ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Издается с сентября 2010 г. DOI 10.17587/issn.2220-3397 ISSN 2220-3397

Журнал издается при поддержке Отделения математических наук РАН, Отделения нанотехнологий и информационных технологий РАН, МГУ имени М.В. Ломоносова, МГТУ имени Н.Э. Баумана

СОДЕРЖАНИЕ

Хританков А. С. Линейки программных продуктов: современное состояние и стандарты 387

Сухов А. О. Предметно-ориентированный язык описания трансформаций визуальных моделей 396

Светушков Н. Н. Программные механизмы для моделирования процессов теплопередачи в геометрически сложных изделиях 407

Курако Е. А., Орлов В. Л. Сервис-браузеры для информационных систем 413

Астапов И. С., **Астапов Н. С.** Алгоритмы символьного решения алгебраических уравнений 422

Журнал зарегистрирован в Федеральной службе по надзору в сфере связи, информационных технологий и массовых коммуникаций. Свидетельство о регистрации ПИ № ФС77-38590 от 24 декабря 2009 г.

Журнал распространяется по подписке, которую можно оформить в любом почтовом отделении (индексы: по каталогу агентства "Роспечать" — **22765**, по Объединенному каталогу "Пресса России" — **39795**) или непосредственно в редакции.
Тел.: (499) 269-53-97. Факс: (499) 269-55-10.
Http://novtex.ru/prin/rus E-mail: prin@novtex.ru
Журнал включен в систему Российского индекса научного цитирования.
Журнал входит в Перечень научных журналов, в которых по рекомендации ВАК РФ должны быть опубликованы научные результаты диссертаций на соискание ученой степени доктора и кандидата наук.

THEORETICAL AND APPLIED SCIENTIFIC AND TECHNICAL JOURNAL

SOFTWARE ENGINEERING

PROGRAMMNAYA INGENERIA

Vol. 8
N 9
2017

Published since September 2010 **DOI 10.17587/issn.2220-3397** **ISSN 2220-3397**

CONTENTS

Khritankov A. S. Product Line Engineering: Standards and State-of-Practice 387

Sukhov A. O. Domain-Specific Language for Visual Models Transformation Creation 396

Svetushkov N. N. Software Facilities for Heat Transfer Modeling in Geometrically Complex Objects 407

Kurako E. A., Orlov V. L. Service Browsers for Information Systems 413

Astapov I. S., Astapov N. S. Algorithms for Symbolic Solving of Algebraic Equations 422

Information about the jornal is available online at:
http://novtex.ru/prin/eng e-mail:prin@novtex.ru

УДК 004.41 DOI: 10.17587/prin.8.387-395

А. С. Хританков, канд. физ.-мат. наук, доц., e-mail: anton.khritankov@acm.org,
Московский физико-технический институт, Москва

Линейки программных продуктов: современное состояние и стандарты

В июне 2017 г. вступил в силу стандарт ГОСТ Р ИСО/МЭК 26555, описывающий процессы разработки линеек программных и программно-аппаратных продуктов. В данной статье дан краткий обзор этого стандарта и освещены текущее состояние в области разработки линеек продуктов в мире, а также основные задачи и направления исследований, приведены примеры применения методов разработки линеек продуктов.

Ключевые слова: *индустриальный интернет, интернет вещей (IoT), линейки продуктов, ГОСТ Р ИСО/МЭК 26550, ЛППС*

Введение

Современное состояние в области разработки программного обеспечения и программно-аппаратных комплексов определяется переходом к так называемой парадигме "Индустрии 4.0" [1, 2], которая подразумевает гибкость и массовую настраиваемость продуктов под нужды заказчиков и пользователей в соответствии с их требованиями.

Методы и программные средства разработки линеек программных продуктов и систем (ЛППС) позволяют компаниям-разработчикам эффективно создавать и развивать семейства программно-аппаратных комплексов в рамках одной предметной области, используя модельно-ориентированные методы управления повторным использованием между продуктами семейства. Некоторые методы управления повторным использованием с помощью моделей изложены в настоящей статье.

Используемый далее в контексте статьи термин "продукт" обозначает программный или программно-аппаратный комплекс, который предоставляется на рынке потребителям, как правило, с фирменным обозначением. В данной статье термин "система" употребляется для обозначения программных или программно-аппаратных комплексов в случаях, когда это не приводит к противоречиям.

Принятый в 2016 г. и вступающий в действие стандарт ГОСТ Р ИСО/МЭК 26555 [3] описывает основные понятия, процессы организационного и технического управления разработкой линеек продуктов, взаимосвязи с другими процессами жизненного цикла, в том числе — определяемыми стандартами ГОСТ Р ИСО/МЭК 12207 [4], ГОСТ Р ИСО/МЭК 15288 [5].

Область разработки линеек продуктов развивается в течение уже более тридцати лет [6—8], и ее методы применяются в IT-индустрии, об этом речь пойдет далее. Тем не менее в отечественных научно-технических журналах практически отсутствуют публикации, посвященные этой теме. В работе [9] проведено сравнение нескольких программных средств, поддерживающих разработку линеек продуктов. В статье [10] представлен краткий обзор методологии продуктовых линеек и фабрик программного обеспечения. В работе [11] авторы рассматривают возможность применения методов моделирования изменчивости и языков представления изменчивости в исходном коде для описания сложных операционных систем, адаптируемых к разным конфигурациям аппаратного обеспечения.

В данной статье предпринята попытка сориентировать читателя в области разработки линеек программных продуктов и систем, а также дать обзор вступающего в действие стандарта ГОСТ Р ИСО/МЭК 26555.

Замечание по терминологии. В стандарте ГОСТ Р ИСО/МЭК 26555 английские термины, по мнению автора, переведены не всегда удачно. Поэтому далее в статье наряду с обозначением термина, который используется в стандарте, будут приведены и другие варианты перевода.

