УДК 004.942
DOI: 10.26102/2310-6018/2023.42.3.018

Высокоуровневая структура модулей для построения специальных систем автоматизированного проектирования

А.С. Троценко[1]✉, А.А. Успехов[2], М.И. Чижов[1]

[1]Воронежский государственный технический университет, Воронеж, Российская Федерация
[2]ООО «ИНОБИТЕК», Воронеж, Российская Федерация

Резюме. В современном производстве существует потребность в проектировании специализированных изделий, заданных определенным набором изменяющихся параметров. Повторное проектирование изделия, связанное с изменением некоторых из этих параметров, становится одной из задач инженера. Использование тяжелых систем автоматизированного проектирования в таких случаях может приводить к значительному повышению трудозатрат. Создание истории построений твердотельной модели изделия, сбалансированной относительно заданного набора ее параметров, оказывает существенное влияние на общую сложность процесса проектирования. Повысить эффективность этого процесса позволяет применение специальных систем автоматизированного проектирования, направленных на создание параметризированной модели определенного изделия. В данной работе представлена структура высокоуровневых модулей, обеспечивающих быструю разработку специальных систем автоматизированного проектирования. Одним из методов, обеспечивающих быструю разработку, является сокращение большого объема знаний классов и методов используемого геометрического ядра. Наличие отдельных функциональных блоков позволяет строить системы твердотельного моделирования различной наполненности: от простых линейных систем до систем с расширенными возможностями моделирования, анализа и импорта/экспорта данных. Для снижения зависимости разрабатываемых систем от конкретных геометрических ядер предложенная структура обеспечивает сокрытие используемого геометрического ядра с помощью шаблона проектирования закрытой реализации.

Ключевые слова: архитектура, твердотельное моделирование, системы автоматизированного проектирования, шаблоны проектирования, сокрытие реализации.

Для цитирования: Троценко А.С., Успехов А.А., Чижов М.И. Высокоуровневая структура модулей для построения специальных систем автоматизированного проектирования. *Моделирование, оптимизация и информационные технологии.* 2023;11(3). URL: https://moitvivt.ru/ru/journal/pdf?id=1430 DOI: 10.26102/2310-6018/2023.42.3.018

The high-level structure of modules for building special computer-aided design systems

A.S. Trotsenko[1]✉, A.A. Uspehov[2], M.I. Chizhov[1]

[1]Voronezh State Technical University, Voronezh, the Russian Federation
[2]INOBITEC LLC, Voronezh, the Russian Federation

Abstract. In modern production, there is a need to design specialized products predetermined by a certain set of changing parameters. Re-designing of a product associated with adjusting some of these parameters becomes one of the tasks for an engineer to complete. Using of heavy computer-aided design systems in such cases can lead to a significant increase in labor costs. Creating a history of building a solid model of a product balanced according to a given set of its parameters has a significant impact on the overall complexity of the design process. Increasing the efficiency of this process allows the use of special computer-aided design systems aimed at creating a parameterized model of a particular product.

This paper presents the structure of high-level modules that ensures the rapid development of special computer-aided design systems. One of the methods that provide rapid development is the reduction of a large amount of knowledge of the classes and methods of the geometric core being used. The presence of separate functional blocks helps to build various solid-state modeling systems: from simple linear systems to systems with advanced modeling, analysis and data import/export capabilities. To reduce the dependency of the developed systems on a specific geometric core, the high-level structure that is being proposed provides the hiding of the geometric core being used by means of the private implementation design pattern.

Keywords: architecture, solid modeling, computer-aided design systems, design patterns, pointer to implementation.

For citation: Trotsenko A.S., Uspehov A.A., Chizhov M.I. The high-level structure of modules for building special computer-aided design systems. *Modeling, Optimization and Information Technology*. 2023;11(3). URL: https://moitvivt.ru/ru/journal/pdf?id=1430 DOI: 10.26102/2310-6018/2023.42.3.018 (In Russ.).

