны быть использованы в процессе работы над новыми проектами.

Современная крупная проектная организация обладает значительным по объему проектным заданием, состоящим из множества артефактов проектирования, полученных в процессе работы над предыдущими (завершенными) проектами. Фактически, артефакты проектирования содержат в себе множество единиц проектного опыта (ЕПО) большого числа высококвалифицированных специалистов, которые на протяжении многих лет занимались разработкой и проектированием ПС. В данном исследовании под ЕПО понимаются удачные проектные и архитектурные решения, полученные при работе над предыдущими проектами.

В работах [2, 3] отмечается, что именно на начальных этапах разработки новых проектов принципиально важным является использование опыта предыдущих разработок. Решение указанной проблемы может основываться на применении интеллектуальных методов и алгоритмов анализа артефактов проектирования с целью построения базы знаний (БЗ) и опыта проектной организации. При увеличении количества гетерогенных неструктурированных данных затрудняется поиск необходимых сведений. В результате, довольно часто важный опыт предыдущих разработок, зафиксированный в артефактах проектирования, остается не востребуемым, и, как следствие, увеличивается время выполне-

ния цикла разработки ПС, снижается качество конечного продукта.

Сложность разработки и поддержки ПС достигла уровня, при котором необходимо применять принципы и процедуры процесса архитектуризации, описанные в стандарте ISO/IEC/IEEE 42010:2011 [4].

Архитектуризация – осмысление архитектуры ПС для лучшего понимания сути ПС и основных свойств, имеющих отношение к ее поведению, составу и развитию и влияющих на выполнимость, полезность и сопровождаемость системы. Процесс архитектуризации определяет требования и особенности разработки, функционирования и сопровождения ПС на всех стадиях жизненного цикла системы. Описание архитектуры используется для координации заинтересованных сторон, вовлеченных в процессы разработки, внедрения, поддержки и использования ПС: заказчик, пользователи, разработчики. Артефакты проектирования, составляющие основу архитектуры ПС, позволяют формировать архитектуру ПС в контекстах заинтересованных сторон и предметных областей.

В рамках данного исследования описание архитектуры ПС рассматривается как совокупность артефактов проектирования, полученных при работе над предыдущими проектами. При этом ПС разрабатывается для функционирования в рамках некоторого контекста, представленного особенностями предметной области. Контекст определяет множество архитектурных требований и ограничений [4]:

- компоненты или элементы ПС;
- взаимодействие элементов ПС;
- организация проекта ПС;
- развитие ПС в рамках ее жизненного цикла.

Представление множества артефактов проектирования в виде фрагментов БЗ позволит сократить пространство поиска и тем самым ускорить процедуру нахождения требуемых сведений в виде ЕПО для автоматизации процесса архитектуризации ПС.

Основной задачей данного исследования является изучение возможностей и ограничений методов анализа данных и инженерии знаний для построения БЗ, способных автоматизировать процесс проектирования ПС.

1 БЗ ПРОЕКТНОЙ ОРГАНИЗАЦИИ

На процесс развития ПС влияют требования соответствия одной или более потребностям заинтересованных сторон целям, ожиданиям, обязанностям, ограничениям проекта, предположениям, зависимостям, качественным атрибутам, архитектурным решениям,

рискам или иным источникам, имеющим отношение к ПС [4]. Таким образом, развитие ПС необходимо рассматривать в рамках жизненного цикла, требований к ПС, множества принятых проектных решений.

В рамках данного исследования архитектура ПС рассматривается как целостная концепция основных свойств системы, воспринимаемая наилучшим образом через множественные представления этой архитектуры: деловое представление, физическое представление, техническое представление. Артефакты проектирования – самые примитивные конструкции, составляющие описание архитектуры ПС. В процессе архитектуризации ПС формируется описание архитектуры. Описание архитектуры также может быть получено в процессе переработки описания архитектуры предыдущих проектов.

Учет специфики проектных знаний и опыта, зафиксированных в описании архитектуры ПС, приводит к необходимости формирования БЗ проектной организации особой структуры, основанной на системе онтологий.

Ведущие исследователи в области онтологического анализа и инженерии знаний, такие как Хорошевский В.Ф., Загоруйко Ю.А. [5], Гаврилова Т.А. [6], Соловьев В.Д., Лукашевич Н.В., Добров Б.В., Ландэ Д.В., Смирнов С.В. [7], Gruber T. [8], Berners-Lee T., Uschold M. и другие отмечают актуальность исследований, основанных на онтологическом подходе.