Моделирование изменчивости между продуктами линейки

Укажем области применения и обозначим основные понятия, используемые при разработке линеек продуктов. Рассмотрим несколько программных продуктов, разрабатываемых одной организацией и ориентированных на решение сходных задач в общей для них предметной области. Сходство решаемых задач определяется возможностью выделения общих функциональных частей между программными системами, реализующими эти продукты. Эффективная организация процессов совместной разработки этих систем предполагает повторное использование их общих частей.

Линейкой программных продуктов и систем [6] называют семейство продуктов, обладающих общим,

управляемым набором функций (фич), которые удовлетворяют определенные потребности выбранного рынка или имеют собственную миссию, разрабатываются из общего набора базовых активов заранее определенным образом.

При компонентно-ориентированном или сервис-ориентированном подходе к совместной разработке систем и продуктов на их основе решения по выделению и применению повторно используемых частей принимаются в соответствии с архитектурой этих систем и технологическими ограничениями. В линейках программных продуктов решение о том, что будет повторно использовано и в каких системах, принимается в соответствии с предполагаемым применением продуктов, реализуемых этими системами, с помощью модели изменчивости (модели вариабельности [3]).

Модель изменчивости является результатом анализа предметной области в целях выявления общих частей между продуктами, которые могли бы существовать в этой предметной области. Модель изменчивости может быть выражена в виде *модели черт (фич)* продуктов (*feature model*), либо в виде *модели альтернатив* (*decision model*). *Конфигурация продукта* (*configuration/variant*) является результатом выбора из представленных в модели альтернатив или черт. *Базовые активы* (*core assets*) линейки включают предназначенные для повторного использования общие между продуктами программные модули, проектную и сопровождающую продукт документацию, планы и сценарии тестирования, их реализацию в тестах и другие артефакты и рабочие продукты. *Эталонная архитектура линейки* (*reference architecture*) определяет общие для всех продуктов линейки решения по созданию систем путем сборки из активов согласно конфигурации, по сопровождению совместно с моделью изменчивости, по использованию продуктов на протяжении их времени жизни (жизненного цикла).

Модель черт [12] обозначает повторно используемые функциональные части или модули системы, отмечает ограничения на совместное включение этих частей в конфигурацию продукта. Современные модели черт позволяют добавлять чертам атрибуты с изменяемыми при выборе отдельной черты значениями (атрибутивные модели черт), а также включать несколько экземпляров черты в конфигурацию продукта (модели черт с кратностью). Пример модели черт приведен на рис. 1.

Модель альтернатив [14] перечисляет решения по выбору одной их представленных альтернатив,

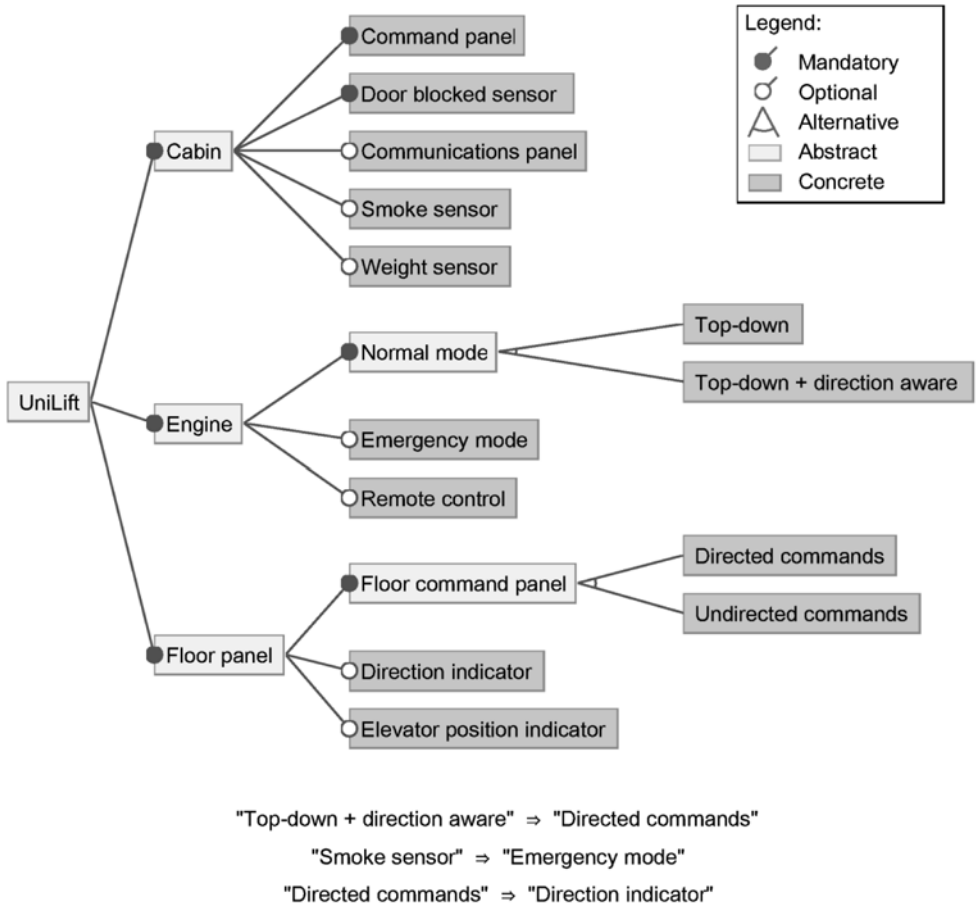

Рис. 1. Иллюстративный пример модели изменчивости для линейки лифтов. Черты обозначены прямоугольниками, логические ограничения показаны линиями с различными окончаниями (adornments). Диаграмма построена с помощью программного средства FeatureIDE [13]

которые необходимо принять для составления конфигурации продукта. Модель также включает ограничения на принимаемые решения и зависимости между альтернативами.

Применение продуктовых линеек

Прямым применением ЛППС является совместная разработка семейств продуктов в рамках одной предметной области. В этом случае конфигурации продуктов существуют одновременно и реализованы в различных системах семейства или линейки. Управление изменчивостью и сходством между продуктами используется для снижения стоимости, повышения качества и выстраивания процессов разработки новых и совершенствования существующих продуктов.