Введение

Применение специализированных программных средств в области систем автоматизированного проектирования (САПР) оказывается востребованным на производстве и при проведении исследований. Такие программные средства позволяют оперативно создавать модели различных конфигураций определенного типа изделия. Однако при их разработке существует проблема, основанная на большом объеме знаний классов и методов. Например, популярное сегодня открытое ядро геометрического моделирования OpenCasCade [1] включает несколько тысяч классов, распределенных в более чем 40 модулях. Другая проблема – это отсутствие необходимой функциональности в выбранном геометрическом ядре. Так, в ядре OpenCasCade отсутствует решатель геометрических ограничений для двумерных и трехмерных моделей, хотя некоторые коммерческие ядра такой функциональностью обладают [2]. Вариантом решения проблемы является создание цифровой платформы готовых алгоритмов реализации базовых модулей системы твердотельного моделирования, которые обеспечили бы сокращение трудоемкости разработки самостоятельных приложений для проектирования изделий.

Структура системы

Предлагаемая в работе архитектура ПО построена на основе модели C4 [3]. Модель C4 позволяет проектировать систему на четырех уровнях детализации, раскрывая на каждом уровне ее отдельные особенности.

На 1-м уровне модели рассмотрим архитектуру в разрезе систем и пользователей этих систем (Рисунок 1). Пользователями системы выступают инженеры или студенты инженерных специальностей, деятельность которых сопряжена с твердотельным моделированием. В качестве потенциальных сценариев использования специальных САПР рассматриваются как построение единичных твердотельных моделей с требуемыми параметрами, так и создание целых серий моделей с параметрами, заданными в пределах определенного допуска. Серии идентичных моделей становятся необходимыми при решении задач инженерного анализа, связанных с поиском наиболее оптимальной конфигурации изделия.

Предлагаемая структура модулей САПР размещается внутри системы, названной EduCad. Система EduCad предлагает API (Application Programming Interface) [4] как для языка программирования C++, так и для языка Python, что расширяет возможные сценарии использования системы. Если C++ оказывается эффективным при разработке

конечных производительных пользовательских приложений, то Python хорошо себя зарекомендовал при решении исследовательских задач. При этом оба языка программирования обладают широким набором библиотек различной направленности, которые позволяют разрабатывать приложения разного типа: серверные приложения (HTTP Server) и приложения рабочего стола (Desktop).

Преимущества desktop-приложений перед серверными приложениями заключаются в относительной простоте разработки, так как серверная реализация включает дополнительную систему веб-приложения (Web), которая взаимодействует с сервером посредством REST API (Representational State Transfer API) [5]. Проектирование REST API, равно как и C++/Python API, требует определенных навыков разработчика, что также повышает общую сложность системы. С другой стороны, разработанное однажды веб-приложение может быть запущено практически на любом устройстве, обладающем веб-браузером: от настольного компьютера до мобильного устройства (смартфон, планшет и т. п.). При этом desktop-приложения требуют конфигурации и перекомпиляции под каждую отдельную операционную (ОС) систему. Возможности компиляции desktop-приложений под мобильные ОС нет, но существуют практики ограниченного портирования функциональности в мобильные приложения на языке C++.