В трудах [9–13] отмечается важность использования онтологического инжиниринга в процессах проектирования и разработки ПС, рассматриваются различные виды онтологий:

1. Онтология верхнего уровня – определяет набор основных понятий для любых проблемных областей.
2. Онтология процесса разработки – описывает основные этапы жизненного цикла разработки ПС.
3. Прикладная онтология – определяет набор понятий для конкретной предметной области.
4. Онтология модели требований – описывает основные характеристики ПС.
5. Онтология поведения системы – описывает набор поведений системы – набор действий, свойственных определенным сценариям работы с ПС.
6. Онтология архитектуры системы – описывает архитектуру системы, содержит перечень используемых архитектурных стилей, компонентов, их свойств и отношений.
7. Онтология логики системы – содержит набор причин, приводящих к определенному поведению ПС.
8. Онтология объектно-ориентированного дизайна – содержит набор основных классов, их полей, методов и отношений между классами.
9. Онтология шаблонов проектирования – содержит перечень часто используемых шаблонов проектирования ПС.

В работах [9, 12] говорится, что онтологии могут быть использованы вместо традиционных языков моделирования ПС, например UML, так как онтологии позволяют контролировать логическую целостность и непротиворечивость полученной модели.

Однако существующие методы формирования БЗ в виде набора онтологий для поддержки и автоматизации процесса архитектуризации ПС предполагают привлечение экспертов проблемной области и специалистов в области инженерии знаний, что требует существенных временных затрат на формирование такой БЗ.

Следовательно, актуальной является разработка моделей, методов и алгоритмов построения базы знаний проектной организации в процессе автоматизированного анализа артефактов проектирования, полученных в процессе работы над предыдущими проектами.

Основная сложность данного подхода заключается в необходимости унификации артефактов проектирования для формирования БЗ, являющейся основой информационного обеспечения комплекса средств автоматизации архитектурного проектирования ПС. Также необходимо включить в БЗ сведения о различных предметных областях, в рамках которых осуществлялась работа над предыдущими проектами.

В рамках данного исследования БЗ проектной организации состоит из пяти компонентов, каждый из которых позволяет описывать и хранить следующие необходимые сведения:

1. О различных предметных областях,
2. О моделях жизненного цикла разработки ПС,
3. О проектах ПС и их взаимосвязях,
4. Об особенностях и структуре артефактов проектирования,
5. Об ЕПО.

Для организации БЗ проектной организации используется совокупность взаимосвязанных онтологий. В качестве логической основы БЗ проектной организации используется расширение дескрипционной логики *ALC* – дескрипционная логика *SHOIF(D)* [14, 15].

В рамках дескрипционной логики *SHOIF(D)* БЗ имеет следующий вид [16]:

$$KV = \{TBox, ABox\},$$

где *TBox* – набор терминологических аксиом, представляющих общие знания о понятиях БЗ проектной организации и их взаимосвязях;

ABox – набор утверждений (фактов) об индивидах.

Логическое представление онтологии БЗ имеет вид:

$$\begin{aligned} TBox = TBox^{dom} \cup TBox^{dev} \cup TBox^{proj} \cup \\ \cup TBox^{art} \cup TBox^{exp}, \\ ABox = ABox^{dom} \cup ABox^{dev} \cup ABox^{proj} \cup \\ \cup ABox^{art} \cup ABox^{exp}, \end{aligned}$$

где *TBox^{dom}* и *ABox^{dom}* – терминология и факты онтологии предметных областей, в рамках которых производилась работа над предыдущими проектами;

TBox^{dev} и *ABox^{dev}* – терминология и факты онтологии моделей жизненного цикла разработки ПС, применяемых в проектной организации;

TBox^{proj} и *ABox^{proj}* – терминология и факты онтологии завершенных проектов ПС проектной организации;

$TBox^{art}$ и $ABox^{art}$ – терминология и факты онтологии артефактов проектирования;

$TBox^{exp}$ и $ABox^{exp}$ – терминология и факты онтологии ЕПО.

Терминология и набор фактов записывается с применением следующих операторов и аксиом дескрипционной логики:

- $\top$ – домен;
- $A \sqsubseteq B$ – аксиома вложенности концептов;
- $A \equiv B$ – аксиома эквивалентности концептов (или определение концептов с необходимыми и достаточными условиями);
- $A \sqcap B$ – пересечение (или конъюнкция) концептов;
- $A \sqcup B$ – объединение (или дизъюнкция) концептов;
- $\forall R.A$ – ограничение на значения роли (или ограничение квантором всеобщности);
- $\exists R.A$ – экзистенциальное ограничение (или ограничение квантором существования);
- $a: A$ – утверждение о принадлежности индивида a концепту A ;
- $(a, b): R$ – утверждение о связи двух индивидов a и b ролью R .

Рассмотрим онтологии модели БЗ проектной организации более подробно.