Другое применение линеек программных продуктов — разработка больших программных систем с продолжительным жизненным циклом (совместно со стандартом по системной инженерии ГОСТ Р ИСО/МЭК 15288). В этом случае конфигурации разделены во времени. Более ранние версии системы рассматриваются как реализации продуктов с определенными конфигурациями. Изменчивость между продуктами используется для управления развитием системы и совместимостью между версиями систем — разными продуктами в линейке.

Во многих случаях линейки программных продуктов применяют для управления конфигурациями, как разделенными во времени, так и одновременно существующими. Такой подход применим для разработки семейств программных систем с продолжительным жизненным циклом [6].

Обзор стандарта продуктовых линеек

Международные стандарты. Следует отметить, что во время написания настоящей статьи разработка серии международных стандартов ISO/IEC 26550—26599 в области линеек программных и программно-аппаратных систем продолжается. На рис. 2 представлена диаграмма взаимосвязей между принятыми и запланированными стандартами в редакции от 2015 г. На настоящее время серия включает следующие опубликованные стандарты:

- ISO/IEC 26550:2015 — определяет эталонную модель для реализации линеек продуктов, устанавливает рамки остальных стандартов серии;
- ISO/IEC 26551:2016 — описывает методы и инструменты разработки и управления требованиями в линейке продуктов;
- ISO/IEC 26555:2015 — определяет методы и инструментальные средства технического управления линейкой продуктов, такие как управление изменчивостью, управление базовыми активами и процессами разработки и управления линейки;
- ISO/IEC 26557:2016 — определяет механизмы реализации изменчивости в линейках продуктов.

На стадии утверждения находятся стандарты по процессам и возможностям методов и программных средств по моделированию изменчивости (ISO/IEC 26558), по поддержанию соответствия в линейках продуктов (ISO/IEC 26559) между моделью изменчивости и базовыми активами.

В процессе разработки стандарты по архитектурному проектированию для линеек продуктов (ISO/IEC 26552); по реализации базовых активов (приложений) (ISO/IEC 26553); по верификации и проверке

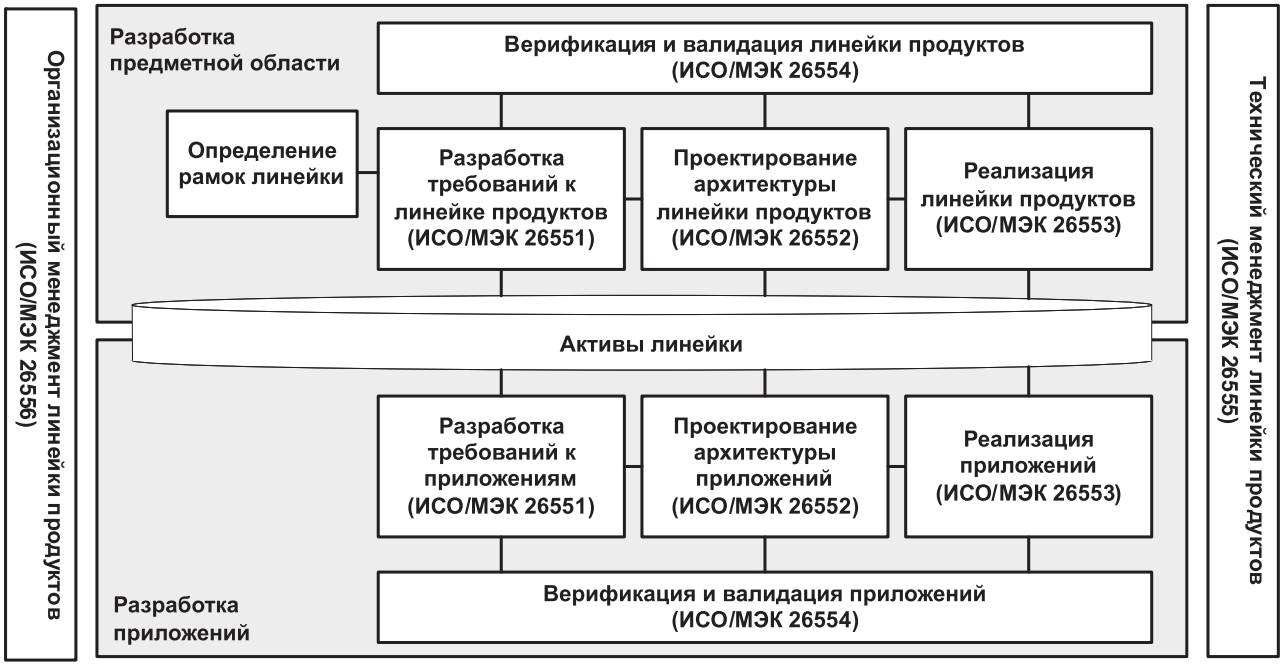

Рис. 2. Эталонная модель линеек программных продуктов и систем согласно ISO/IEC 26550:2015 [15]

применимости (ISO/IEC 26554); по организационному управлению (ISO/IEC 26556); по управлению продуктами в контексте бизнес-процессов организации-исполнителя (ISO/IEC 26560).

Планируется выпуск стандартов по определению готовности организации к реализации линейки продуктов (ISO/IEC 26561); к процессу перехода к ЛППС (ISO/IEC 26562); к процессу конфигурационного управления активами (ISO/IEC 26563).

Обзор стандарта ГОСТ Р ИСО/МЭК 26555—2016. Стандарт основан на предыдущей версии стандарта ISO/IEC 26555:2013, которая уже заменена более новой версией от 2015 г. Стандарт описывает методы и инструментальные средства технического управления линейками продуктов. Целью разработки этого стандарта является содействие формированию единого понимания процессов, инструментальных средств и методов технического управления в создании и сопровождении линеек продуктов, принятию решений о возможностях этих средств и методов, а также представлению обладающих спецификой для линейки продуктов процессов и возможностях инструментария и методов технического управления (менеджмента) [3].

