• Целесообразно не разделять, а совмещать в рамках отдельных заданий изучение 2D- и 3D-технологий создания моделей и КД изделий.

• На курсах повышения квалификации специалистов и в рамках учебных практик студентов можно рекомендовать после выполнения отдельных упражнений практикума обращаться к вызывающим интерес и профессиональную заинтересованность урокам Азбук системы Компас.

Примеры выполнения тестовых заданий следует дополнять аналогичными вариантами заданий для самостоятельного решения.

СПИСОК ЛИТЕРАТУРЫ

1. Большаков В. П., Бочков А. Л., Лячек Ю. Т. Твердотельное моделирование деталей в CAD-системах: AutoCAD, КОМПАС-3D, SolidWorks, Inventor. СПб.: Питер, 2015. 480 с.

2. Кидрук М. И. Видеосамоучитель. КОМПАС-3D (+DVD). СПб.: Питер, 2009. 288 с.

3. Потемкин А. Е. Твердотельное моделирование в системе КОМПАС-3D. СПб.: БХВ-Петербург, 2004. 512 с.

4. Талалай П. Г. КОМПАС-3D V11 на примерах. СПб.: БХВ-Петербург, 2010. 624 с.

V. P. Bolshakov, A. V. Chagina
Saint Petersburg Electrotechnical University «LETI»

COMPUTER PRACTICUM FOR TRAINING WORK IN THE KOMPAS-3D SYSTEM

In the study of engineering computer graphics CAD-systems (Computer Aided Design) are used and also methods for the preparation of design documentation of products based on 3D-modeling of these products. The effectiveness of mastering the basics of 3D modeling can be enhanced by using electronic workshops or «Azbuk» for training. They allow the novice user of a 2D or 3D editor to perform the necessary actions using the information from the help window. An overview of the content of the «Azbuk» included in the various versions of the Kompas-3D system is presented. Section 1 of the workshop discusses the techniques for creating 12 3D-models and associative drawings of parts. Section 2 in six examples shows the techniques for creating models of parts and assembly units from sheet material. In section 3 of the workshop, the stages of creating 3D-models and design documentation of assembly units are considered. Section 4 presents examples of two test tasks.

Solid modeling, engineering and computer graphics, Kompas-3D

УДК 681.324(03)

В. А. Дубенецкий, А. Г. Кузнецов, В. В. Цехановский
Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В. И. Ульянова (Ленина)

Технология формирования проектной модели классов информационной системы

Рассматривается технология проектирования информационных систем, основанная на использовании моделей, допускающих исполнение. В качестве базового языка для описания компонентов предметной области, шаблонов используется язык UML. Показано, что расширение UML посредством введения новых стереотипов и правил работы с ними позволяет обеспечить необходимые свойства описания и преобразования объектных моделей информационных систем. Предлагаются приемы преобразования объектной модели информационной системы этапа анализа для получения объектной модели этапа проектирования, обладающей высоким уровнем абстрагирования, расширяющей границы повторного использования. Приводится пример построения объектной модели высокого уровня абстрагирования последовательным применением правил преобразования исходной модели этапа анализа. За счет применения некоторых паттернов и предложенных правил преобразования модель для работы с тарифами на грузоперевозки, построенная на основе конкретного документа, преобразуется в модель для работы с тарифами на услуги.

Информационная система, объектная модель, модель классов, этапы проектирования, преобразования объектной модели

Создание, внедрение и сопровождение информационных систем (ИС), используемых в управлении деятельностью организаций, является трудоемким и затратным процессом. Неправиль-

ный выбор архитектурных решений может привести к тому, что проектируемая система станет заказной с оригинальными метаданными и кодом реализации. Кроме того, классические решения приводят к тому, что модель домена предметной области, претерпевающая изменения, растворяется в коде приложения. Обеспечить согласование этой модели с моделью, реализованной в приложении, становится трудно и требует внесения изменений в код и метаданные приложения. Разрыв между доменом проблемы и доменом реализации остается достаточно большим. Разнообразие процессов, объектов и событий даже в рамках одного домена предметной области, для поддержки которой строится информационная система, велико.