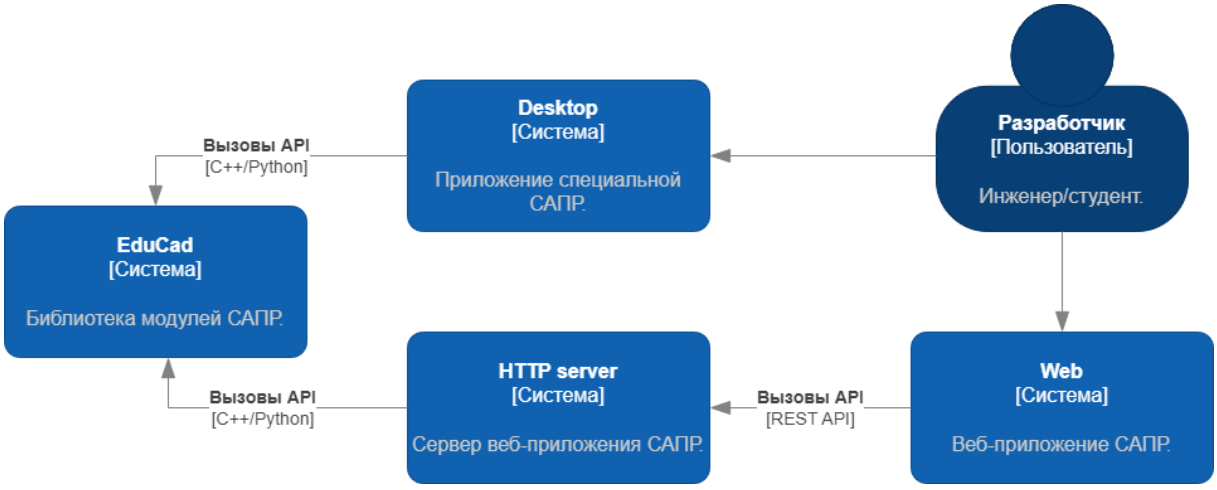

Рисунок 1 – Структура системы (первый уровень модели C4)
Figure 1 – System structure (the first level of a C4 model)

Контейнеры системы

На следующем уровне модели проектирования определены контейнеры системы EduCad и их функции (Рисунок 2). Рассмотрим их по-отдельности.

Контейнер Core состоит из набора компонентов, обеспечивающих основную функциональность твердотельного моделирования: геометрические и топологические объекты моделирования, их обмен между другими системами и визуализацию. Более подробно компоненты рассмотрены в следующем разделе.

Контейнер Project организует сущность «Проект», включающую в себя историю построений в виде последовательности команд с заданными параметрами построения. Каждая команда принимает на вход параметры и/или целевой объект (геометрия или твердотельная модель), а на выходе либо возвращает преобразованный объект, либо отправляет результат во внешний ресурс. Этот подход аналогичен схеме фильтров, используемой в пакете VTK (Visualization Tool Kit) [6]. Главное отличие от схемы VTK заключается в использовании параметров как объектов, изменение состояния которых

обязательно приводит к перерасчету истории построений. По умолчанию контейнер Project предоставляет функциональность для трансляции последовательности команд построения в Json-файл и обратно. Опционально может быть реализована возможность конвертации проекта в специальную единицу трансляции, которая позволяет переносить историю построений между программными средствами САПР различных производителей [7].

Контейнер Modules включает статические и динамические подключаемые компоненты, управление которыми позволит создавать приложения различной функциональной наполненности с помощью комбинации элементарных функциональных блоков. Подробнее о компонентах контейнера рассказано в следующем разделе.

Контейнер «Desktop builder» обеспечивает функциональность для блочного построения приложения рабочего стола (desktop) из подключаемых компонентов. В основе контейнера предлагается использовать шаблон проектирования «Строитель» [8], где каждый строительный метод отражает элементарный функциональный блок из контейнера Modules. Помимо работы с элементарными функциональными блоками, контейнер позволяет строить приложения с заранее заданными предустановками в зависимости от прикладного назначения разрабатываемой САПР.