На рис. 3 приведена схема процессов, определяемых стандартом. Процесс управления изменчивостью (менеджмента вариабельности [3]) определяет возможные общие и различающиеся черты в продуктах линейки и управляет моделью изменчивости. Этот процесс:

- обеспечивает создание и внесение изменений в модель изменчивости линейки продуктов с сохранением их совместимости с существующими продуктами;

Рис. 3. Определяемые стандартом ГОСТ Р ИСО/МЭК 26555—2016 процессы [3]

- определяет способ и время связывания черт в модели изменчивости с базовыми активами при создании продуктов;
- управляет документированием модели изменчивости и условий связывания, установлением и поддержанием соответствий (трассировкой [3]) между чертами и решениями модели изменчивости и базовыми активами для поддержания свойств непротиворечивости.

В перечень задач общего для всех процессов линейки процесса принятия решений (менеджмент решения для ЛППС [3]) входят, в том числе, задачи управления изменениями в модели изменчивости линейки.

Учитывая важность поддержки ретроспективного анализа принимаемых решений по управлению изменчивостью предпочтительным является явное включение обоснований (*rationale*) в процесс управления изменчивостью, как это сделано в стандарте документирования архитектуры ГОСТ Р ИСО/МЭК 42010 [16].

Процесс управления базовыми активами ("менеджмент активов" согласно работе [3]) создает и поддерживает повторно используемые базовые активы (*core assets*) линейки. Этот процесс включает:

- идентификацию повторно используемых активов среди артефактов, создаваемых при разработке продуктов;
- структурирование и адаптацию выделяемых активов для повторного использования, создание базы активов для эффективного поиска отдельных активов;
- верификацию возможности повторного использования актива;
- развитие и внесение изменений в базовые активы.

Разделяют два основных подхода к идентификации и формированию базы активов. При *проактивном* (*proactive*) подходе идентификация базового актива проводится до его создания, и необходимость его существования вытекает из модели изменчивости. При *ретроактивном* (*reactive*) подходе идентификация активов проводится в разработанных продуктах, после чего устанавливаются связи между моделью изменчивости и выделенными базовыми активами.

Процесс управления процессами ("менеджмент процессов" согласно терминологии стандарта [3]) определяет специализированные процессы технического управления линейками продуктов. К их числу относят процессы:

- разработки предметной области ("разработки домена" [3]);
- разработки продуктов ("разработки приложений" — [3]);
- мониторинга, контроля и совершенствования самих процессов;
- обеспечения готовности к их реализации.

Процессы управления аналогичного характера определены в стандартах ГОСТ Р ИСО/МЭК 12207 и ГОСТ Р ИСО/МЭК 5288.

Процессы поддержки управления (менеджмента) обеспечивают выполнение остальных процессов линейки продуктов. Управление качеством процессов линеек продуктов обеспечивает выполнение и контроль выполнения процессов линейки согласно определениям процессов. Процесс управления конфигурацией решает задачи прослеживания изменений базовых активов, процессов и продуктов в линейке. Управление конфигурацией происходит как во времени, когда одна версия актива сменяет предыдущую, так и в пространстве, когда одновременно существуют несколько версий актива в разных продуктах. Процесс управления программными средствами ("менеджмент инструментов" [3]) направлен на повышение производительности линейки продуктов за счет автоматизации процессов управления и разработки. Процесс управления рисками ("технического менеджмента рисков" согласно терминологии стандарта [3]) направлен на решение вопросов, связанных с техническими рисками в достижении поставленных перед линейкой продуктов целей. Управление решениями ("менеджмент решений" [3]) определяет порядок подготовки и принятия решений, их выполнения, программной реализации и изменения процессов, сопровождающих создание линейки продуктов.

Стандарт ГОСТ Р ИСО/МЭК 26555—2016 использует предписывающую модель процессов, которые разделены на подпроцессы, а подпроцессы, в свою очередь — на задачи. Применение стандарта требует значительной процессной дисциплины и опыта реализации процессов разработки программных систем. Такой подход вполне согласуется с тем, что основное применение линейки продуктов в настоящее время находят в отраслях промышленности с повышенными требованиями к нефункциональным показателям качества программного обеспечения.

Анализируя опыт разработки и применения других стандартов в области процессов в программной инженерии (таких как ГОСТ Р ИСО/МЭК 12207, ГОСТ Р ИСО/МЭК 15288, CMMI-DEV [17]), развитие стандарта ГОСТ Р ИСО/МЭК 26555—2016 должно предусматривать проведение оценки соответствия организации — разработчика продуктов или ее подразделения требованиям стандарта, например, согласно ISO/IEC 15504 (и замещающему его ISO/IEC 33001 [18]). Предусматривают несколько уровней возможностей (*capability*) организации, исходя из объема реализованных процессов или моделей стандарта. Данный подход позволяет внедрять стандарт постепенно, оценивать уровень соответствия продукта стандартам и управлять процессом внедрения стандартов на уровне организации. В применении к линейкам продуктов более подробный материал об этом представлен в работе [6].

В настоящей версии стандарта ГОСТ Р ИСО/МЭК 26555—2016 не отражен процесс перехода от разработки продуктов по отдельности к их разработке в рамках линейки. Отсутствие описания этого процесса ограничивает возможность применения стандарта случаями только создаваемых линеек продуктов, в то время как большинство организаций-разработчиков,

в которых линейки могли бы найти применение, уже разрабатывают семейства продуктов. Возможно, запланированный стандарт ISO/IEC 26562 будет посвящен процессу перехода к линейкам продуктов.

Кроме того, представляется существенным включение в серию рассматриваемого стандарта, описывающего совместную разработку семейства продуктов согласно модели копирования, частей продуктов (*clone-and-own*) без объединения в линейку. Совместная разработка таких продуктов без их объединения может служить промежуточным этапом перед полноценным внедрением линеек продуктов.