Одним из направлений решения этих проблем является использование технологии проектирования информационных систем, основанное на использовании моделей, допускающих исполнение (MDD). В [1, с. 148], в частности, определяется основная особенность MDD: «… вместо выражения реализации на языках, определенных технологиями реализации, они должны выразить спецификации на языках, определенных намерениями». В информационную систему встраиваются конструкторы шаблонов, доступные на этапе исполнения специалистам предметной области. Эти шаблоны фактически предоставляют доменные языки для решения разнообразных задач управления деятельностью. В качестве базового языка для описания компонентов предметной области, шаблонов используется UML. В [2] выделяется подход UML PIM, который использует UML (Unified Modeling Language) для построения не зависимой от платформы модели (Platform Independent Model, PIM), которая преобразовывается (или реализуется человеком) в исходные коды. В качестве основной среды реализации модели и вычислений используется расширенный язык SQL.

Будем рассматривать упрощенный цикл разработки ИС, включающий в себя следующие этапы: анализ и разработка требований; проектирование; разработка. На каждом из этапов используется свой комплекс моделей. Важно, чтобы модели были взаимосвязаны. Все проблемы, выявленные и зафиксированные в модели анализа, должны быть поддержаны в модели этапа проектирования. В свою очередь, все свойства модели проектирования должны быть реализованы в модели реализации. Качество проектной модели можно проверить посредством расширения и уточнения модели этапа анализа и оценки возможности моделирования этих изменений средствами модели этапа проектирования. Модель этапа реализации отражает особенности платформы реализации, дизайна пользовательского интерфейса, алгоритмов обработки данных.

В [1, с. 128] делается следующий вывод: «Несмотря на огромное количество инструментария UML, этот язык так и не повысил уровня абстракции, на котором разработчики пишут исходные артефакты, и не позволил автоматизировать разработку. Язык UML был спроектирован для документирования, а не для разработки».

Не соглашаясь с такими выводами, постараемся показать, что UML при правильном подходе к его расширению позволяет существенно повысить уровень абстрагирования проектных решений ИС и автоматизировать разработку информационных систем.

Для повышения моделирующих возможностей потребуется определенное расширение метамодели UML. Одним из направлений расширения является формулирование правил оперирования метаклассами и их взаимосвязями. Варианты расширения рассмотрены, например, в [3, с. 20]. Сконструированные шаблоны должны обладать высоким уровнем абстрагирования и повторного использования. С их помощью конкретные варианты моделей предметной области в рамках проекта могут быть реализованы значительно быстрее, точнее, с меньшим количеством ошибок.

Если проектировать приложение нелегко, инструментальные библиотеки – еще сложнее, то проектирование каркасов – задача самая трудная [4, с. 42].

Рассмотрим некоторые приемы преобразования модели классов для получения нужных свойств по гибкости и расширяемости. Использование известных паттернов, несомненно, помогает в решении задач преобразования объектной модели. Дополнительно выделим ряд полезных правил преобразования:

1. В качестве исходной необходимо построить модель классов для фрагмента предметной области (например, для выбранного бизнес-процесса или прецедента). В качестве источника данных можно использовать образцы документов, применяемые в выбранном бизнес-процессе. Желательно воздержаться от преобразований при построении модели. Модель должна точно отражать структуры источника.

2. Выделение зависимостей. Выявление n-арных зависимостей. Выявление рекурсивных зависимостей. Выявление агрегатов. Выявление

ограничений. Выявление в бинарных зависимостях n-арности больше двух.

3. Анализ перечислений на принадлежность к квалификаторам (классификаторам).

4. Выделение отношений обобщения ассоциаций.

5. Выявление и описание единиц измерения свойств численных типов.

6. Анализ возможных расширений полученной модели в части появления новых классов-сущностей, ассоциаций на основе анализа других документов данной предметной области.

7. Выявление узких мест в исходной модели, препятствующих поддержке новых выявленных элементов и связей.

8. Выявление ассоциативных классов.

9. Расширение состава классов-сущностей на основе обобщения и конкретизации. Введение классификаторов.

Рассмотрим пример последовательного формирования модели классов этапа анализа и модели классов этапа проектирования на основе исходных данных, представленных в виде документа «Тарифы на перевозку грузов». Фрагмент документа представлен на рис. 1. Для построения модели классов проведем анализ этого документа. Выделяем классы-сущности. Основным классом является *Тарифы_на_перевозку*.

