УНИВЕРСИТЕТ ИТМО
НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
ноябрь–декабрь 2015 Том 15 № 6 ISSN 2226-1494 http://ntv.ifmo.ru/
SCIENTIFIC AND TECHNICAL JOURNAL OF INFORMATION TECHNOLOGIES, MECHANICS AND OPTICS
November–December 2015 Vol. 15 No 6 ISSN 2226-1494 http://ntv.ifmo.ru/en

**УДК 004.04**

# ПРЕДМЕТНО-ОРИЕНТИРОВАННОЕ ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ ИНФОРМАЦИОННЫХ СИСТЕМ ДЛЯ ПРЕДМЕТНЫХ ОБЛАСТЕЙ МАССОВОГО ОБСЛУЖИВАНИЯ КЛИЕНТОВ

**П.П. Олейник**[a]

[a] Шахтинский институт (филиал) Южно-Российского государственного политехнического университета им. М.И. Платова, Шахты, 346500, Российская Федерация
Адрес для переписки: xsl@list.ru

**Информация о статье**
Поступила в редакцию 20.04.15, принята к печати 15.09.15
doi:10.17586/2226-1494-2015-15-6-1105-1114
Язык статьи – русский

**Ссылка для цитирования:** Олейник П.П. Предметно-ориентированное проектирование и реализация информационных систем для предметных областей массового обслуживания клиентов // Научно-технический вестник информационных технологий, механики и оптики. 2015. Т. 15. № 6. С. 1105–1114.

**Аннотация**

Рассмотрена применимость предметно-ориентированного проектирования информационных систем для предметных областей массового обслуживания клиентов. При этом были выдвинуты следующие критерии оптимальности, которым должна соответствовать конечная реализация: возможность автоматизации с помощью единой системы как небольшого заведения, так и целой сети заведений; развитый графический интерфейс с поддержкой сенсорных экранов; многопользовательский учет заказов, поступающих от клиентов; гибкая архитектура приложения с возможностью расширения в будущем; возможность интеграции с различными периферийными устройствами. Показана необходимость определения каждого критерия. Для оценки реализуемости была спроектирована тестовая информационная система, автоматизирующая систему массового обслуживания. Использован унифицированный язык моделирования UML. Приведено описание назначения каждого класса и ассоциации с другими классами. Уделено внимание проектированию древовидных (иерархических) структур и процедуре выделения базовых классов на основе анализа имеющихся общих атрибутов. Для реализации системы использована собственная среда разработки SharpArchitect RAD Studio, предлагающая MDA-подход для систем и основанная на унифицированной метамодели объектной системы. Представлен графический вид разработанного прототипа формы заказа, описаны состав и структура, а также разработанная автором нотация, позволяющая упростить процесс прототипирования. Показаны подходы к разграничению прав доступа для различных ролей пользователей. Определено соответствие полученной реализации каждому выделенному критерию оптимальности. Даны рекомендации по дальнейшей разработке системы.

**Ключевые слова**

проектирование информационных систем, реализация информационных систем, UML, базы данных, объектно-реляционное несоответствие, методы (шаблоны, паттерны) объектно-реляционного отображения, метамодель объектной системы, DDD, MDA.

# DOMAIN-DRIVEN DESIGN APPLICATION AND IMPLEMENTATION OF INFORMATION SYSTEMS FOR CLIENTS QUEUING SUBJECT AREAS

**P.P. Oleynik**[a]

[a] Shakhty Institute (Branch) of Platov South Russian State Polytechnic University (NPI), Shakhty, 346500, Russian Federation
Corresponding author: xsl@list.ru

**Article info**
Received 20.04.15, accepted 15.09.15
doi:10.17586/2226-1494-2015-15-6-1105-1114
Article in Russian
**For citation:** Oleynik P.P. Domain-driven design application and implementation of information systems for clients queuing subject areas. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2015, vol. 15, no. 6, pp. 1105–1114.

**Abstract**

The paper deals with domain-driven design applicability of information systems for client queuing subject areas. The following optimality criteria were put forward for the final implementation: the possibility of automation with a single system both for small institution and a whole network of institutions; advanced graphical interface with support for sensor screens; implementation of multi-users account of orders from clients; flexible application architecture with the ability of future

enhancement; ability of integration with a variety of peripherals. The necessity of each criterion definition is shown. For implementability estimation, test information system was designed, automating the queuing system. Unified modeling language UML is used. Description of each class functionality is given and the association with other classes as well. Attention is paid to the design of tree (hierarchical) structures and selection procedure of base classes based on the analysis of existing common attributes. For the system implementation, its own development environment SharpArchitect RAD Studio is used, offering MDA approach for implementation of systems based on standardized meta object system. A graphical view of order form developed prototype is presented, composition and structure are described, and notation developed by the author is given simplifying the prototyping process. Approaches to differentiation of access rights for different user roles are shown. Conformity of the received implementation to each selected optimality criterion is determined. Recommendations for further system development are given.

**Keywords**
design of information systems, implementation of information systems, UML, databases, object-relational mismatch, object-relational mapping patterns, object system metamodel, DDD, MDA.

## Введение

Посещение ресторанов быстрого питания и супермаркетов стало неотъемлемым элементом современной жизни. Это привело к повсеместному появлению подобных заведений, и как следствие, перед руководителями возникает задача автоматизации деятельности, выбор и внедрение информационной системы (ИС). От успешного решения этой задачи зависит успех всего бизнеса. Как правило, существуют два основных решения – выбор и внедрение существующей системы или самостоятельная разработка программного продукта для нужд организации.