Текущее состояние и основные направления

По данным Scopus/SciVal[1] наиболее активная по числу публикаций исследовательская деятельность в области линеек продуктов ведется в Европейском союзе (University Litz, TU Darmstadt, University Magdeburg, University Passau и др.), Бразилии (University Pernambuco, University Namur, University Bahia и др.), США и Канаде (University Waterloo, University Toronto, SEI/CMU), Китае и Южной Корее.

В Европейском союзе было выполнено несколько крупных проектов в области создания линеек продуктов. Результаты проектов PRAISE (1998—2001 гг.), ESAPS (1999—2001 гг.), CAFE (2001—2003 гг.), FAMILIES (2003—2005 гг.), MoSiS (2007—2010 гг.) и VARIES (2012—2015 гг.) были использованы для формулировки основных положений разработки и управления линейками продуктов, для выработки серии стандартов ISO/IEC 26520 (незавершен) и последующих стандартов в ISO/IEC 26550:2013 и ISO/IEC 26550:2015 [19].

В то же время работа [20] указывает ряд причин все еще ограниченного распространения линеек продуктов. К их числу относятся отсутствие: четко сформулированных условий применения линеек продуктов в индустрии (бизнес-кейсов) и отсутствие поддержки на уровне высшего руководства компаний и отсутствие подходящих решений по организационным структурам и процессам.

Результатами проекта REVAMP2 (2016—2019 гг.) должны стать интегрированные программные средства и процесс поддержки генерации и обратной разработки (*round-trip engineering*) линеек программных продуктов и систем с применением моделей (MBSE). При этом именно MBSE становится основным подходом к реализации продуктовых линеек.

Можно привести следующие современные направления исследований в области линеек продуктов, а также характерные публикации по этим направлениям:

- моделирование предметной области и изменчивости продуктов линеек с применением SysML/UML [21, 22], стандартизированных языков моделирования изменчивости CVL [23] или OVM [24], дельта-моделирование (*delta-modeling*) [25];
- верификация моделей и продуктов в линейках, тестирование, в том числе с применением формальных методов [26];
- управление изменениями, развитие и совместное развитие моделей изменчивости и продуктов (*co-evolution*) в линейках [27];
- восстановление (*extraction*) моделей изменчивости из моделей или исходного кода, и анализ предметных областей [28];
- реализация семейств похожих продуктов без объединения их в линейку [29].

Применение в индустрии и программные средства

Основные области применения продуктовых линеек в производственной сфере — разработка программно-аппаратных комплексов с длительным жизненным циклом, таких как встраиваемые системы и системы с особыми требованиями по безопасности и надежности для аэрокосмической, автомобильной, оборонной, машиностроительной и других отраслей.

По данным Scopus из найденных 115 публикаций коммерческих компаний совместно с академическими институтами за 2012—2017 гг. примерно треть приходится на Siemens [30], IBM Research (Haifa Research Center) [29] и ABB Group [31]. Далее следуют ASELSAN Inc., General Motors, Hitachi, Daimler AG, Aerospace Corporation, SAP Research, Tata Consultancy Services, Volkswagen AG, Cisco Systems и Lockheed Martin, на которых приходится еще треть публикаций. При этом результаты Cisco по автоматическому тестированию линеек продуктов цитируются существенно выше, чем в среднем по области.

Около двух десятков хорошо документированных случаев успешного применения продуктовых линеек собрано в зале славы продуктовых линеек[2]. Отметим первое использование линеек для разработки систем управления для боевых кораблей от CelsiusTech (в настоящее время часть SAAB), внедрение продуктовых линеек в General Motors для двигателей и приводов, систем управления двигателем Robert Bosch Inc., линейку телекоммуникационных коммутаторов Ericsson AXE, а также реализацию продуктовых линеек медицинского оборудования в Philips.

Среди программных средств, поддерживающих разработку линеек продуктов и распространяемых на коммерческой основе, следует отметить BigLever Gears [32] и pure-systems pure:variants [33]. Оба средства используют модельно-ориентированный подход к управлению изменчивостью, предлагают автоматические средства компоновки готовых продуктов из базовых активов линейки по их конфигурациям, предусматривают средства технического управления линейкой и совместного развития моделей изменчивости и базовых активов.

Свои услуги и решения в области внедрения управления линейками продуктов также предлагают PTC Inc. (CAD, PLM)[3], TCS (Software

1 http://www.scopus.com, http://www.scival.com, получено в июне 2017 г.

2 http://www.splc.net/fame.html, получено в июне 2017.

3 http://www.ptc.com, Integrity Modeler, получено в июне 2017.

Factories)[4]. Многие компании используют средства собственной разработки или сочетают их со свободно распространяемыми академическими программными средствами, например, с FeatureIDE, FeatureHouse и др. [34].

Заключение

Методы и программные средства разработки линеек программных продуктов и систем развиваются уже более тридцати лет. В последнее десятилетие линейки продуктов нашли применение в автомобильной, электротехнической, авиационной и других отраслях промышленности. Индустриальный Интернет как часть Интернета вещей (IoT) и концепция четвертой индустриализации с индивидуализацией массовых продуктов требуют от организаций ведения разработки одновременно большого числа программных продуктов и систем на протяжении длительного времени.

В данной статье представлено краткое изложение современного состояния области разработки линеек продуктов, которое может быть решением обозначенных задач. Основное направление исследований в этой области смещается от базовых принципов к прикладным аспектам, применению и масштабированию методов. Исследовательские центры сосредоточены в США, Европейском союзе, Бразилии, Китае. Множество крупных компаний проводят апробацию или уже применяют линейки продуктов в производстве.

Вследствие высокой сложности процессов и методов реализации линеек продуктов, наличия нескольких центров развития и необходимости совмещения усилий возникает необходимость стандартизации в области. Работа над серией стандартов активно ведется в последние несколько лет, по ее результатам будет создан, вероятно, наиболее объемлющий набор стандартов в области разработки программных продуктов и систем.