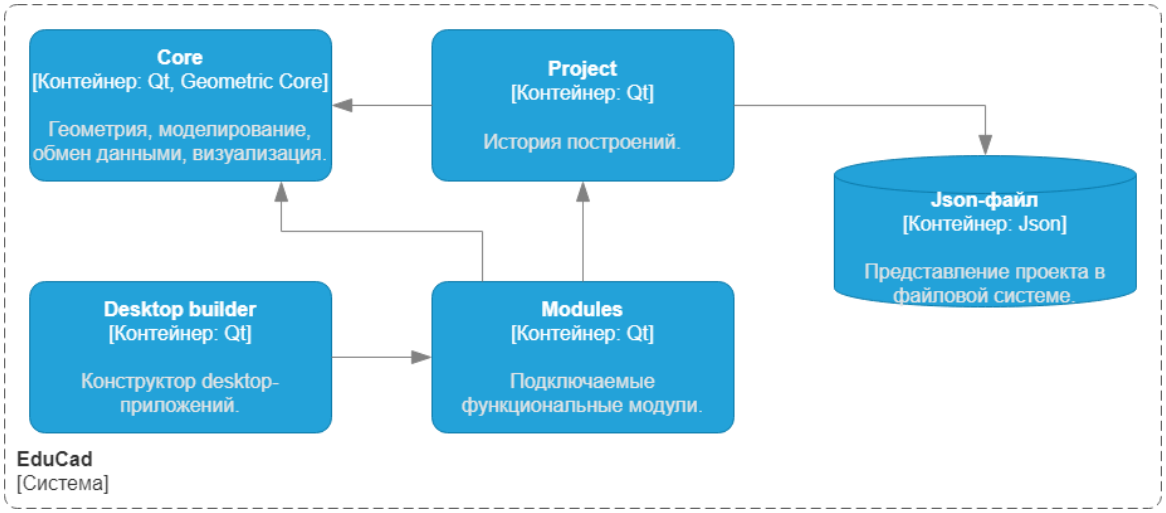

Рисунок 2 – Структура контейнеров системы (второй уровень модели C4)
Figure 2 – Structure of the system containers (the second level of a C4 model)

Каждый контейнер реализуется как динамическая библиотека с собственным открытым API. Это может позволить использовать некоторые контейнеры независимо на разных уровнях абстракций приложения. Так, контейнер Core может быть использован для реализации простых САПР твердотельного моделирования, не требующих сохранения истории построений. Так как Core предоставляет блоки на самом низком уровне абстракции, то пользовательская реализация будет выглядеть достаточно объемной. Включение в приложение контейнера Project открывает возможности для создания параметризированной истории построений и ее сохранения в поддерживаемые файловые форматы. Если необходимы составные готовые компоненты интерактивности приложения, то тогда задействуется группа компонентов контейнера Modules. И, наконец, использование функциональности контейнера «Desktop builder» позволит разработчику быстро скомпоновать desktop-приложение необходимой конфигурации, полностью сосредоточившись на создании параметризированной твердотельной модели.

Структура компонентов системы

Детализация контейнеров системы представлена на структурной схеме третьего уровня модели C4 (Рисунок 3). В общем случае компонент реализуется как объектная, статическая или динамическая библиотека классов. Объектные и статические библиотеки могут быть объединены в динамические.

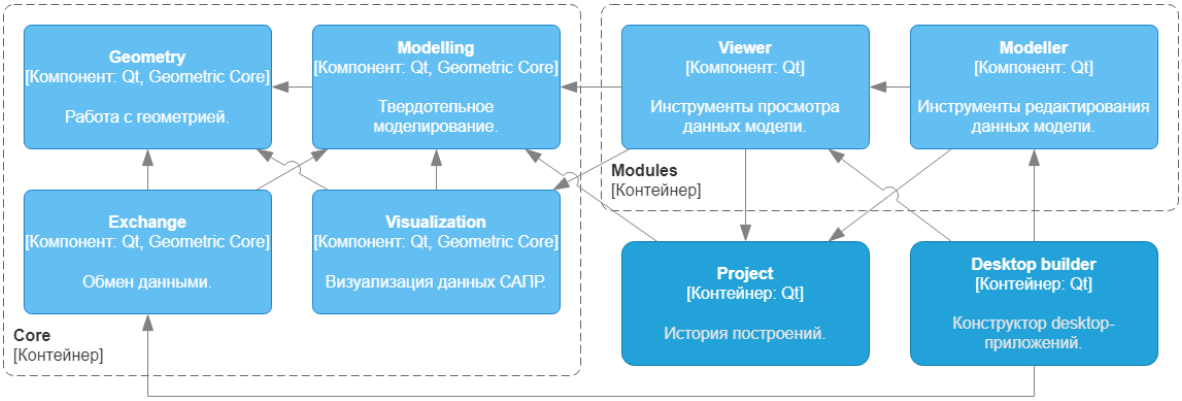

Рисунок 3 – Структура компонентов системы (третий уровень модели C4)
Figure 3 – Structure of the system components (the third level of a C4 model)

Компонент «Core: Geometry» предлагает сущности и алгоритмы для работы с двумерной и трехмерной геометрией. Этот компонент по своей сути является высокоуровневой оберткой над геометрическим ядром.

Компонент «Core: Modelling» предлагает сущности и алгоритмы для твердотельного моделирования. Компонент состоит в зависимости от компонента Geometry, дополняя геометрию топологической информацией. Является высокоуровневой оберткой над геометрическим ядром.

Компонент «Core: Exchange» предлагает алгоритмы для импорта и экспорта геометрических и топологических объектов. Поддержка популярных свободных форматов, таких как STEP, IGES, OBJ, STL и других, позволит использовать созданные твердотельные модели во внешних системах. Является высокоуровневой оберткой над геометрическим ядром.