Настоящая работа представляет опыт проектирования и реализации ИС для предметных областей массового обслуживания. При этом применено предметно-ориентированное проектирование (Domain-Driven Design, DDD). Для его реализации выбрана архитектура, управляемая моделью (Model-Driven Architecture, MDA), используемая в реализованной автором среде разработки SharpArchitect RAD Studio. Эта система использует развитую метамодель объектной системы, позволяющей проектировать систему с помощью создания экземпляров метаклассов.

В работе представлена UML-диаграмма классов, содержащая все выделенные классы и позволяющая продемонстрировать разработанный функционал. Кроме того, выполнено прототипирование основной формы приложения, позволяющей управлять заказами клиентов. Использована разработанная автором графическая нотация, которая в удобном виде показывает привязку атрибутов и методов классов к графическим элементам, отображаемым пользователю и позволяющим реализовать графический интерфейс пользователя для сенсорных экранов.

## Обзор имеющихся работ

Системы обслуживания клиентов используются повсеместно, поэтому интерес к их изучению, формализации и автоматизации предметных областей возник довольно давно. С точки зрения разработки информационных систем предметная область может быть представлена в виде рис. 1. Через информационную систему (IS) последовательно регистрируются заказы (O) клиентов (C). Каждый сотрудник (W) последовательно получает свой заказ и исполняет его параллельно на исполнительном устройстве (E). Выполненный заказ (EO) последовательно регистрируется работником и выдается клиенту.

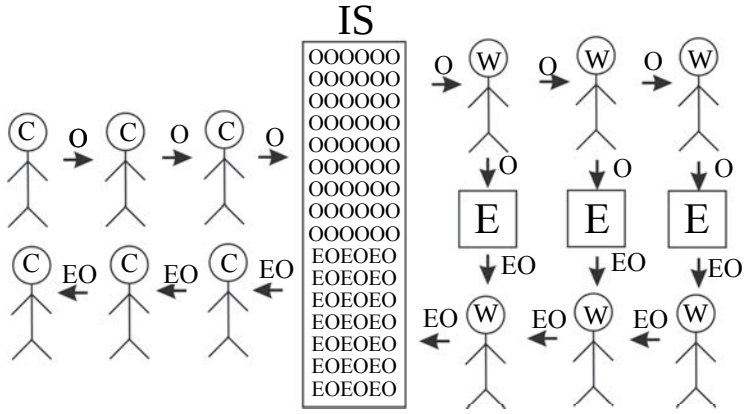


Рис. 1. Использование информационных систем в предметной области массового обслуживания клиентов

В работе [1] автор обосновывает необходимость создания программного комплекса, позволяющего имитировать поведение посетителей ресторанов и создающего различные типовые операции бизнес-процессов. Описаны состав и структура модулей ИС, поток выполнения различных тестов, проблемы валидации и верификации созданных моделей.

В [2] представлена система, реализованная в корпорации Toshiba ИС, которая распознает устную речь клиента и на основе выделенных корней формирует запросы на создание заказов в системе обслуживания клиентов. Работа подробно раскрывает процесс выделения ключевых слов, определения корня слов и рассматривает пример распознания голоса и последующего распарсивания слов и последовательностей. Затем на основе имеющейся грамматики находится соответствие с продуктами в ИС ресторана, и формируется заказ на приготовление.

Более поздняя работа [3] представляет собственный метод оценки удовлетворенности клиентов услугами ресторанов быстрого обслуживания, который авторы назвали TOPSIS. Этот подход основан на многофакторном анализе, при котором все выделенные критерии распределены на несколько групп. Авторы предлагают собственный формальный аппарат для ранжирования критериев в пределах группы на основе рассчитанного веса и получения интегрированного показателя. Подход апробирован в ресторанах США и Китая.

В [4] подробно описана реализация многозвенной ИС, позволяющей полностью автоматизировать систему массового обслуживания и все бизнес-процессы – от формирования заказа клиентом до получения заказа. В начале работы авторы рассматривают процесс формирования заказов с точки зрения различных участников, таких как клиент и обслуживающая система. В результате найденных отличий делается вывод о необходимости реализации различных звеньев системы и различных интерфейсов.

В работе [5] авторы рассмотривают процесс разработки программного продукта, который использует карманные персональные компьютеры (КПК, PDA) для создания и обслуживания заказов клиентов. Представлен сенсорно-ориентированный графический интерфейс, описаны основные задачи, выполняемые с его помощью.

Проблема внедрения современных сенсорно-ориентированных приложений рассмотрена и в работе [6], посвященной процессу автоматизации системы обслуживания клиентов на примере столовой. Авторы выделяют в бизнес-процессе ключевые процедуры и распределяют их на несколько групп, каждая из которых выполняется либо с помощью сенсорно-ориентированного приложения, либо с помощью приложения, использующего манипулятор «мышь». С аппаратной точки зрения предложено сделать специальные виды обеденных столов, схема которых также представлена в работе [6].

В [7] авторы выполнили реинжиниринг процесса приготовления комплексных обедов, а затем на основании построенных диаграмм осуществили оптимизацию процесса. Это позволило построить диаграмму «сущность-связь» в понятиях реляционной модели данных.

Одной из последних работ, посвященных разработке ИС для ресторанов, является работа [8]. Авторы описывают собственный продукт, который позволяет заказывать товары самостоятельно с помощью приложений, запущенных на сенсорных устройствах. При этом в начале (только один раз) необходимо ввести дополнительную информацию о себе, которая потом используется при формировании заказов и сокращает тем самым время обслуживания.

Работа [9] посвящена моделированию системы массового обслуживания для ресторанов быстрого питания. В ней построена модель очереди обслуживания клиентов типового реально существующего кафе. В качестве математического аппарата использовалась модель марковских процессов.