Вступивший в действие стандарт ГОСТ Р ИСО/МЭК 26555—2016 описывает процессы технического управления линейками программных продуктов и систем. Ключевыми дополнениями к стандарту, необходимыми для его применения, должны стать аналоги стандарта ISO/IEC 26550:2015 с описанием эталонной модели процессов линеек продуктов и стандартов по управлению требованиями ISO/IEC 26551:2016 и механизмам реализации изменчивости ISO/IEC 26557:2016.

Список литературы

1. **Dhungana D., Falkner F., Haselböck A., Schreiner H.** Smart factory product lines: a configuration perspective on smart production ecosystems // In Proceedings of the 19th International Conference on Software Product Line (SPLC'15). ACM, New York, NY, USA. 2015. P. 201—210.

2. **Lasi H., Fettke P., Kemper H. G. et al.** Industry 4.0 // Business Information Systems Engineering. 2014, Vol. 6. P. 239—242.

3. **ГОСТ Р ИСО/**МЭК 26555—2016. Инструменты и методы технического менеджмента линейки продуктов. ISO/IEC 26555:2013 Systems and software engineering — Tools and methods for product line technical management.

4. **ГОСТ Р ИСО/**МЭК 12207—2010. Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств.

5. **ГОСТ Р ИСО/**МЭК 15288—2005. Информационная технология. Системная инженерия. Процессы жизненного цикла систем.

6. **Linden F. J. van der, Schmid K., Rommes E.** Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering. Springer Science & Business Media. 2007. 333 p.

7. **Clements P., Northrop L.** Software Product Lines: Practices and Patterns. Addison-Wesley Longman Publishing Co. Inc., Boston, MA, USA. 2001. 563 p.

8. **Royer J. C., Arboleda H.** Model-Driven and Software Product Line Engineering. Wiley-ISTE. 2012. 278 p.

9. **Кознов Д. В., Новицкий И. А., Смирнов М. Н.** Инструменты для управления вариативностью — готовность к промышленному применению // Труды СПИИРАН. 2013. № 3. С. 297—331.

10. **Гусс С. В.** Разработка семейства программных систем в специфической предметной области // Математические структуры и моделирование. 2011. № 22. С. 55—68.

11. **Кулямин В. В., Лаврищева Е. М., Мутилин В. С., Петренко А. К.** Верификация и анализ вариабельных операционных систем // Труды ИСП РАН. 2016. Том 28, № 3. С. 189—208.

12. **Kyo C. K., Sholom G. C., James A. H. et al.** Feature-Oriented Domain Analysis (FODA) Feasibility Study. CMU Technical Report. 1990.

13. **Pereira J. A., Krieter S., Meinicke J. et al.** FeatureIDE: Scalable Product Configuration of Variable Systems // In Proceedings of the International Conference on Software Reuse (ICSR). 2016. P. 397—400.

14. **Czarnecki K., Grünbacher P., Rabiser R. et al.** Cool features and tough decisions: a comparison of variability modeling approaches // In Proceedings of the Sixth International Workshop on Variability Modeling of Software-Intensive Systems (VaMoS'12). 2012. P. 173—182.

15. **Chimalakonda S., Dan H. L.** On the Evolution of Software and Systems Product Line Standards // SIGSOFT Softw. Eng. Notes. 2016. Vol. 41, No. 3. P. 27—30.

16. **ГОСТ Р** 57100—2016/ISO/IEC/IEEE 42010—2011 Системная и программная инженерия. Описание архитектуры.

17. **CMMI-DEV** (Version 1.3, November 2010). Carnegie Mellon University Software Engineering Institute. 2010.

18. **ISO**/IEC 33001:2015 Information technology — Process assessment — Concepts and terminology.

19. **ISO**/IEC 26550:2015. Software and systems engineering — Reference model for product line engineering and management.

20. **Metzger A., Pohl K.** Software product line engineering and variability management: achievements and challenges // In Proceedings of the on Future of Software Engineering (2014). 2014. P. 70—84.

21. **Oliveira Junior E. A., Gimenes I. M. S., Maldonado J. C.** Systematic Management of Variability in UML-based Software Product Lines // Journal of Universal Computer Science. 2010. Vol. 16, No. 17. P. 2374—2393.

22. **Gomaa H.** Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures. Addison-Wesley. 2004. 701 p.

23. **Haugen O., Wąsowski A., Czarnecki K.** Cvl: Common variability language // Proceedings of the 17th International Software Product Line Conference. 2013. P. 277—277.

24. **Pohl K., Böckle G., van der Linden F.** Software Product Line Engineering: Foundations, Principles and Techniques. Springer. 2005. 467 p.

25. **Haber A., Rendel H., Rumpe B., Schaefer I.** Delta modeling for software architectures. // Tagungsband — Dagstuhl-Workshop MBEES: Modellbasierte Entwicklung eingebetteter Systeme VII (MBEES 2011). 2011. P. 1—10.

26. **Benduhn F., Thüm T., Lochau M., Leich T., Saake G.** A Survey on Modeling Techniques for Formal Behavioral

[4] http://www.tcs.com, Tata Consultancy Services (TCS) Software Engineering Services, получено в июне 2017.

Verification of Software Product Lines // In Proceedings of the Ninth International Workshop on Variability Modelling of Software-intensive Systems (VaMoS'15). 2015. P. 80—87.

27. **Laguna M. A., Crespo Y.** A systematic mapping study on software product line evolution: From legacy system reengineering to product line refactoring // Sci. Comput. Program. 2013. Vol. 78, No. 8. P. 1010—1034.

28. **Nadi S., Berger T., Kästner C., Czarnecki K.** Mining configuration constraints: static analyses and empirical results // In Proceedings of the 36th International Conference on Software Engineering (ICSE 2014). 2014. P. 140—151.

29. **Dubinsky Y., Rubin J., Berger T. et al.** An Exploratory Study of Cloning in Industrial Software Product Lines // In Proceesings of the 17th European Conference on Software Maintenance and Reengineering. 2013. P. 25—34.