Будем считать, что каждая транспортная фирма предоставляет услуги по перевозке в соответствии со своими тарифами. Следующий класс-сущность – *Транспортная_фирма*. Рассмотрим как описываются тарифы на перевозку в представленном документе. Каждый вариант тарифа представлен в виде строки таблицы. Однако строки сгруппированы по классам авто. Введем ассоциативный класс *Группа_по_транспорту* и ассоциацию с ролью *Группы_по_транспорту*. Введем подчиненный класс *Позиция_тарифа* и композицию с ролью *Варианты_тарифов*. Таблица тарифов содержит 7 полей. Поле «Грузоподъемность, объем» является агрегатом и делится на 3 составляющие: *Вес_груза, Объем_груза, Количество_паллет*. Заметим, что каждый атрибут класса *Позиция_тарифа* в качестве типа имеет соответствующую единицу измерения. Для описания единиц измерения введем класс *Единица_измерения*. Поля 2, 3 и 4, 5 могут быть представлены атрибутами класса *Позиция_тарифа.* Однако видно, что эти поля скрывают n-арную зависимость. Можно выделить класс *Вариант_интервала,* в котором указано значение временного интервала в часах. Тогда для класса *Позиция_тарифа* объявляется ассоциация *Для_варианта_интервала*: *Вариант_интервала*. Для описания стоимости вводим атрибут *Стоимость_на_интервале* с единицей измерения р./ч. Аналогично, для полей 6,7 необходимо выделить класс *Вариант_маршрута*, в котором в качестве атрибута указан конкретный вариант маршрута. Для описания стоимости перепробега вводим атрибут *Стоимость_перепробега* с единицей измерения р./км. Построенная модель позволяет учесть особенности описания тарифа, отраженные в полях таблицы тарифов. Документ «Таблица тарифов» представляет данные по иерархической схеме. Вторая строчка таблицы указывает на класс авто, для которого описаны тарифы. Поэтому в объектную модель придется ввести метакласс *Классификатор_авто*, а для класса *Позиция_тарифа* ввести ассоциацию с ролью *Для_класса_авто: Классификатор*. Вариант модели классов для работы с тарифами на перевозки представлен на рис. 2.

Рассмотрим возможности расширения области применения полученной модели. Построим объектную модель этапа анализа для работы с тарифами на грузоперевозку. Выделяем классы-сущности. Основным классом является *Тарифы_на_перевозку*. Следующий класс-сущность –

Грузподъемность, объем	Тариф 8 ч, р.	Тариф 4 ч, р.	Стоимость часа на тарифе 8, р.	Стоимость часа на тарифе 4, р.	Стоимость километра перепробега по городу, р./км	Стоимость километра перепробега по области, межгороду, р./км
Закрытые автомобили						
0.5 т	3160	1900	395	420	14	16
1 т	3480	2120	435	470	15	17
1.5 т. до 10 м^3 до четырех паллет	3960	2220	495	495	16	18

Рис. 1

Транспортная_фирма. Каждый вариант тарифа представлен в виде строки таблицы. Однако строки сгруппированы по классам авто. Для описания списка классов авто введем перечисление *Класс_авто*, ассоциативный класс *Группа_по_транспорту* и ассоциации с ролями *Группы_по_транспорту, Для_класса_авто*. Далее введем подчиненный класс *Позиция_тарифа* и композицию с ролью *Состав_тарифа*. Таблица тарифов содержит 7 полей. Поле «Грузоподъемность, объем» является агрегатом и делится на 3 составляющие: *Вес_груза, Объем_груза, Кол_паллет*. Однако значения этих полей задают ограничения на параметры груза, поэтому введены следующие атрибуты: *Вес_груза_от, Объем_груза_от, Кол_паллет_от, Вес_груза_до, Объем_груза_до, Кол_паллет_до*. Заметим, что каждый атрибут класса *Позиция_тарифа* в качестве типа имеет соответствующую единицу измерения. Построенная модель позволяет вести данные в соответствии с исходным документом. Однако для расчета стоимости перевозки требуется использовать знания об особенностях полей (например, выбор расценки по ожидаемому времени перевозки). Для описания единиц измерения введем класс *Единица_измерения*.