Компонент «Core: Visualization» предлагает набор представлений для визуализации геометрических и топологических данных. Является высокоуровневой оберткой над геометрическим ядром.

Компонент «Modules: Viewer» предлагает элементарные функциональные блоки для просмотра данных твердотельной модели и проекта. Это списки, таблицы, древовидные представления истории построений, двумерные и трехмерные представления геометрии. Также модуль включает функциональные блоки импорта и экспорта данных.

Компонент «Modules: Modeller» предлагает элементарные функциональные блоки для редактирования данных твердотельной модели. Включает функциональность создания эскизов и выполнения отдельных операций твердотельного моделирования.

Обозначенные на Рисунке 3 контейнеры Project и Desktop builder не содержат компонентов третьего уровня.

Сокрытие реализации

Четвертый уровень модели C4 как правило включает в себя схемы классов и алгоритмов, входящих в компоненты третьего уровня. При определении

высокоуровневой структуры, такая детализация представляется избыточной. Однако следует закрыть вопрос, связанный с сокрытием реализации компонентов контейнера Core.

Сокрытие реализации необходимо по следующим причинам:

– инкапсуляция сложности геометрического ядра;

– устранение зависимости пользовательского кода от конкретного геометрического ядра;

– возможность использовать разные геометрические ядра или их композиции в рамках одной системы.

Идея подхода к сокрытию реализации заключается в создании классов-оберток для необходимой функциональности геометрического ядра на основе шаблона проектирования PIMPL (Pointer to IMPLementation) [9]. Подход позволит скрыть классы геометрического ядра внутри компонентов системы и обеспечить возможность подмены выбранного ядра на другое ядро с аналогичной функциональностью. Безусловно, это усложняет процесс разработки, но в то же время обеспечивает известную гибкость.

Общая структурная схема классов паттерна PIMPL показана на Рисунке 4. При разработке системы важно, чтобы элементы используемого геометрического ядра не оказывались в области открытого API (Public API).

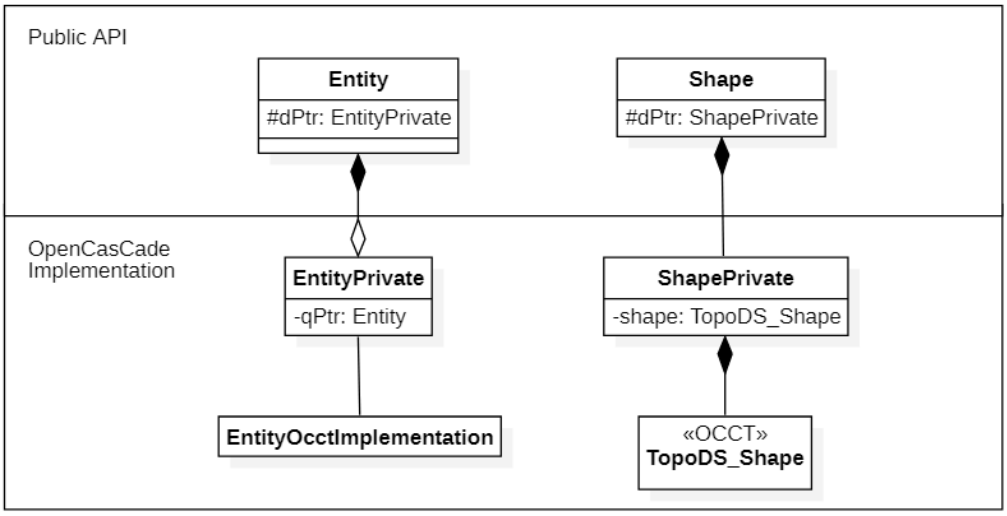

Рисунок 4 – Структурная схема классов сокрытия реализации выбранного геометрического ядра (слева представлена обобщенная абстрактная схема, а справа – пример конкретной сущности Shape)

Figure 4 – Structural diagram of private implementation classes of the selected geometric core (on the left is a generalized abstract diagram, and on the right is an example of a specific Shape entity)

В PIMPL существуют понятия публичных и приватных (закрытых) классов. Приватные классы предлагается именовать с суффиксом «Private». Только этот тип классов может включать в себя объекты геометрического ядра. В общем случае приватный класс определяется как абстракция, реализуемая выбранной функциональностью.