30. **Fang M., Leyh G., Doerr J., Elsner C.** Multi-variability modeling and realization for software derivation in industrial automation management // In Proceedings — 19th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MODELS 2016). 2016. P. 2—12.

31. **Koziolek H., Goldschmidt T., De Gooijer T. et al.** Experiences from identifying software reuse opportunities by domain analysis // ACM International Conference Proceeding Series. 2013. P. 208—217.

32. **Krueger C., Clements P.** Systems and Software Product Line Engineering with Gears from BigLever Software // In Proceedings of the 18th International Software Product Line Conference: Companion Volume for Workshops, Demonstrations and Tools. 2014, vol. 2. P. 121—124.

33. **Beuche D.** Using Pure: Variants Across the Product Line Lifecycle // In Proceedings of the 20th International Systems and Software Product Line Conference. 2016. P. 333—336.

34. **Bashroush R., Garba M., Rabiser R., Groher I., Botterweck G.** CASE Tool Support for Variability Management in Software Product Lines // ACM Comput. Surv. 2017. Vol. 50, No. 1, Article 14.

Product Line Engineering: Standards and State-of-Practice

A. S. Khritankov, anton.khritankov@acm.org, Moscow institute of physics and technology, Dolgoprudny, Moscow Region, 141701, Russian Federation

Corresponding author:

Khritankov Anton S., Associate professor, Moscow institute of physics and technology, Dolgoprudny, Moscow Region, 141701, Russian Federation,
E-mail: anton.khritankov@acm.org

Received on June 14, 2017
Accepted on June 20, 2017

A recent version of the ISO/IEC 26555 standard comes into force as a national standard in Russia in June 2017. The standard describes technical management processes for systems and software product lines. In this paper, we review the standard and the current state of research and industrial practice in the field.

Mass product customization in Industry 4.0, Industrial Internet (IoT), ubiquitous computing are major pillars in the Horizon 2020 programme of the EC. And this means there is need for a lot of customizable software systems. Software product lines may be the key.

While a popular topic worldwide, SPLE has missed attention in Russian Federation with only a few papers on the topic. Thus we briefly introduce main concepts of the SPLE such as product line, variability model, strategic reuse, reference architecture, feature and decision models, product configuration. Then we provide an example feature model built with FeatureIDE.

Applications of PLE include shared development of a family of products each targeting a specific market segment, development of a large evolving cyber-physical system with several versions delivered over time and a combination of both.

International standardization process started with a failed attempt of ISO/IEC 26520 series and continued with more successful ISO/IEC 26550-26564 standards, some of which is still under development. Russian national standardization committee adopted the SPL technical management ISO/IEC 26555:2013 as national standard in 2016, omitting other published standards in the series.

Extensions and additions to the standard should consider tracking rationale for the changes being introduced to the SPL configuration and variability model in particular. Another shortcoming is absence of a model for gradual introduction of the standard (e.g. capability or level-based). Process capability assessment may be governed by ISO/IEC 33001, but this option is not mentioned in the standard.

Current research centers in SPLE are Univ. Litz, TU Darmstadt, Univ. Magdeburg, Univ. Passau and other in EU; Univ. Pernambuco, Univ. Namur, Univ. Bahia and other in Brazil; Univ. Waterloo, Univ. Toronto, SEI/CMU in US and Canada, several universities in China and South Korea. EU also runs a series of projects uniting research universities and commercial companies to help spread fresh developments in PLE, such as FAMILIES, VARIES and current REVAMP2 (2016-2019).

Major research directions are variability modeling and languages (with and without UML/SysML), verification and traceability, variability model extraction, concurrent evolution of variability models and assets, transition to clone-and-own and PLE, and, notably, model-based software engineering (MBSE) for SPLE.

Notable industrial users of SPLE are Siemens, Philips, General Motors, ABB Group with others following, including Thales and SAAB (through Celsius Tech).

Although PLE may be the next (old) big thing in software development to support mass customization through strategic reuse, it requires a considerable upfront investment and strong process discipline. Currently developed standards and available tools may help practitioners and tool vendors lower this investment to a level, manageable by small and mediun range companies.

Keywords: *Internet of Things (IoT), PLE, product lines, ISO/IEC 26550, strategic reuse*

For citation:

Khritankov A. S. Product Line Engineering: Standards and State-of-Practice, *Programmnaya Ingeneria,* 2017, vol. 8, no. 9, pp. 387—395.

DOI: 10.17587/prin.8.387-395

References

1. **Dhungana D., Falkner F., Haselböck A., Schreiner H.** Smart factory product lines: a configuration perspective on smart production ecosystems, *Proceedings of the 19th International Conference on Software Product Line (SPLC'15),* ACM, New York, NY, USA, 2015, pp. 201—210.

2. **Lasi H., Fettke P., Kemper H. G., Feld T., Hoffmann M.** Industry 4.0, *Business Information Systems Engineering*, 2014, vol. 6, pp. 239—242.

3. **GOST R ISO/**MJeK 26555—2016. Instrumenty i metody tehnicheskogo menedzhmenta linejki produktov. ISO/IEC 26555:2013 Systems and software engineering — Tools and methods for product line technical management.

4. **GOST R ISO**/MJeK 12207—2010. Informacionnaja tehnologija. Sistemnaja i programmnaja inzhenerija. Processy zhiznennogo cikla programmnyh sredstv. ISO/IEC 12207 Systems and software engineering — Software life cycle processes.

5. **GOST R ISO**/MJeK 15288—2005. Informacionnaja tehnologija. Sistemnaja inzhenerija. Processy zhiznennogo cikla sistem. ISO/IEC/IEEE 15288:2015 — Systems and software engineering — System life cycle processes.

6. **Linden F. J. van der, Schmid K., Rommes E.** *Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering*, Springer Science & Business Media, 2007, 333 p.

7. **Clements P., Northrop L.** *Software Product Lines: Practices and Patterns.* Addison-Wesley Longman Publishing Co. Inc., Boston, MA, USA, 2001, 563 p.