Наиболее популярной коммерческой ИС, автоматизирующей деятельность ресторанов в России, которая успешно используется в заведениях массового обслуживания, является система iiko (Автоматизация ресторана iiko, http://iiko.ru/).

**Формирование требований к ИС для предметных областей массового обслуживания**

Перед реализацией любого программного продукта необходимо выделить требования (КО), которые определяют функциональные особенности будущей реализации. Требования были выдвинуты после изучения функционала, имеющегося у программного продукта iiko, а также иных продуктов, построенных на базе 1С. Для ИС, автоматизирующих предметные области обслуживания клиентов, выделены следующие требования:

1. предусмотреть возможность автоматизации с помощью единой системы как небольшого заведения, так и целой сети заведений ($КО_1$);
2. разработать развитый графический интерфейс с поддержкой сенсорных экранов ($КО_2$);
3. реализовать многопользовательский учет заказов, поступающих от клиентов ($КО_3$);
4. реализовать гибкую архитектуру приложения с возможностью расширения в будущем ($КО_4$);
5. предусмотреть возможность интеграции с различными периферийными устройствами ($КО_5$).

Рассмотрим каждое требование более подробно. Требование $КО_1$ предполагает использование единой ИС независимо от масштабов организации – от одиночного ресторана до крупной сети. Часто подобные требования называют масштабируемостью системы. Ключевым достоинством в свете рассматриваемой задачи является снижение затрат на обучение персонала при естественном расширении организации.

Применение сенсорных экранов позволяет исключить клавиатуру и мышь и управлять пальцами рук. Это значительно сокращает время на выполнение типовых бизнес-процессов, что очень важно в системах массового обслуживания. Именно поэтому в ИС необходим развитый графический интерфейс с поддержкой сенсорных экранов, что описано в требовании $КО_2$. Анализ функциональных возможностей самой популярной в России системы iiko, а также наблюдение за устройствами, используемыми в различных предметных областях систем массового обслуживания, показали, что разрешение используемых сенсорных экранов составляет 1024×768 пикселей, и на нем необходимо оптимально поместить выводимую информацию и элементы графического интерфейса пользователя.

Бизнес-процессы систем обслуживания клиентов в упрощенном виде предполагают наличие одного компьютера с установленным программным обеспечением, доступ к которому имеют множество пользователей системы, оформляющих и обслуживающих заявки клиентов. Следовательно, необходим механизм идентификации и разграничения прав доступа. Для этого достаточно каждому пользователю системы присвоить уникальный код, который может быть введен любым удобным способом. Чтобы сократить время на ввод кода, часто используются прокси-карты и их считыватели. Пользователь, подходя к компьютеру, подносит личную прокси-карту к считывающему элементу и, авторизовавшись в системе, выполняет требуемые действия. Затем он выходит из системы, нажав кнопку на графической форме, и подходит следующий пользователь. Следовательно, скорость обслуживания во многом зависит от скорости входа в систему, и использование прокси-карт является оптимальным решением в данном случае. Таким образом, можно реализовать оперативный многопользовательский учет заказов, что записано выше как требование $КО_3$. При этом различным группам пользователей необходимо предоставить различный интерфейс, оптимальный для выполнения их должностных обязанностей. Например, одной категории сотрудников необходимо вносить и редактировать заявки клиентов. Менеджеру необходимо лишь просматривать и иметь возможность удалять ошибочные заказы из системы.

Требование $КО_4$ состоит в реализации гибкой архитектуры приложения с возможностью расширения в будущем, например, интеграции разработанной системы с различными бухгалтерскими программами. Из-за различий в организации и обработке данных в объектно-ориентированных (ОО) языках программирования (ЯП) и в реляционных системах управления данными возникает объектно-реляционное несоответствие, для преодоления последствий которого используют методы (шаблоны, паттерны) объектно-реляционного отображения. Популярные подходы решения описанной проблемы подробно представлены в работах [10–16]. Таким образом, для реализации требования $КО_4$ одним из решений является разработка клиентского приложения на ООЯП, а в качестве хранилища информации выбирается реляционная СУБД.

Возможность интеграции с различными периферийными устройствами особенно актуальна в нашем случае, так как к рабочему месту пользователя ИС подключены принтер чеков (по USB или RS-232), использующий для вывода на печать рулон бумаги шириной 8 см, и считыватели прокси-карт. Разнообразие периферийных устройств постоянно растет, что связано с уменьшением скорости печати заданий и тем самым приводит к сокращению времени обслуживания клиентов. Отсюда понятна важность требования $КО_5$.

Для оценки соответствия полученной реализации сформированному набору требований необходимо рассмотреть полученные архитектурные решения и обсудить их с разработчиками сходных ИС.

**Выбор подхода и инструмента разработки**

В настоящее время при проектировании новых программных продуктов на ООЯП используется предметно-ориентированный подход (Domain-Driven Design, DDD) [17]. Разработчики создают программные абстракции ( модели предметных областей), которые представляются в виде диаграмм классов унифицированного графического языка моделирования UML. Подход DDD полезен в ситуациях, когда разработчик программного продукта не является экспертом в прикладной предметной области разрабатываемого продукта. Программист не может знать все области, но с помощью правильного представления структуры может быть спроектировано работоспособное приложение. Одним из достоинств данного подхода, по мнению автора, является поддержка объектно-ориентированного проектирования и программирования, что позволяет с помощью инкапсуляции, наследования и полиморфизма создать повторно используемые фрагменты кода.