Рассмотрим возможности расширения области применения полученной модели. Перейдем к преобразованию (рис. 2). Поля *Стоимость_перепробега_по_городу_р/км, Стоимость_перепробега_по_области_р/км, Стоимость_на_тарифе_8ч_р./ч, Стоимость_на_тарифе_4ч_р./ч* (номера 7, 8, 9, 10) скрывают n-арную зависимость. Можно выделить класс *Вариант_интервала,* в котором указано значение временного интервала в часах. Тогда для класса *Позиция_тарифа* объявляется ассоциация *Для_варианта_интервала*: *Вариант_интервала*. Для описания стоимости вводим атрибут *Стоимость_на_интервале* с единицей измерения р./ч. Аналогично, для полей 7, 8 необходимо выделить перечисление *Вариант_маршрута* и ассоциацию с ролью *Для_варианта_маршрута*. Для описания стоимости перепробега вводим атрибут *Стоимость_перепробега* с единицей измерения р./км. Вместо перечисления *Класс_авто* введем метакласс *Классификатор_авто* с рекурсивной зависимостью по обобщению. Построенная модель позволяет учесть особенности описания тарифа, отраженные в полях таблицы тарифов. Расширяются возможности для описания временных интервалов, вариантов маршрутов, схем классификации авто.

Второй вариант модели классов для работы с тарифами на перевозки представлен на рис. 3. Данная модель позволяет упростить расчет стоимости перевозки, так как выбор тарифа непосредственно связан с параметрами груза.

В модели учтены следующие расширения: каждая транспортная фирма может иметь несколько тарифов и задавать свои значения стоимостных показателей; список классов авто открыт и может дополняться.