2 ОНТОЛОГИЯ ПРЕДМЕТНОЙ ОБЛАСТИ

Онтология предметной области позволяет адаптировать БЗ проектной организации к особенностям предметных областей, в рамках которых проводилась работа над предыдущими проектами. Фактически, онтология предметной области выступает некоторым базисом, позволяющим сформировать единый понятийный аппарат, объединяющий и унифицирующий сущности различных предметных областей, имеющих единую семантику, но обозначенных различными терминами.

Терминологию $TBox^{dom}$ онтологии предметной области имеет вид:

$$\begin{aligned} C_1 &\sqsubseteq C & C_2 &\sqsubseteq C & \dots & C_i &\sqsubseteq C \\ C_{11} &\sqsubseteq C_1 & C_{12} &\sqsubseteq C_1 & C_{21} &\sqsubseteq C_2 & C_{22} &\sqsubseteq C_2 \\ C &\equiv \top \sqcap \exists hasName.String \sqcap \forall hasName.String \\ \{c\} &\equiv \top \sqcap \exists represents.C \sqcap \forall represents.C \\ T &\equiv \top \sqcap \exists hasName.String \sqcap \forall hasName.String \sqcap \\ &\sqcap \exists associateWith.\{c\} \sqcap \forall associateWith.\{c\} \end{aligned}$$

где C – понятие предметной области;

$\{c\}$ – номинал понятия предметной области;

T – термин, описывающий понятие предметной области;

$hasName$ – наименование функциональной роли имеет наименование;

$String$ – конкретный домен (concrete domain) строкового типа;

$associateWith$ – наименование ассоциативной роли, формирующей связи между понятиями и терминами, описывающими данные понятия («понятие-термин»);

$\sqsubseteq$ – наименование обобщающей роли, определяющей иерархию понятий предметной области вида «понятие-понятие»;

$represents$ – наименование функциональной роли, связывающей понятие с его номиналом.

3 ОНТОЛОГИЯ МОДЕЛЕЙ ЖИЗНЕННОГО ЦИКЛА

Онтология моделей жизненного цикла используется в рамках БЗ проектной организации для определения динамики развития ЕПО. Достаточно часто в рамках процессов внедрения и поддержки в ПС вносятся изменения, которые были не учтены в исходном описании архитектуры. Привязка изменений к отдельным стадиям модели жизненного цикла позволяет в будущем учитывать такие особенности в рамках процесса проектирования.

Терминологию $TBox^{dev}$ онтологии моделей жизненного цикла можно представить в виде следующего выражения:

$$S \equiv \top \sqcap \exists hasName.String \sqcap \forall hasName.String \sqcap \forall hasNextStage.S,$$

где S – стадия модели жизненного цикла;

$hasNextStage$ – наименование иррефлексивной роли, определяющей порядок следования стадий модели жизненного цикла.

4 ОНТОЛОГИЯ ПРОЕКТОВ

Онтология проектов используется в БЗ проектной организации для сопоставления ЕПО и проектов, в рамках которых они были сформированы. Сохранение связей между ЕПО и проектами позволяет определить родственные проекты, имеющие схожие описания архитектурных решений, а также отследить динамику их развития для выделения удачных проектных решений.

Терминологию $TBox^{proj}$ онтологии проектов представим в виде следующего выражения:

$$P \equiv \top \sqcap \exists hasName.String \sqcap \forall hasName.String \sqcap \forall hasFork.P,$$

где P – проект ПС;

$hasFork$ – наименование иррефлексивной ассоциативной роли, определяющей родственные связи между проектами вида «родительский проект-дочерний проект». Данная роль иллюстрирует ситуацию создания ответвленного проекта (fork) – использование кодовой базы проекта ПС в качестве старта для другого проекта, при этом родительский проект может как продолжать существование, так и прекратить его.

5 ОНТОЛОГИЯ АРТЕФАКТОВ ПРОЕКТИРОВАНИЯ

Онтология артефактов проектирования используется в рамках БЗ проектной организации для опреде-

ления множества поддерживаемых типов артефактов проектирования. Данные сведения необходимы как в процессе анализа артефактов проектирования при формировании содержимого БЗ, так и в процессе генерации прототипов артефактов проектирования на основе содержимого БЗ.

Терминологию $TBox^{art}$ онтологии артефактов проектирования запишем как:

$$T \equiv \top \sqcap \exists hasName.String \sqcap \forall hasName.String$$

$$I \equiv \top \sqcup \sqcap \exists hasName.String \sqcap$$

$$\sqcap \forall hasName.String \sqcap \forall hasType.T,$$

где T – тип артефакта проектирования;

I – артефакт проектирования;

$hasType$ – ассоциативная роль, определяющая связь между типом артефакта проектирования и артефактом проектирования вида «тип артефакта-артефакт».