Автор настоящей работы разработал собственную унифицированную среду быстрой разработки корпоративных информационных систем SharpArchitect RAD Studio [11]. В работах [12–16] была представлена полная диаграмма классов метамодели и подробно описано назначение классов, а также ее эволюция и расширение. Здесь рассмотрим лишь значимый для данной работы функционал системы. Суть последующей разработки приложений в описанной среде заключается в создании экземпляров метаклассов. Метаклассы позволяют описать различные типы сущностей в зависимости от того, являются они сохраняемыми или нет, используются для вспомогательных целей или в виде параметров различных методов, позволяющих выполнить определенные вычисления. Добавление экземпляров метаклассов – это

визуальный процесс, т.е. разработчику предоставляется множество графических форм, позволяющих вносить только допустимые значения. В момент запуска приложения выполняется генерация программного кода на основе значений, присутствующих в метамодели, и регистрация полученных динамических библиотек, содержащих классы сущностей предметной области. Метаинформация используется также при генерации графического интерфейса пользователя. Описанная среда имеет следующие преимущества:

- автоматическое создание интерфейса пользователя на основе модели приложения с возможностью настройки конечным пользователем;
- возможность вносить изменения, а именно – описывать классы, связывать их между собой в соответствии с UML-диаграммами классов и задавать сигнатуру методов в момент выполнения приложения, при этом для вступления изменений достаточно перезапустить приложение одного конкретного пользователя (остальные могут продолжить работу);
- возможность декларативного описания валидационных правил, позволяющих проверять корректность и непротиворечивость данных в момент сохранения их в БД;
- возможность декларативного описания визуализационных правил, позволяющих выделить с помощью изменения цвета различные графические элементы на форме, а также управлять видимостью и доступностью элементов;
- множество системных (встроенных) классов, упрощающих реализацию поведения часто используемых видов сущностей (например, внутренняя поддержка древовидных структур).

На рис. 2 представлен фрагмент унифицированной метамодели объектной системы с отображением ключевых ассоциаций, важных для дальнейшего обсуждения.

Корнем иерархии, представляющим сущности предметной области, является абстрактный метакласс Class, имеющий два унаследованных:

1. InheritableClass – используется для представления метаклассов, которые поддерживают наследование;
2. NotInheritableClass – применяется для представления метаклассов, которые не могут быть унаследованы. Метакласс Enum позволяет представить ненаследуемое перечисление или множество значений одного простого типа.

Абстрактный базовый метакласс CustomAttributedClass используется для представления метаклассов, которые имеют атрибуты. Метакласс DomainClass применяется для представления классов предметной области. Экземпляры класса предметной области позволяют описывать классы сущностей (например, Клиент, Продукт, Продажа), объекты которых (например, Иванов, Хлеб) сохраняются в БД. Для упрощения описания будем называть экземпляры класса предметной области просто классами предметной области (если не предполагается иное).

Абстрактный метакласс ComputationalClass<TBaseClass> является базовым для всех вычисляемых метаклассов, т.е. тех классов, экземпляры которых не сохраняются в БД, а вычисляются в момент выполнения программы (транзиентные).

Перейдем к рассмотрению метаклассов, используемых для описания атрибутов классов. Корневым абстрактным метаклассом, представляющим атрибут, является AbstractAttribute. Унаследованные от VirtualAttribute метаклассы применяются для представления атрибутов, которые не были созданы разработчиком прикладной предметной области, а были предоставлены системой. Они необходимы для понимания метамодели и упрощения процесса разработки программного обеспечения. SystemAttribute позволяет описать атрибуты, которые являются системными и присутствуют в языке C#. Класс GeneratedAttribute применяется для представления атрибутов, которые автоматически генерируются системой. Например, при наследовании от базового древовидного класса автоматически добавляется атрибут Node, позволяющий получить дочерние узлы и, таким образом, образовать иерархическую структуру.

Для представления атрибутов, значения которых может задавать пользователь, используется абстрактный базовый метакласс ConcreteAttribute. Так как система реализуется на языке C#, то при сохранении значений в БД используются типы данных этого языка. Для описания этого момента добавлен параметрированный метакласс TypedAttribute<TDefaultValue>. TypeAttribute используется для представления свойств, значения которых могут сохранять ссылку на тип данных языка C#.

Для реализации поведения в SharpArchitect RAD Studio используются различные синтаксические конструкции и метаклассы. Корневым абстрактным метаклассом метода является Method. В настоящий момент в системе поддерживаются только методы, реализуемые в виде программного кода и представляемые в виде экземпляров метаклассов, унаследованных от CodeMethod. Метакласс VisualCodeMethod позволит создать визуальный метод, который отображается пользователю в виде графического элемента в интерфейсе. HelperCodeMethod представляет собой вспомогательный метод, который используется для вызова другими методами и свойствами, т.е. не участвует в формировании интерфейса.

В крупных приложениях часто возникает задача визуального выделения отдельных полей, столбцов или целых строк в зависимости от их значений, а также необходимость скрыть или сделать неактивными некоторые поля ввода или скрыть их из формы. Корневым базовым метаклассом является VisualizationRule. ActionsVisualizationRule позволяет описать визуализационные правила для различных

действий, таких как кнопки Сохранить, Создать новый, Редактировать и т.п. Метакласс AttributesVisualizationRule позволяет описать визуализационные правила для атрибутов класса. Экземпляры MethodsVisualizationRule позволяют определить правила для визуализационных методов (экземпляры VisualCodeMethod).