Взаимодействие между приватным и публичным классами одной сущности обеспечивается с помощью d-указателя и q-указателя. Объект d-указателя – это указатель на скрытую реализацию функциональности публичного класса. Размещается в публичном классе. Объект q-указателя – это указатель на публичный класс внутри приватного класса. Однако q-указатель требуется довольно редко. Связь между

приватным классом и реализацией на геометрическом ядре может быть как ассоциацией, так и агрегацией.

Для решения проблемы конвертации приватных объектов в их публичные представления и обратно предлагается использовать конвертирующие методы ToImpl() и FromImpl(). Рассмотрим подход на примере класса Shape (Рисунок 5). Класс ShapeBuilder предоставляет абстракцию для создания экземпляров Shape. Класс ShapeBuilderPrivate, как и ShapePrivate, является дружественным классом для класса Shape. Это ключевой момент, который позволяет классу-строителю изменять реализацию Shape. Классы ShapeBuilder и ShapeBuilderPrivate – абстрактные. Конкретные строители реализуются в классах-наследниках. Например, для создания модели сферы реализован класс SphereShapeBuilder. Использование методов ToImpl() и FromImpl() может быть затруднительным при комбинации геометрических ядер. Тогда закрытый класс любой сущности должен отделяться от его реализаций интерфейсом-адаптером [10].

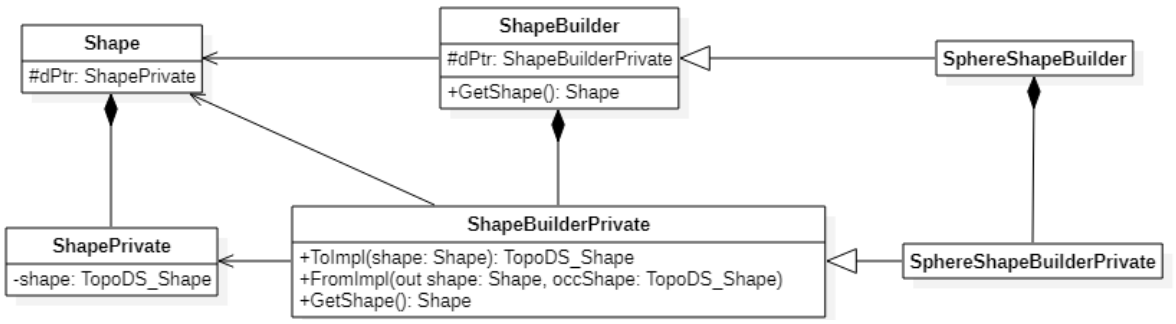

Рисунок 5 – Структурная схема классов сущности Shape с демонстрацией методов конвертации открытых объектов в закрытые и обратно

Figure 5 – Structural diagram of the Shape entity classes with methods for converting public objects into private ones and vice versa

Заключение

В работе предложена высокоуровневая структура модулей, направленных на быстрое создание специальных систем автоматизированного проектирования для параметрического твердотельного моделирования. Применение подхода к сокрытию реализации позволяет использовать внутри структуры как разные геометрические ядра, так и их комбинации.

В перспективе требуется детализировать контейнеры, обеспечивающие функциональность управления историей построения модели и конструирование приложения рабочего стола. Для апробации разработанных структур подразумевается реализация функциональных прототипов специализированных систем автоматизированного проектирования для решения актуальных прикладных конструкторских задач.