6 Онтология ЕПО

Онтология ЕПО содержит полученные в процессе анализа артефактов проектирования ЕПО. ЕПО – удачные проектные и архитектурные решения, содержащиеся в предыдущих проектах, которые могут быть использованы в процессе разработки новой ПС. Онтология ЕПО состоит из двух компонентов:

1. Компонент для формализации описания модели разрабатываемой ПС.

2. Компонент для формализации описания технологической составляющей разрабатываемой ПС.

Данное разделение обусловлено существенными концептуальными различиями между моделью системы и ее технологической составляющей.

Таким образом, терминология $TBox^{exp}$ и набор фактов $ABox^{exp}$ имеют вид:

$$TBox^{exp} = TBox^{mexp} \cup TBox^{texp},$$

$$ABox^{exp} = ABox^{mexp} \cup ABox^{texp},$$

где $TBox^{mexp}$ и $ABox^{mexp}$ – терминология и факты описания модели ПС;

$TBox^{texp}$ и $ABox^{texp}$ – терминология и факты описания технологической составляющей ПС.

Терминология $TBox^{mexp}$ описания модели ПС будет иметь вид:

$$C \equiv \top \sqcap \forall hasValue.$$

$$(Integer \sqcup String \sqcup Double \sqcup E)$$

$$A \equiv \top \sqcap \exists hasName.String \sqcap \forall hasName.String \sqcap$$

$$\sqcap \forall hasConstraint.C$$

$$P \equiv \top \sqcap \exists hasName.String \sqcap \forall hasName.String$$

$$E \equiv \top \sqcap \exists hasName.String \sqcap \forall hasName.String \sqcap$$

$$\sqcap \forall hasAttribute.A \sqcap \forall hasBusinessProcess.P \sqcap$$

$$\sqcap \forall hasParent.E,$$

где C – ограничение атрибута сущности;

A – атрибут сущности;

P – бизнес-процесс, в который вовлечена сущность;

E – сущность описания модели ПС;

$hasValue$ – наименование ассоциативной роли имеет значение;

$Integer$ – конкретный домен целочисленного типа;

$Double$ – конкретный домен типа вещественных чисел;

$hasAttribute$ – наименование ассоциативной роли, определяющей связь между сущностью и ее атрибутом вида «сущность-атрибут»;

$hasConstraint$ – наименование ассоциативной роли, определяющей связь между атрибутом сущности и его ограничением вида «атрибут-ограничение»;

$hasBusinessProcess$ – наименование ассоциативной роли, определяющей связь между сущностью и бизнес-процессом вида «сущность-бизнес-процесс»;

$hasParent$ – наименование ассоциативной роли, определяющей связь между сущностью и ее родителем вида «сущность потомок-сущность родитель».

Терминологию $TBox^{texp}$ технологической составляющей ПС представим в следующем виде:

$$U \equiv \top \sqcap \exists hasName.String \sqcap \forall hasName.String \sqcap$$

$$\sqcap \forall operateWith.U \sqcap \forall hasParent.U,$$

где U – функциональный блок описания технологической составляющей ПС;

$operateWith$ – наименование иррефлексивной ассоциативной роли, определяющей взаимодействие между функциональными блоками;

$hasParent$ – наименование ассоциативной роли, определяющей вложенность функциональных блоков описания технологической составляющей ПС.

7 Модель БЗ проектной организации

Рассмотренные ранее онтологии формируют компоненты БЗ проектной организации, позволяющей концептуально представить проекты ПС, работа над которыми проводилась в проектной организации. На рисунке представлена схема, иллюстрирующая взаимодействие онтологий БЗ проектной организации.

Терминология $TBox$ БЗ проектной организации будет иметь вид:

$$E^{mexp}, U^{texp} \equiv \top \sqcap \forall associateWith.C^{dom} \sqcap$$

$$\sqcap \forall createOn.S^{dev} \sqcap \forall includeIn.P^{proj} \sqcap$$

$$\sqcap \forall extractFrom.I^{art},$$

где $associateWith$ – наименование ассоциативной роли для связи понятий онтологии предметной области с ЕПО;

$createOn$ – наименование ассоциативной роли для связи стадий модели жизненного цикла с ЕПО;

includeIn – наименование ассоциативной роли для связи проектов с ЕПО;

extractFrom – наименование ассоциативной роли для связи артефактов проектирования с ЕПО.

8 ИЛЛЮСТРАТИВНЫЙ ПРИМЕР ФОРМИРОВАНИЯ БЗ ПРОЕКТНОЙ ОРГАНИЗАЦИИ

Для лучшего понимания структуры БЗ проектной организации рассмотрим иллюстративный пример формирования содержимого различных онтологий БЗ.