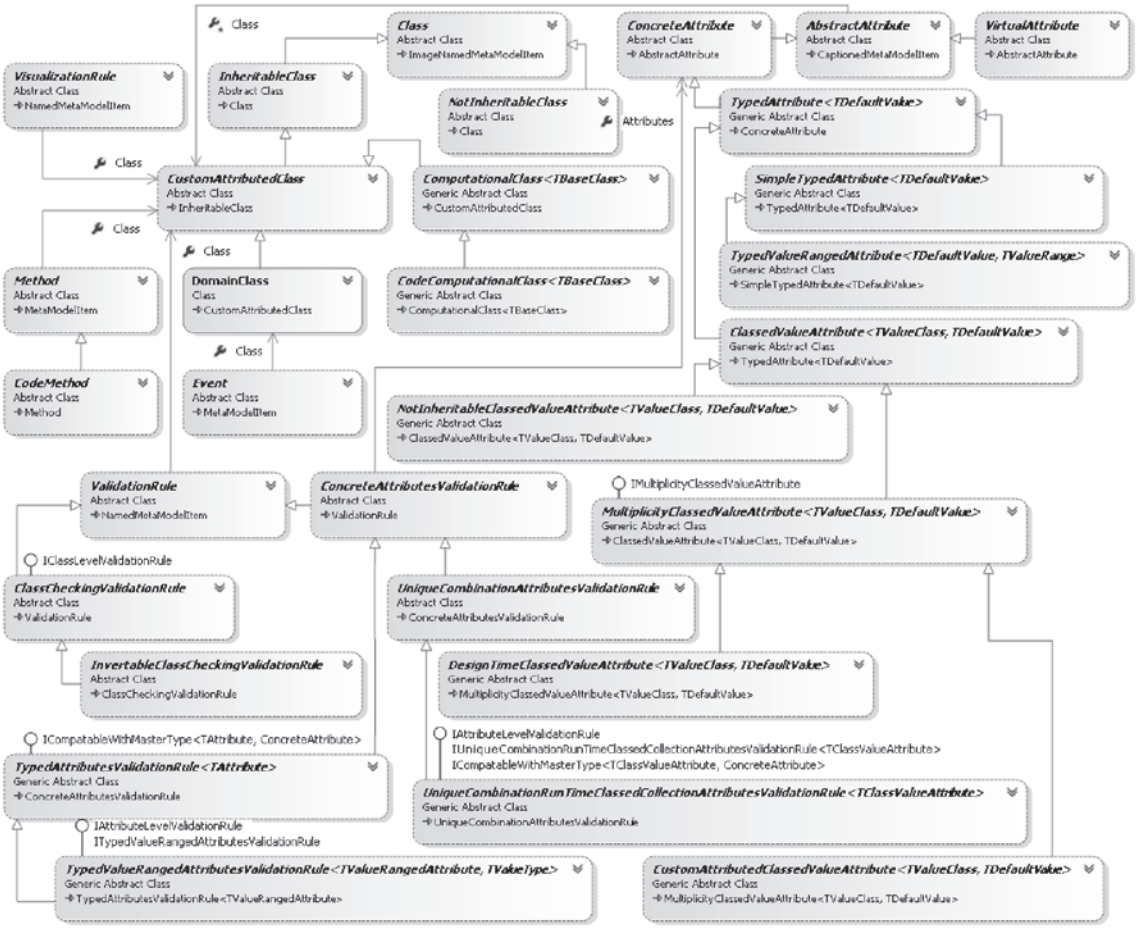


Рис. 2. Фрагмент унифицированной метамодели объектной системы

**Реализация тестовой системы, удовлетворяющей выделенным критериям**

С целью апробации выделенных требований было принято решение реализовать тестовую систему, автоматизирующую предметную область обслуживания клиентов, которая будет использована в ресторанах быстрого питания. На рис. 3 представлена диаграмма классов ИС в нотации UML. Рассматриваемая предметная область содержит множество справочников, таких как Столы, Скидки, Контрагенты, Рабочие станции, Места хранения продуктов, Единицы измерения. Все эти сущности содержат только один атрибут – Название, поэтому имеет смысл выделить базовый абстрактный класс NamedObject и унаследовать все справочники от него (рис. 3).

Рассмотрим соответствие разработанной иерархии выделенным ранее требованиям. $КО_1$ требует предусмотреть возможность автоматизации с помощью единой системы как небольшого кафе (или одного ресторана), так и целой сети заведений. Как видно из рис. 3, отсутствует какая-либо привязка к конкретной организации, т.е. удалось унифицировать систему, что и требовалось сделать.

Требование разработать развитый графический интерфейс с поддержкой сенсорных экранов предусмотрено в $КО_2$. В настоящий момент реализуется именно это требование, так как оно является основным для конечных пользователей.

С помощью отдельной структуры групп пользователей и пользователей реализуется оперативный многопользовательский учет заказов. Отметим, что разработанная иерархия ортогональна к этому требованию, поэтому реализовать $КО_3$ не составит особого труда.

В приложении реализована классическая двухзвенная архитектура «клиент–сервер», где в качестве СУБД используется Microsoft SQL Server 2012, а клиентское приложение написано на платформе .Net Framework. Следовательно, система удовлетворяет $КО_4$, так как удалось реализовать гибкую архитектуру приложения с возможностью расширения в будущем.

В настоящий момент к системе подключается множество периферийных устройств по USB и COM-порту. В результате реализовано требование $КО_5$.