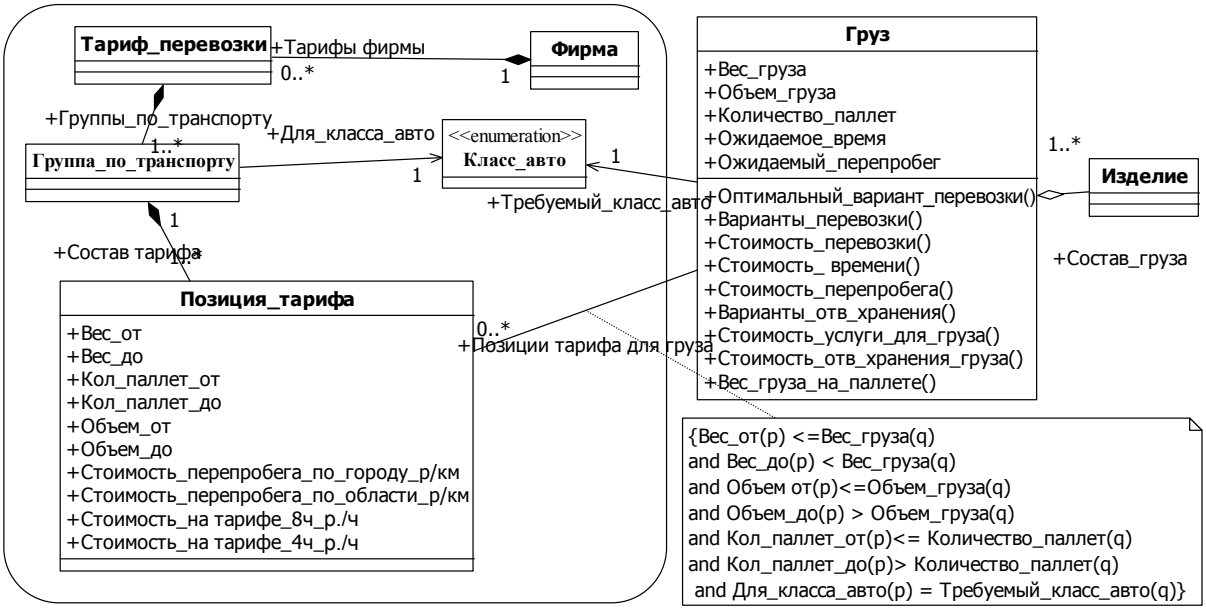

Рис. 2

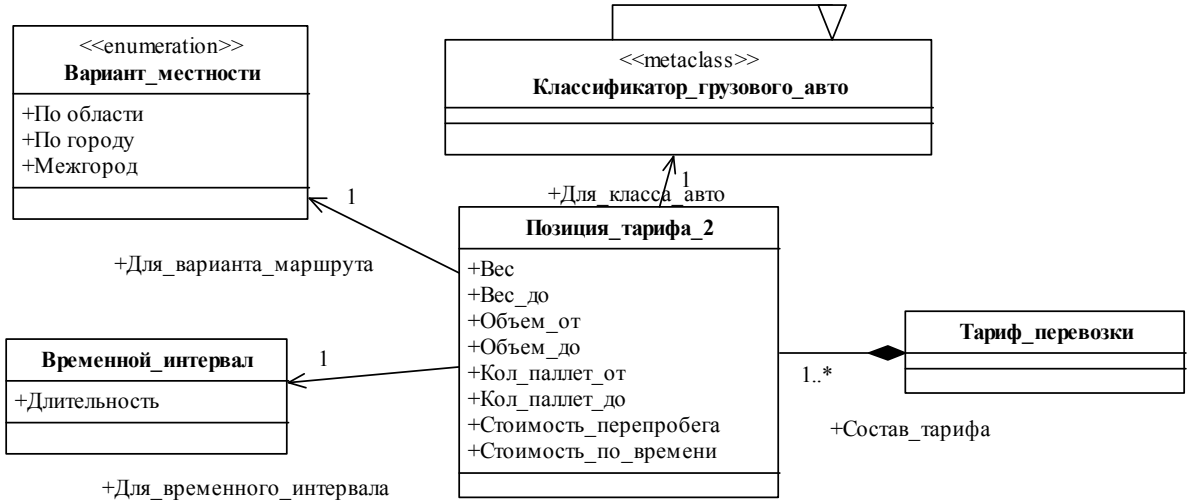

Рис. 3

Несложно сформировать на основе модели классов (рис. 3) модель данных (ERD) и скрипты базы данных объектно-реляционного моделирования (ORM). Однако любые изменения в метаданных модели тарифов потребуют изменить структуру базы данных, модифицировать и расширить набор поддерживающих SQL-процедур.

Разработка модели этапа проектирования для ведения тарифов на услуги. Рассмотрим «узкие места» модели, представленной на рис. 3, и выделим точки расширения и абстрагирования этой модели. Данная модель ориентирована на такой вариант услуги, как *Грузоперевозки*. Введем метакласс *Классификатор_услуг* и ассоциацию с ролью *Услуга_по_тарифу*. Это расширение позволит вести классификатор услуг и создавать тарифы по выбранному классу услуг. Следующее узкое место в исходной модели связано с ограничением на состав параметров класса *Позиция_тарифа*. Этот состав определен в метаданных и рассчитан только на параметры конкретной структуры тарифа на грузоперевозки, хотя и описан в метаданных класса *Позиция_тарифа*. Следующий шаг расширения – введение специального метакласса *Параметр*. Этот метакласс позволит описывать и вести глобальный список параметров и ссылаться на него при описании введенных классов услуг. Ассоциативный класс *Параметр_услуги*, ассоциации с ролями *Параметр_для_услуги* и *Состав_параметров* поддержат сервисы описания характеристик классов услуг. *Квалификатор* для отбора позиций тарифа по параметрам услуги представляет собой конъюнкцию предикатов, определенных на параметрах класса *Позиция_тарифа* и параметрах класса *Услуга*. Область значений параметров определяется через роль *Тип_параметра*. Расширенная модель метаклассов для примера представлена на рис. 4. Для вещественных параметров требуется дополнительно ввести ряд ограничений, позволяющих уточнить спецификацию вещественных параметров для услуги (атрибуты *Мин, Макс*). Для параметров типа *Перечисление* необходимо разработать свою локальную модель. Для описания структуры тарифов вводятся метаклассы *Шаблон_тарифа, Шаблон_предиката*, ассоциации с ролями *Состав_выходных_параметров, Состав_шаблонов_предикатов*.

Модель метаклассов, представленная на рис. 4, обладает существенно более высоким уровнем абстрагирования по сравнению с исходной моделью этапа анализа (рис. 2). Новый вариант модели позволяет создать конструктор, описывающий структуру различных классов услуг, и на основе этих шаблонов создавать конкретные тарифы, описывать параметры конкретных услуг и решать задачи поиска подходящих тарифов для этих услуг. Набор сервисов расширился. Процедурная поддержка этой модели не зависит от конкретных структур тарифов. Разработанная модель допускает исполнение и предоставляет своеобразный язык для описания тарифов и работы с ними. Однако остались еще «узкие места» в представленной модели.