Для формирования содержимого онтологии моделей жизненного цикла используется модель жизненного цикла, состоящая из следующих этапов: определение требований, спецификация, проектирование, реализация, тестирование, сопровождение, развитие:

Рис. Система онтологий БЗ проектной организации

ОпределениеТребований : *S*

Проектирование : *S*

Тестирование : *S*

Развитие : *S*

(ОпределениеТребований, Спецификации) : *hasNextStage*

(Спецификации, Проектирование) : *hasNextStage*

(Проектирование, Реализация) : *hasNextStage*

(Реализация, Тестирование) : *hasNextStage*

(Тестирование, Сопровождение) : *hasNextStage*

(Сопровождение, Развитие) : *hasNextStage*

Спецификации : *S*

Реализация : *S*

Сопровождение : *S*

Онтология проектов будет иметь следующий вид:

TestProject : *P*.

Для успешного анализа артефактов проектирования необходима разработка отдельного программного модуля, ориентированного на работу с конкретными типами артефактов проектирования.

В данном примере в качестве доступных артефактов проектирования рассматриваются модели данных реляционных баз данных (РБД). Таким образом, содержимое онтологии артефактов проектирования будет иметь вид:

RDBModel : *T*

PostgresqlRDBModel : *I*.

(*PostgresqlRDBModel*, *RDBModel*) : *hasType*

Рассмотрим пример формирования содержимого онтологии ЕПО в процессе анализа модели данных РБД. Допустим, анализируемая РБД содержит следующие таблицы:

1. Tools – таблица для хранения перечня инструментов (табл. 1).

Таблица 1

Структура таблицы Tools

Атрибут	Тип данных	Ограничение	Описание
id	INTEGER	Primary key	Первичный ключ
name	VARCHAR(255)	-	Название
production_date	DATE	nullable	Дата производства
tool_types_id	INTEGER	Foreign key	Тип инструмента

2. Tool types – таблица для хранения перечня типов инструментов (табл. 2).

Таблица 2

Структура таблицы Tool types

Атрибут	Тип данных	Ограничение	Описание
id	INTEGER	Primary key	Первичный ключ
name	VARCHAR(255)	-	Название

Таблица 3 содержит сопоставление структурных компонентов модели данных РБД с компонентами онтологии O^{exp} .

Таблица 3

Соответствие компонентов модели данных РБД и компонентов онтологии O^{exp}

Компонент модели данных РБД	Компонент онтологии O^{exp}
Таблица	Индивидуал класса E
Представление	Индивидуал класса E
Иерархия таблиц (при наследовании)	Множество ролей $hasParent$
Колонка таблицы	Индивидуал класса A
Тип данных колонки	Множество индивидуалов класса C и множество ролей $hasConstraint$
Ограничения колонки	Множество индивидуалов класса C и множество ролей $hasConstraint$ и $hasValue$

После извлечения ЕПО из модели данных РБД, представленной таблицами Tools (табл. 1) и Tool types (табл. 2), получим следующие сущности онтологии O^{exp} :

$RelationalTable \sqsubseteq E$

$RelationalDataType \sqsubseteq C$

$Tools : RelationalTable$

$RelationalAttribute \sqsubseteq A$

$RelationalConstraint \sqsubseteq C$

$ToolTypes : RelationalTable$

$Tools.id : RelationalAttribute$

$Tools.name : RelationalAttribute$

$Tools.production_date : RelationalAttribute$

$Tools.tool_types_id : RelationalAttribute$

$ToolTypes.id : RelationalAttribute$

$ToolTypes.name : RelationalAttribute$

$INTEGER : RelationalDataType$

$DATE : RelationalDataType$

$LENGTH_255 : RelationalConstraint$

$PrimaryKey : RelationalConstraint$

$VARCHAR : RelationalDataType$

$NULLABLE : RelationalConstraint$

$ForeignKey : RelationalConstraint$

$ToolTypes_FK : RelationalConstraint$

$(Tools, Tools.id) : hasAttribute$

$(Tools, Tools.name) : hasAttribute$

$(Tools, Tools.production_date) : hasAttribute$

$(Tools, Tools.tool_types_id) : hasAttribute$

$(ToolTypes, ToolTypes.id) : hasAttribute$

$(ToolTypes, ToolTypes.name) : hasAttribute$

$(LENGTH_255, 255 : Integer) : hasValue$

$(ToolTypes_FK, ToolTypes) : hasValue$

(Tools.id,INTEGER) : hasConstraint
(Tools.id,PrimaryKey) : hasConstraint
(Tools.name,VARCHAR) : hasConstraint
(Tools.name,LENGTH_255) : hasConstraint
(Tools.production_date,DATE) : hasConstraint
(Tools.production_date,NULLABLE) : hasConstraint
(Tools.tool_types_id,INTEGER) : hasConstraint
(Tools.tool_types_id,ForeignKey) : hasConstraint
(Tools.tool_types_id,ToolTypes_FK) : hasConstraint

(ToolTypes.id,INTEGER) : hasConstraint
(ToolTypes.id,PrimaryKey) : hasConstraint
(ToolTypes.name,VARCHAR) : hasConstraint
(ToolTypes.name,LENGTH_255) : hasConstraint

Как видно из данного примера, фрагмент онтологии *О_{техп}* содержит следующие сущности:

1. Множество индивидуалов класса *E* – сущности, например, *Tools* и *ToolTypes*.