СПИСОК ИСТОЧНИКОВ

1. Russell J., Cohn R. *Open Cascade Technology*. Book on Demand Ltd; 2012. 140 p.
2. Чагина А.В., Большаков В.П. *3D моделирование в КОМПАС-3D версий v17 и выше*. СПб: Питер; 2021. 256 с.
3. Vázquez-Ingelmo A., García-Holgado A., García-Peñalvo F.J. C4 model in a Software Engineering subject to ease the comprehension of UML and the software. *IEEE Global*

Engineering Education Conference (EDUCON), Porto, Portugal; 2020. p. 919–924, DOI: 10.1109/EDUCON45650.2020.9125335.
4. Preibisch S. *API Development. A practical guide for business implementation success.* Canada, CA Press; 2018. 178 p.
5. Masse M. *REST API Design Rulebook.* O'Reilly Media; 2012. 114 p.
6. Schroeder W., Martin K., Lorensen B. *Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*. Kitware; 2018. 557 p.
7. Болотцев Д.А., Чижов М.И., Успехов А.А., Чувенкова Т.О. Перенос параметрической модели между САПР. *Новейшие научные достижения: сб. тр. XI междунар. науч. конф.* 2015;14:45–48.
8. Фримен Э., Сьерра К., Бейтс Б. *Паттерны проектирования.* СПб: Питер; 2011. 656 с.
9. Bancila M. *Modern C++ Programming Cookbook*. Packt Publishing Ltd; 2017. 583 p.
10. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. *Приемы объектно-ориентированного проектирования. Паттерны проектирования.* СПб: Питер; 2015. 368 с.

REFERENCES

1. Russell J., Cohn R. *Open Cascade Technology*. Book on Demand Ltd; 2012. 140 p.
2. Chagina A.V., Bolshakov V.P. *3D Modeling in KOMPAS-3D (version v17 and up)*. Saint Petersburg, Piter; 2021. 256 p. (In Russ.).
3. Vázquez-Ingelmo A., García-Holgado A., García-Peñalvo F.J. C4 model in a Software Engineering subject to ease the comprehension of UML and the software. *IEEE Global Engineering Education Conference (EDUCON), Porto, Portugal*; 2020. p. 919–924, DOI: 10.1109/EDUCON45650.2020.9125335.
4. Preibisch S. *API Development. A practical guide for business implementation success.* Canada, CA Press; 2018. 178 p.
5. Masse M. *REST API Design Rulebook.* O'Reilly Media; 2012. 114 p.
6. Schroeder W., Martin K., Lorensen B. *Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*. Kitware; 2018. 557 p.
7. Bolotsev D.A., Chizhov M.I., Uspehov A.A., Chuvenkova T.O. Transferring a parametric model between CAD. *Noveyshie nauchniye dostizheniya: sb. tr. XI mezhdunar. nauchn. conf.* 2015;14:45–48. (In Russ.).
8. Frimen E., Sierra K., Beits B. *Design Patterns.* Saint Petersburg, Piter; 2011. 656 p. (In Russ.).
9. Bancila M. *Modern C++ Programming Cookbook*. Packt Publishing Ltd; 2017. 583 p.
10. Gamma E., Helm R., Jonson R., Vlissides J. *Methods of Object-Oriented Design. Design Patterns.* Saint Petersburg, Piter; 2015. 368 p. (In Russ.).

ИНФОРМАЦИЯ ОБ АВТОРАХ / INFORMATION ABOUT THE AUTHORS

Троценко Александр Сергеевич, кандидат технических наук, ассистент Воронежского государственного технического университета, Воронеж, Российская Федерация.
e-mail: trotsenko93@mail.ru

Aleksandr Sergeevich Trotsenko, Candidate of Technical Sciences, Assistant Lecturer at Voronezh State Technical University, Voronezh, the Russian Federation.

Успехов Андрей Александрович, генеральный директор ООО ИНОБИТЕК, Воронеж, Российская Федерация.
e-mail: auspehov@inobitec.com

Andrei Aleksandrovich Uspehov, CEO of Inobitec LLC, Voronezh, the Russian Federation.

Чижов Михаил Иванович; доктор технических наук, профессор Воронежского государственного технического университета, заведующий кафедрой Компьютерных интеллектуальных технологий проектирования, Воронеж, Российская Федерация.
e-mail: mihailc@list.ru

Mikhail Ivanovich Chizhov, Doctor of Technical Sciences, Professor at Voronezh State Technical University, Head of the Department of Computer Intellectual Technologies of Design, Voronezh, the Russian Federation.

Статья поступила в редакцию 25.07.2023; одобрена после рецензирования 01.09.2023; принята к публикации 13.09.2023.

The article was submitted 25.07.2023; approved after reviewing 01.09.2023; accepted for publication 13.09.2023.