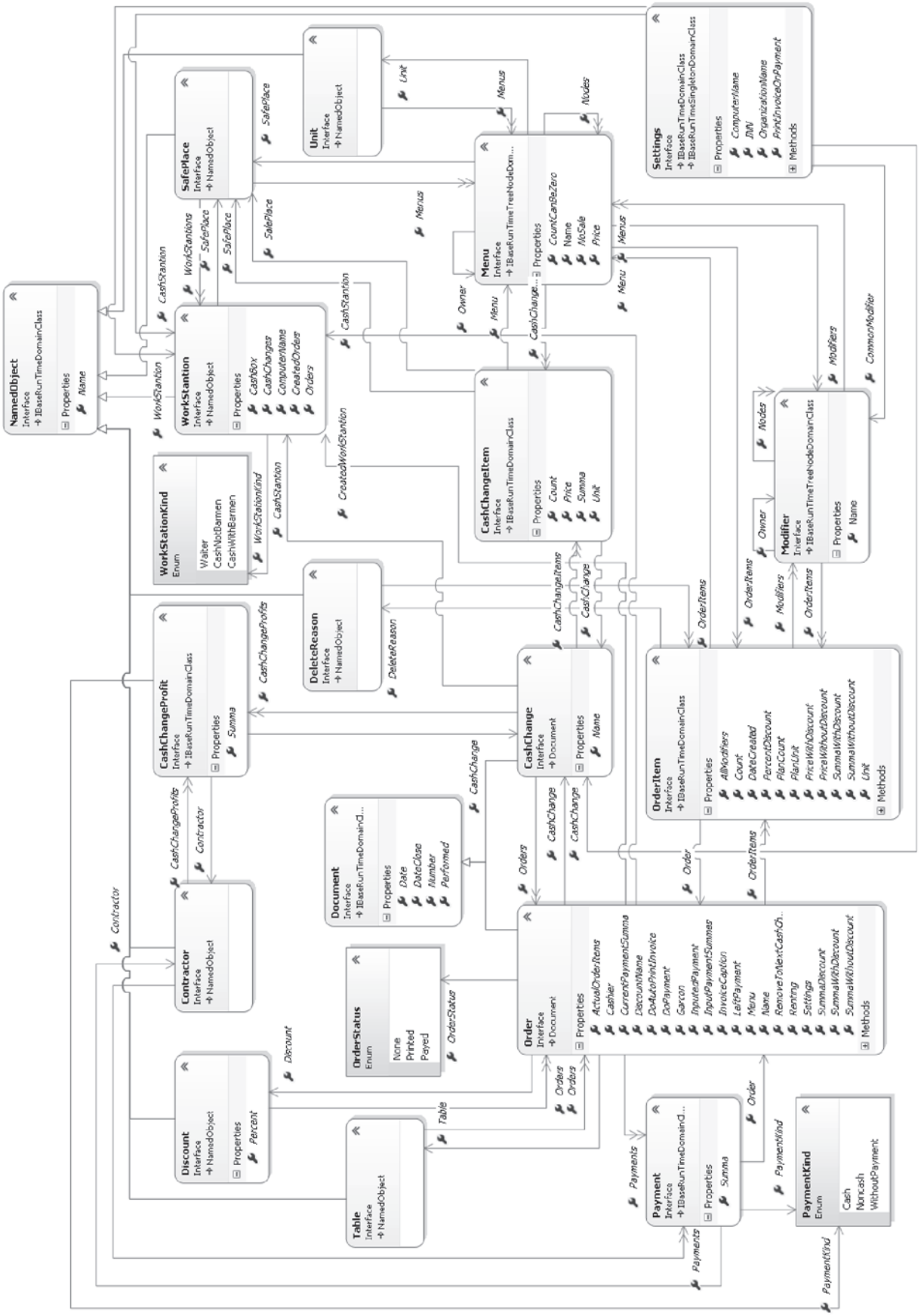


Рис. 3. UML-диаграмма классов тестовой информационной системы массового обслуживания

**Прототипирование графической формы заказа клиентов для сенсорных экранов**

Выше отмечалось, что разработка развитого графического интерфейса с поддержкой сенсорных экранов ($КО_2$) является основным требованием к ИС. Из анализа бизнес-процессов стало ясно, что ключевой является форма заказа клиентов. Именно эта форма, ее состав и структура подробно рассмотрены в данном разделе. При этом уделено внимание привязке графических элементов к классам и атрибутам/методам классов.

На рис. 4 изображен фрагмент модифицированной диаграммы классов, содержащий важные для дальнейшего обсуждения элементы.