Использование объектной модели правил. Существенным ограничением является то, что позиция тарифа описывается в виде конъюнкции предикатов, определенных на параметрах класса услуги. В более общем случае для описания позиций тарифов можно применить механизм про-

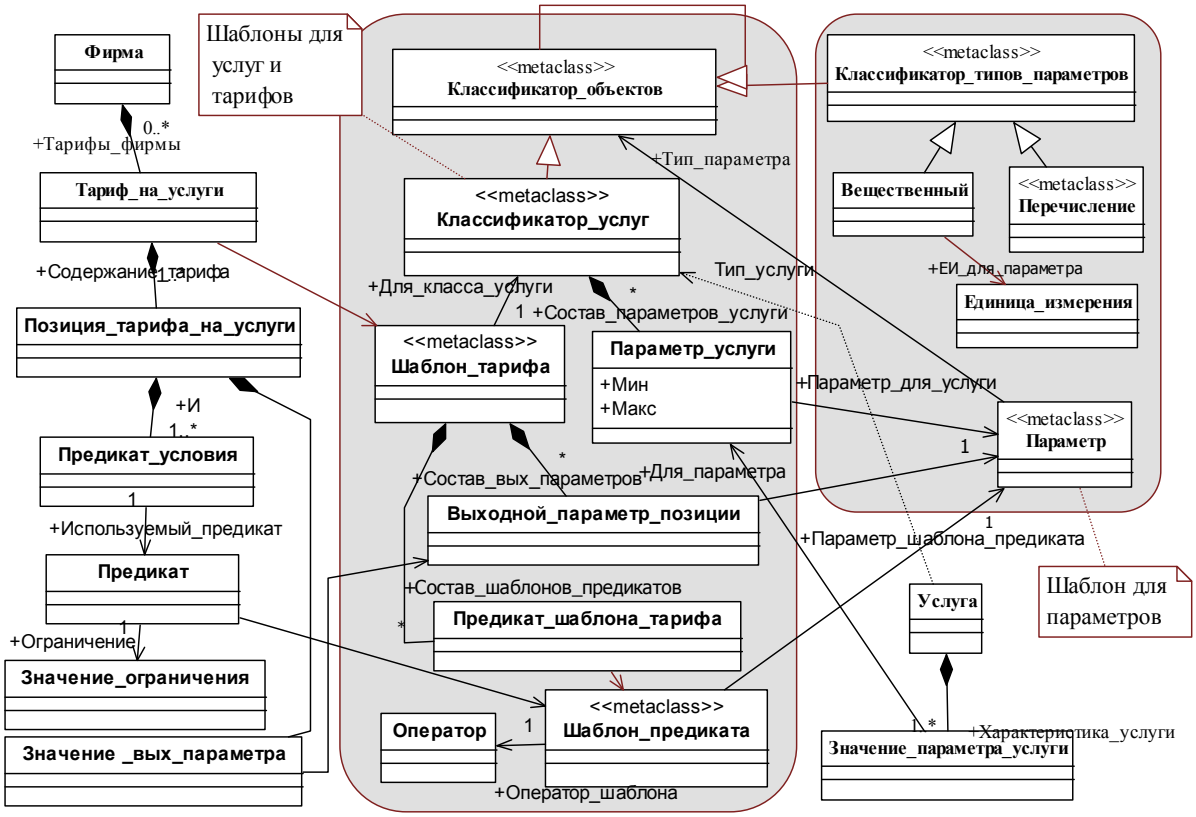

Рис. 4

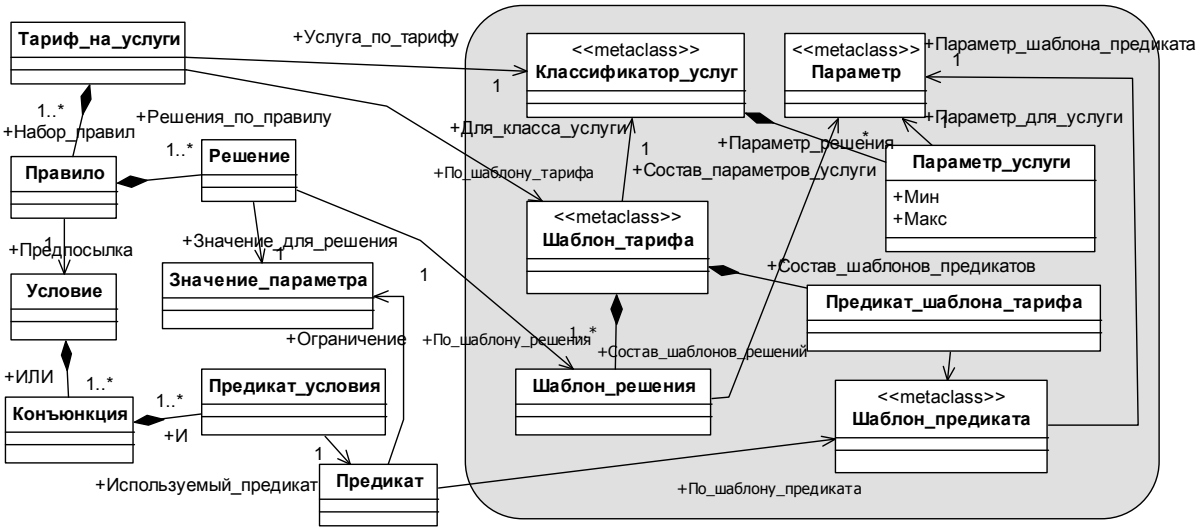

Рис. 5

дукций. Использование модели правил позволяет существенно улучшить свойства проектной модели и снять ряд ограничений при описании и работе с нормативными данными. На рис. 5 представлен фрагмент модели метаклассов, в котором использована модель правил. Класс *Правило* позволяет работать с правилами, определенными на компонентах модели тарифов. Класс *Условие* и ассоциация с ролью *Предпосылка* позволяют описать предпосылку правила. Класс *Решение* и ассоциация с ролью *Решения_по_правилу* позволяют описать решения для каждого введенного правила. В свою очередь условие описывается в виде сокращенной дизъюнктивной нормальной формы. Для этого в модель введены классы *Конъюнкция*, *Предикат* и ассоциации с ролями *ИЛИ* и *И*.

Каждое решение представляется в виде значения для соответствующего параметра.

Вернемся к рассмотрению модели классов для примера, представленной на рис. 2. Объектная модель этапа анализа позволила учесть все особенности работы с тарифами на грузоперевозки. Модели метаклассов, представленные на рис. 3–5, показывают последовательные шаги обобщения проектной модели для повышения уровня ее абстрагирования. Однако даже модель, представленная на рис. 5 и содержащая класс *Правило*, может быть применена только для описания тарифов на услуги. При необходимости работу по повышению уровня абстрагирования модели можно продолжить.

Выявлены основные проблемы технологии проектирования информационных систем. Предложен оригинальный подход к проектированию информационных систем посредством встраивания конструкторов шаблонов, доступных на этапе исполнения предметной области. Эти шаблоны фактически предоставляют доменные языки для решения разнообразных задач управления деятельностью.