2. Множество индивидуалов класса *A* – атрибуты, например, *Tools.id*, *Tools.name*, *Tools.inventory_number* и т. д.

3. Множество индивидуалов класса *C* – ограничения, представляющие типы данных атрибутов, требования к их представлению, а также связи с другими сущностями РБД через внешние ключи, например:

- *INTEGER : RelationalDataType* – тип данных Integer;
- *LENGTH_255 : RelationalConstraint* – ограничение на максимальный размер строки;
- *ForeignKey : RelationalConstraint* – внешний ключ;

• *ToolTypes_FK : RelationalConstraint* – конкретная реализация внешнего ключа.

4. Множество ролей, определяющих связи между сущностями и их атрибутами, атрибутами и их ограничениями – *hasAttribute* и *hasConstraint* соответственно, например:

• *(Tools,Tools.id) : hasAttribute* – сущность *Tools* имеет атрибут *Tools.id*;

• *(Tools.name, VARCHAR) : hasConstraint* – атрибут *Tools.name* имеет тип данных *VARCHAR*.

5. Множество ролей, определяющих значения параметров ограничений, например:

• *(LENGTH_255, 255 : Integer) : hasValue* – ограничение на максимальный размер строки должен составлять 255 символов;

• *(ToolTypes_FK, ToolTypes) : hasValue* – значением первичного ключа должен быть индивидуал класса *ToolType* (запись из таблицы *Tool types*).

Онтология предметной области будет выглядеть следующим образом:

Инструменты $\sqsubseteq$ C

ТипыИнструментов $\sqsubseteq$ Инструменты

инструменты : T	инструмент : T
типы : T	тип : T
(инструменты, Инструменты) : <i>associateWith</i>	
(инструмент, Инструменты) : <i>associateWith</i>	
(инструменты, ТипыИнструментов) : <i>associateWith</i>	
(инструмент, ТипыИнструментов) : <i>associateWith</i>	
(типы, ТипыИнструментов) : <i>associateWith</i>	
(тип, ТипыИнструментов) : <i>associateWith</i>	

После заполнения всех онтологий БЗ проектной организации необходимо установить связи между их сущностями:

(Tools, Инструменты) : associateWith
(Tools, Реализация) : createOn
(Tools, TestProject) : includeId

(Tools, PostgresqlRDBModel) : extractFrom
(ToolTypes, ТипыИнструментов) : associateWith
(ToolTypes, Реализация) : createOn
(ToolTypes, TestProject) : includeId
(ToolTypes, PostgresqlRDBModel) : extractFrom

9 ИЛЛЮСТРАТИВНЫЙ ПРИМЕР ИСПОЛЬЗОВАНИЯ БЗ ПРОЕКТНОЙ ОРГАНИЗАЦИИ

Основным источником данных для автоматизации процесса архитектурызации ПС с применением БЗ проектной организации выступает техническое задание, либо спецификация на разработку новой ПС. Анализируя такие документы, необходимо определить основные сущности предметной области и бизнес-процессы,

формирующие модель ПС на концептуальном уровне. Обнаруженные в процессе анализа сущности и бизнес-процессы используются для организации поиска ЕПО в БЗ проектной организации.

Рассмотрим иллюстративный пример использования БЗ проектной организации для сущности «инструмент». Название сущности передается на вход БЗ и к содержимому БЗ проектной организации формируется запрос на языке SQWRL [17] следующего вида:

$$\begin{aligned}
 & \text{associateWith} (? \text{expUnit}, \text{инструмент}) \wedge \text{includeIn} (? \text{expUnit}, ? \text{project}) \wedge \\
 & \wedge \text{createOn} (? \text{expUnit}, ? \text{stage}) \wedge \text{extractFrom} (? \text{expUnit}, ? \text{artefact}) - > \\
 & - > \text{sqwrl} : \text{select} (? \text{project}, ? \text{stage}, ? \text{artefact}, ? \text{expUnit})
 \end{aligned}$$

В результате выполнения запроса пользователь получает следующие сведения:

- Проект: *TestProject*; Стадия проектирования: Реализация; Артефакт проектирования: *PostgresqlRDBModel*; ЕПО: *Tools*.
- Проект: *TestProject*; Стадия проектирования: Реализация; Артефакт проектирования: *PostgresqlRDBModel*; ЕПО: *ToolTypes*.