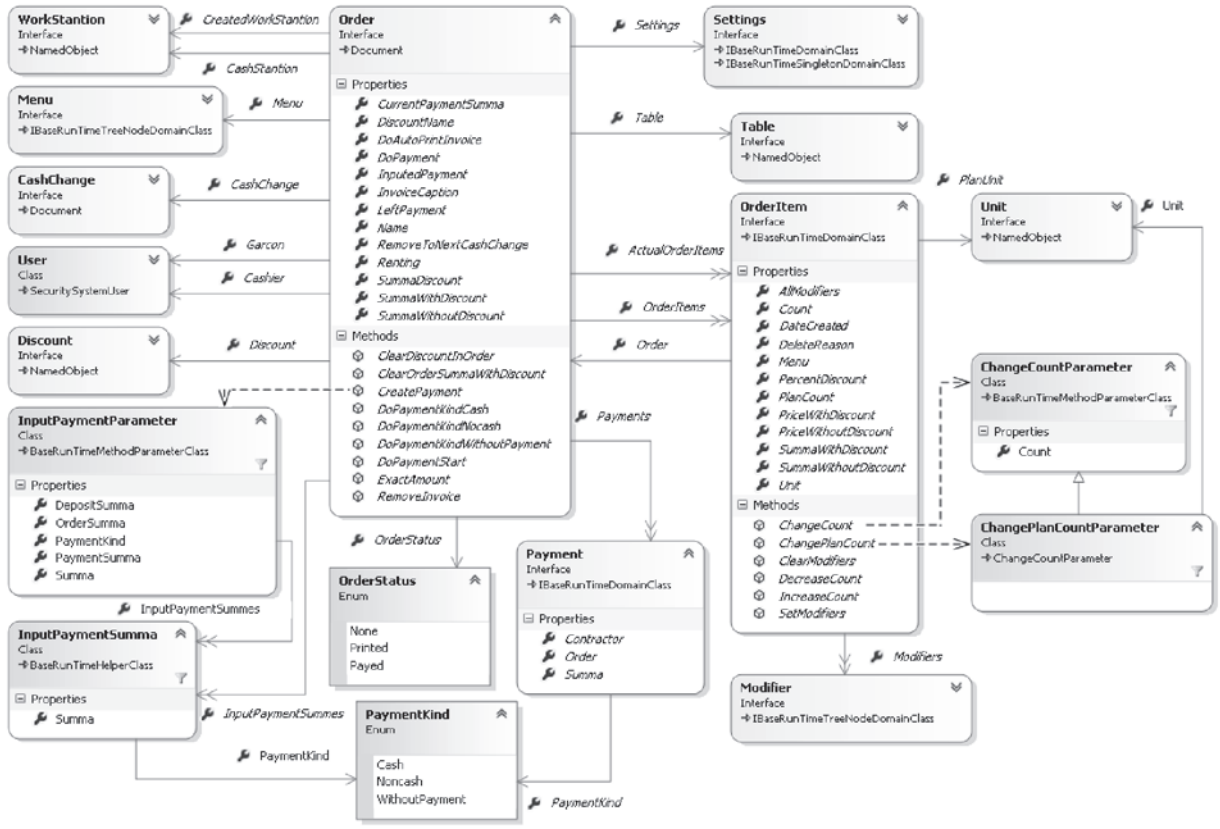


Рис. 4. Фрагмент модифицированной UML-диаграммы классов

При проектировании ИС прежде всего анализируются типы сущностей прикладной предметной области, которые были описаны соответствующими метаклассами представленной ранее метамодели (рис. 2).

1. **Сохраняемые сущности.** Они используются для представления информации, которая физически хранится в БД. Описание подобных сущностей выполняется с помощью создания экземпляров метакласса DomainClass метамодели. При этом для реализации происходит генерация интерфейсов языка C# (см. рис. 4 Interface).
2. **Вспомогательные сущности.** Экземпляры этих сущностей не сохраняются в БД, но их представление в системе необходимо как для упрощения реализации бизнес-логики, так и для отображения вспомогательной информации в интерфейсе пользователя. Описание подобных сущностей выполняется с помощью создания экземпляров метакласса HelperClass метамодели. При этом для реализации происходит генерация классов языка C# (см. рис. 4 Class, связанные отношением ассоциации с использующими интерфейсами).
3. **Классы-параметры методов.** Основная бизнес-логика приложения реализуется в виде методов классов. Этот подход соответствует ОО парадигме. При этом в конечном итоге методы представляются пользователю в виде элементов графического интерфейса, например, в виде кнопок. При этом методы могут принимать набор параметров. В SharpArchitect RAD Studio применяется паттерн проектирования параметр-объект (Parameter Object), предполагающий передачу одного экземпляра классов в виде параметра в метод класса вместо множества атомарных переменных. Описание подобных сущностей выполняется с помощью создания экземпляров метакласса MethodParameterClass метамодели. При этом для реализации выполняется формирование классов языка C# (см. рис. 4 Class, связанные отношением зависимости с использующими методами).

4. **Классы-перечисления/множества.** Эти классы используются в том случае, когда атрибут может принимать определенное значение из описанного ранее набора и этот набор не может расширяться пользователями. Для реализации выполняется генерация перечислений языка C# (см. рис. 4 Enum).

Перед входом в систему пользователю необходимо авторизоваться, что соответствует $КО_3$. В этой связи был выделен класс User, позволяющий сохранить информацию о кассире и официанте, обслуживающим заказ. Постоянным клиентам могут быть предоставлены скидки, что описывается с помощью класса Discount. Для представления платежей по заказу используется класс Payment, а различные типы платежей (наличные, безналичный расчет) описываются перечислением PaymentKind.

С помощью отношений зависимости на представленной UML-диаграмме классов показывают связь между методом класса и классом-параметром метода. Экземпляр класса InputPaymentParameter используется в методе CreatePayment, который позволяет создать оплату заказа. Вспомогательный класс InputPaymentSumma будет использован для упрощения процесса оплаты заказа по частям, т.е. в том случае, когда часть суммы оплачена наличными, а часть – безналичным расчетом.

Для указания различных модификаторов заказа, например, опции «Без лука», используется класс Modifier. Сами модификаторы прописываются в методе SetModifiers класса OrderItem. Для моделирования модификаторов используется модель Entity-Attribute-Value (EAV). При этом ограничения не указываются, что позволяет, например, заказать товар «Молочный коктейль» с модификатором «Без лука». Но такая ситуация не приведет к проблеме, так как в молочном коктейле лук отсутствует по умолчанию. Количество товара указывается в методе ChangeCount и использует параметр типа ChangeCountParameter. Для указания планового количества весового товара, например, шашлыка, используется метод ChangePlanCount и класс ChangePlanCountParameter.

Описанное решение ИС в настоящий момент внедряется и тестируется в нескольких сетях быстрого питания в Ростовской области.

## Заключение

В работе показана применимость предметно-ориентированного проектирования при реализации ИС для предметных областей массового обслуживания клиентов. На основе имеющихся разработок и существующих работ были выделены требования, которым должна соответствовать полученная реализация. Для проверки реализуемости была спроектирована тестовая система. Так как к эргономике интерфейса предъявляются большие требования, то было выполнено прототипирование формы заказа. При этом была предложена собственная графическая нотация. Представлены два подхода к разграничению прав и реализации многопользовательского доступа в системах для предметных областей обслуживания клиентов. Оба подхода были апробированы в реализованной системе.