СПИСОК ЛИТЕРАТУРЫ

1. Гринфилд Дж., Шорт К. Фабрики разработки программ. Потоковая сборка типовых приложений, моделирование, структуры и инструменты / пер с англ. М.: ООО «И. Д. Вильямс», 2007. 592 с.

2. Фаулер М. Языковые фабрики и управляемая моделью архитектура / пер. с англ. URL: http://masters.donntu.edu.ua/2010/fknt/frunt/library/article2.htm (дата обращения 03.02.19).

3. Дубенецкий В. А., Цехановский В. В. Объектно-ориентированные модели корпоративных бизнес-процессов. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2014. 161 с.

4. Приемы объектно-ориентированного проектирования. Паттерны проектирования / Э. Грамма, Р. Хельм, Р. Джонсон, Д. Влиссидес. СПб.: Питер, 2008. 366 с.

V. A. Dubenetsky, A. G. Kuznetsov, V. V. Tsekhanovsky
Saint Petersburg Electrotechnical University «LETI»

TECHNOLOGY OF FORMATION OF A DESIGN CLASS MODEL OF THE INFORMATION SYSTEM

The technology of information systems design based on the use of models that allow execution is considered. The UML language is used as the base language for description of domain components and templates. It is shown that the expansion of UML by introducing new stereotypes and rules of work with them allows to provide the necessary properties of the description and transformation of object models of information systems. The methods of transformation of the object model of the information system of the analysis stage to obtain the object model of the design stage, which has a high level of abstraction, expanding the boundaries of reuse. An example of constructing a high-level object model of abstraction by successive application of the transformation rules of the original model of the analysis stage is given. By applying some of the patterns and proposed transformation rules, a freight tariff model based on a specific document is transformed into a service tariff model.

Information system, object model, class model, design stages, object model transformation