Таким образом, в результате запроса из БЗ проектной организации извлекаются ЕПО с указанием связей с проектом, в рамках которого она была получена, со стадией модели жизненного цикла, на которой она была сформирована, понятиями предметной области, в рамках которой производилась работа, а также с артефактом проектирования, из которого была извлечена данная ЕПО.

ЗАКЛЮЧЕНИЕ

В данной статье представлен подход к автоматизации процессов проектирования и архитектурызации ПС с использованием БЗ проектной организации. БЗ проектной организации представляет собой систему онтологий. Каждая онтология позволяет учитывать различные контексты и аспекты процесса архитектурызации ПС: особенности проблемной области, используемая модель жизненного цикла, удачные проектные и архитектурные решения.

Также в статье рассмотрен иллюстративный пример формирования каждой онтологии, составляющей БЗ проектной организации. Представлен иллюстративный при-

мер использования полученной БЗ проектной организации для поиска проектных и архитектурных решений.

В качестве дальнейшего развития предложенного подхода можно выделить добавление поддержки различных артефактов проектирования, на основе анализа которых будет формироваться содержимое БЗ проектной организации.

СПИСОК ЛИТЕРАТУРЫ

1. Макконнелл С. Совершенный код: Практическое руководство по разработке программного обеспечения. СПб. : Русская редакция/БХВ, 2017.
2. Маклаев В.А., Соснин П.И. Инструментально-технологическая среда формирования и использования опыта проектной организации // Автоматизация процессов управления. 2009. № 2 (16). С. 8–13.
3. Соснин П.И., Шумилов С.С., Ивасев А.Е. Архитектурный подход к прецедентно-ориентированному решению задач в разработке автоматизированных систем // Автоматизация процессов управления. 2019. № 1 (55). С. 49–56.
4. ISO/IEC/IEEE 42010:2011 Systems and Software Engineering – Architecture Description. 2011. URL: <http://docs.cntd.ru/document/1200139542> (дата обращения: 20.01.2020).
5. Загорюлько Ю.А., Боровикова О.И., Загорюлько Г.Б. Использование паттернов онтологического проектирования для разработки онтологий предметных областей // Сб. тр. конф. «Знания-Онтологии-Теории» (ЗОНТ-2017). Новосибирск, 2017. С. 139–148.

6. Интеграция wiki-технологии и онтологического моделирования в задаче управления знаниями предприятия / Д.И. Муромцев [и др.] // Сб. тр. конф. «КИИ-2008». Дубна, 2008. С. 360–368.

7. Самойлов Д.Е., Семенова В.А., Смирнов С.В. Паттерны структурной организации системы измеряемых свойств в онтологическом анализе данных // Сб. тр. конф. «Проблемы управления и моделирования в сложных системах». Самара, 2018. С. 358–366.

8. Shaaban A.M., Gruber T., Schmittner C. Ontology-based security tool for critical cyber-physical systems // Proceedings of the 23rd International Systems and Software Product Line Conference. 2019. Vol. B. pp. 207–210.

9. Bhatia M.P.S., Kumar A., Beniwal R. Ontologies for software engineering: past, present and future // Indian Journal of Science and Technology. 2016. Vol. 9. pp. 1–16.

10. An ontology pattern language for service modeling / R.A. Falbo et al. // Proceedings of the 31st Annual ACM Symposium on Applied Computing. 2016. pp. 321–326.

11. Ilyas Q.M. Ontology Augmented Software Engineering // Software Development Techniques for Constructive Information Systems Design. IGI Global, 2013. pp. 406–413.

12. Ontology driven software engineering: a review of challenges and opportunities / S. Isotani et al. // IEEE Latin America Transactions. 2015. Vol. 13. pp. 863–869.

13. Ontology-driven software development / J.Z. Pan et al. Springer Science & Business Media, 2012.

14. The description logic handbook: Theory, implementation and applications / F. Baader et al. Cambridge university press, 2003.

15. Bonatti P.A., Tettamanzi A.G.B. Some complexity results on fuzzy description logics // International Workshop on Fuzzy Logic and Applications. Springer, Berlin, Heidelberg, 2003. pp. 19–24.

16. Description logic programs: combining logic programs with description logic / B.N. Grosof et al. // Proceedings of the 12th international conference on World Wide Web. 2003. pp. 48–57.

17. O'Connor M.J., Das A.K. SQWRL: a query language for OWL // Proceedings of the 6th International Workshop on OWL: Experiences and Directions (OWLED). 2009. Vol. 529. URL: http://ceur-ws.org/Vol-529/owled2009_submission_42.pdf (дата обращения: 20.01.2020).