## References


1. Kharwat A.K. Computer simulation: an important tool in the fast-food industry. *Winter Simulation Conference Proceedings*, 1991, pp. 811–815. doi: 10.1109/WSC.1991.185689
2. Tsuboi H., Takebayashi Y. A real-time task-oriented speech understanding system using keyword-spotting. *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, ICASSP-92*. San Francisco, 1992, vol. 1, pp. 197–200. doi: 10.1109/ICASSP.1992.225938
3. Xue D., Zhao Q., Guo X. TOPSIS method for evaluation customer service satisfaction to fast food industry. *Proc. 2008 IEEE Int. Conf. on Service Operations and Logistics, and Informatics, IEEE/SOLI 2008*. Beijing, China, 2008, pp. 920–925. doi: 10.1109/SOLI.2008.4686530
4. Shimmura T., Takenaka T., Akamatsu M. Real-time process management system in a restaurant by sharing food order information. *Int. Conf. of Soft Computing and Pattern Recognition, SOCPAR'09*. Malacca, 2009, pp. 703–706. doi: 10.1109/SoCPaR.2009.141
5. Yong L.T. Qi C.Y., Yee C.S., Johnson A., Hoong N.K. Designing and developing a PDA food ordering system using interaction design approach: a case study. *Int. Conf. on Computer Technology and Development, ICCTD'09*. Kota Kinabalu, Malasia, 2009, vol. 1, pp. 68–71. doi: 10.1109/ICCTD.2009.18
6. Cheong S.N., Yeong M.H.T., Neoh J.J., Teh C.Y., Yap W.J. Enriching dining experience with the multi-touchable entertainment applications. *Int. Conf. on Science and Social Research, CSSR 2010*. Kuala Lumpur, Malaysia, 2010, pp. 373–378. doi: 10.1109/CSSR.2010.5773803
7. Chen K.J., Lo Y.J., Trappey A.J.C., Trappey C.V. Reengineer restaurant set-meal design process: A combination of modular product design and system engineering evaluation approach. *Int. Conf. on System Science and Engineering, ICSSE 2010*. Taipei, Taiwan, 2010, pp. 587–592. doi: 10.1109/ICSSE.2010.5551781
8. Fujita T., Shimada H., Sato K. Self-ordering system of restaurants for considering allergy information. *IEEE 11th Consumer Communications and Networking Conference, CCNC 2014*. Las Vegas, 2014, pp. 179–184. doi: 10.1109/CCNC.2014.6940502

9. Muslu I., Jakshylykov J., Soorbekova B., Kutmanova U., Musiralieva M. Restaurant process simulation in Kyrgyzstan. *Proc. 11th Int. Conf. on Electronics, Computer and Computation, ICECCO 2014*. Abuja, Nigeria, 2014, art. 6997576. doi: 10.1109/ICECCO.2014.6997576
10. Oleinik P.P., Yuzefova S.Yu., Nikolenko O.I. Opyt proektirovaniya informatsionnoi sistemy dlya restoranov bystrogo pitaniya [Experience in design of information systems for fast-food restaurants]. *Materialy IX Mezhdunarodnoi Nauchno-Prakticheskoi Konferentsii Ob"ektnye sistemy-2014* [Proc. IX Int. Scientific and Practical Conference on Object Systems-2014]. Rostov-on-Don, 2014, pp. 12–16.
11. Oleinik P.P. *Unifitsirovannaya Sreda Bystroi Razrabotki Korporativnykh Informatsionnykh Sistem SharpArchitect RAD Studio*.Certificate of State Registration of Computer Programs, no. 2013618212, 2013.
12. Oleinik P.P. Ierarkhiya klassov metamodeli ob"ektnoi sistemy [Class hierarchy of object system metamodel]. *Materialy VI Mezhdunarodnoi Nauchno-Prakticheskoi Konferentsii Ob"ektnye sistemy-2012* [Proc. VI Int. Scientific and Practical Conference Object Systems-2012]. Rostov-on-Don, 2012, pp. 37–40.
13. Oleinik P.P. Ierarkhiya klassov predstavleniya validatsionnykh pravil ob"ektnoi sistemy [Class hierarchy submission validation rules of object system]. Materialy VII Mezhdunarodnoi Nauchno-Prakticheskoi Konferentsii Ob"ektnye sistemy-2013 [Proc. VII Int. Scientific and Practical Conference Object Systems-2013]. Rostov-on-Don, 2013, pp. 14–17.
14. Oleynik P.P. Using metamodel of object system for domain-driven design the database structure. *Proc.12th IEEE East-West Design and Test Symposium, EWDTS'2014*. Kiev, Ukraine, 2014, art. 7027052. doi: 10.1109/EWDTS.2014.7027052
15. Oleynik P.P. The elements of development environment for information systems based on metamodel of object system. *Business Informatics*, 2013, no. 4(26), pp. 69–76.
16. Oleynik P.P., Kurakov Yu.I. The concept for creation of service corporate information systems of economic industrial energy cluster. *Applied Informatics*, 2014, no. 6 (54), pp. 5–23.
17. Evans E. *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Addison Wesley, 2003, 560 p.

***Олейник Павел Петрович*** – кандидат технических наук, доцент, доцент, Шахтинский институт (филиал) Южно-Российского государственного политехнического университета им. М.И. Платова, Шахты, 346500, Российская Федерация, xsl@list.ru

***Pavel P. Oleynik*** – PhD, Associate Professor, Associate Professor, Shakhty Institute (Branch) of Platov South Russian State Polytechnic University (NPI), Shakhty, 346500, Russian Federation, xsl@list.ru