REFERENCES

1. MacConnell S. *Sovershennyi kod: Prakticheskoe rukovodstvo po razrabotke programmnogo obespecheniia* [Code Complete: Practical Handbook of Software Construction]. St.Petersburg, BHV Publ., 2017.

2. Maklaev V.A., Sosnin P.I. Instrumentalno-tekhnologicheskaiia sreda formirovaniia i ispolzovaniia opyta proektnoi organizatsii [Instrumental and Technological Environment for Generation and Use of Design

Organization Experience]. *Avtomatizatsiia protsessov upravleniia* [Automation of Control Processes], 2009, no. 2 (16), pp. 8–13.

3. Sosnin P.I., Shumilov S.S., Ivasev A.E. Arkhitekturnyi podkhod k pretsedentno-orientirovannomu resheniiu zadach v razrabotke avtomatizirovannykh system [An Architectural Approach to the Precedent-oriented Problem Solving in the Designing of Automated Systems]. *Avtomatizatsiia protsessov upravleniia* [Automation of Control Processes], 2019, no. 1 (55), pp. 49–56.

4. ISO/IEC/IEEE 42010:2011 *Systems and Software Engineering – Architecture Description*. 2011. Available at: <http://docs.cntd.ru/document/1200139542> (accessed 20.01.2020).

5. Zagorulko Iu.A., Borovikova O.I., Zagorulko G.B. Ispolzovanie patternov ontologicheskogo proektirovaniia dlia razrabotki ontologii predmetnykh oblastei [Application of Ontology Design Patterns in the Development of the Ontologies of Subject Domains]. *Sb. tr. konf. "Znaniia-Ontologii-Teorii" (ZONT-2017)* [Proc. of Conf. "Knowledge-Ontologies-Theories" (ZONT-2017)]. Novosibirsk, 2017, pp. 139–148.

6. Muromtsev D.I. et al. Integratsiia wiki-tekhnologii i ontologicheskogo modelirovaniia v zadache upravleniia znaniiami predpriiatiia [Integration of Wiki Technology and Ontology Modelling in the Context of Knowledge Management in an Enterprise]. *Sb. tr. konf. "KII-2008"* [Proc. of Sci. Conf. on Artificial Intelligence (KII-2018)]. Dubna, 2008, pp. 360–368.

7. Samoilov D.E., Semenova V.A., Sмирнов S.V. Patterny strukturnoi organizatsii sistemy izmeriaemykh svoistv v ontologicheskoi analize dannykh [Patterns of the Structural Organization of the System of Measurable Properties in the Ontological Data Analysis]. *Sb. tr. konf. "Problemy upravleniia i modelirovaniia v slozhnykh sistemakh"* [Proc. of Int. Sci. Conf. on Complex Systems: Control and Modelling Problems]. Samara, 2018, pp. 358–366.

8. Shaaban A.M., Gruber T., Schmittner C. Ontology-Based Security Tool for Critical Cyber-Physical Systems. *Proceedings of the 23rd International Systems and Software Product Line Conference*. 2019, vol. B, pp. 207–210.

9. Bhatia M.P.S., Kumar A., Beniwal R. Ontologies for Software Engineering: Past, Present and Future. *Indian Journal of Science and Technology*, 2016, vol. 9, pp. 1–16.

10. Falbo R.A. et al. An Ontology Pattern Language for Service Modeling. *Proceedings of the 31st Annual ACM Symposium on Applied Computing*. 2016, pp. 321–326.

11. Ilyas Q.M. Ontology Augmented Software Engineering. *Software Development Techniques for Constructive Information Systems Design*. IGI Global, 2013, pp. 406–413.

12. Isotani S. et al. Ontology Driven Software Engineering: A Review of Challenges and Opportunities. *IEEE Latin America Transactions*, 2015, vol. 13, pp. 863–869.

13. Pan J.Z. et al. *Ontology-Driven Software Development*. Springer Science & Business Media, 2012.
14. Baader F. et al. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.
15. Bonatti P.A., Tettamanzi A.G.B. Some Complexity Results on Fuzzy Description Logics. *International Workshop on Fuzzy Logic and Applications*. Springer, Berlin, Heidelberg, 2003, pp. 19–24.
16. Grosz B.N. et al. Description Logic Programs: Combining Logic Programs with Description Logic. *Proceedings of the 12th International Conference on World Wide Web*. 2003, pp. 48–57.
17. O'Connor M.J., Das A.K. SQWRL: A Query Language for OWL. *Proceedings of the 6th International Workshop on OWL: Experiences and Directions (OWLED)*. 2009, vol. 529. Available at: http://ceur-ws.org/Vol-529/owled2009_submission_42.pdf (accessed 20.01.2020).