8. **Royer J. C., Arboleda H.** *Model-Driven and Software Product Line Engineering,* Wiley-ISTE, 2012, 278 p.

9. **Koznov D. V., Novickij I. A., Smirnov M. N.** Instrumenty dlja upravlenija variativnost'ju — gotovnost' k promyshlennomu primeneniju (Tools for managing variability — readiness for industrial applications), *Trudy SPIIRAN*, 2013, no. 3, pp. 297—331 (in Russian).

10. **Guss S. V.** Razrabotka semejstva programmnyh sistem v specificheskoj predmetnoj oblasti (Development of software system families in specific problem domain), *Matematicheskie struktury i modelirovanie*, 2011, no. 22, pp. 55—68 (in Russian).

11. **Kuljamin V. V., Lavrishheva E. M., Mutilin V. S., Petrenko A. K.** Verifikacija i analiz variabel'nyh operacionnyh system (Verification and analysis of variability of operating systems), *Trudy ISP RAN*, 2016, vol. 28, no. 3, pp. 189—208 (in Russian).

12. **Kyo C. K., Sholom G. C., James A. H., William E. N., Peterson A. S.** Feature-Oriented Domain Analysis (FODA) Feasibility Study, *CMU Technical Report,* 1990.

13. **Pereira J. A., Krieter S,, Meinicke J., Schröter R., Saake G., Leich T.** FeatureIDE: Scalable Product Configuration of Variable Systems, in *Proceedings of the International Conference on Software Reuse (ICSR)*, 2016, pp. 397—400.

14. **Czarnecki K., Grünbacher P., Rabiser R., Schmid K., Wąsowski A.** Cool features and tough decisions: a comparison of variability modeling approaches, in *Proceedings of the Sixth International Workshop on Variability Modeling of Software-Intensive Systems (VaMoS'12)*, 2012, pp. 173—182.

15. **Chimalakonda S., Dan H. L.** On the Evolution of Software and Systems Product Line Standards, *SIGSOFT Softw. Eng. Notes.*, 2016, vol. 41, no. 3, pp. 27—30.

16. **GOST R** 57100-2016/ISO/IEC/IEEE 42010:2011 Sistemnaja i programmnaja inzhenerija. Opisanie arhitektury. ISO/IEC/IEEE 42010 Systems and software engineering — Architecture description.

17. **CMMI-DEV** (Version 1.3, November 2010), Carnegie Mellon University Software Engineering Institute, 2010.

18. **ISO**/IEC 33001:2015 Information technology — Process assessment — Concepts and terminology.

19. **ISO**/IEC 26550:2015. Software and systems engineering — Reference model for product line engineering and management.

20. **Metzger A., Pohl K.** Software product line engineering and variability management: achievements and challenges, in *Proceedings of the on Future of Software Engineering (2014)*, 2014, pp. 70—84.

21. **Oliveira Junior E. A., Gimenes I. M. S., Maldonado J. C.** Systematic Management of Variability in UML-based Software Product Lines, *Journal of Universal Computer Science,* 2010, vol. 16, no. 17, pp. 2374—2393.

22. **Gomaa H.** *Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures,* Addison-Wesley, 2004, 701 p.

23. **Haugen O., Wąsowski A., Czarnecki K.** Cvl: Common variability language, in *Proceedings of the 17th International Software Product Line Conference*, 2013, pp. 277—277.

24. **Pohl K., Böckle G., van der Linden F.** *Software Product Line Engineering: Foundations*, Principles, and Techniques. Springer, 2005, 467 p.

25. **Haber A., Rendel H., Rumpe B., Schaefer I.** Delta modeling for software architectures, *Tagungsband — Dagstuhl-Workshop MBEES: Modellbasierte Entwicklung eingebetteter Systeme VII (MBEES 2011)*, 2011, pp. 1—10.

26. **Benduhn F., Thüm T., Lochau M., Leich T., Saake G.** A Survey on Modeling Techniques for Formal Behavioral Verification of Software Product Lines, in *Proceedings of the Ninth International Workshop on Variability Modeling of Software-intensive Systems (VaMoS'15)*, 2015, pp. 80—87.

27. **Laguna M. A., Crespo Y.** A systematic mapping study on software product line evolution: From legacy system reengineering to product line refactoring, *Sci. Comput. Program.*, 2013, vol. 78, no. 8, pp. 1010—1034.

28. **Nadi S., Berger T., Kästner C., Czarnecki K.** Mining configuration constraints: static analyses and empirical results, *in Proceedings of the 36th International Conference on Software Engineering (ICSE 2014)*, 2014, pp. 140—151.

29. **Dubinsky Y., Rubin J., Berger T., Duszynski S., Becker M., Czarnecki K.** An Exploratory Study of Cloning in Industrial Software Product Lines, *in Proceedings of the 17th European Conference on Software Maintenance and Reengineering*, 2013, pp. 25—34.

30. **Fang M., Leyh G., Doerr J., Elsner C.** Multi-variability modeling and realization for software derivation in industrial automation management, *Proceedings — 19th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MODELS 2016)*, 2016, pp. 2—12.

31. **Koziolek H., Goldschmidt T., De Gooijer T., Domis D., Sehestedt S.** Experiences from identifying software reuse opportunities by domain analysis, *ACM International Conference Proceeding Series*, 2013, pp. 208—217.

32. **Krueger C., Clements P.** Systems and Software Product Line Engineering with Gears from BigLever Software, in *Proceedings of the 18th International Software Product Line Conference: Companion Volume for Workshops, Demonstrations and Tools*, 2014, vol. 2, pp. 121—124.

33. **Beuche D.** Using Pure: Variants Across the Product Line Lifecycle, In *Proceedings of the 20th International Systems and Software Product Line Conference*, 2016, pp. 333—336.

34. **Bashroush R., Garba M., Rabiser R., Groher I., Botterweck G.** CASE Tool Support for Variability Management in Software Product Lines, *ACM Comput. Surv.*, 2017, vol. 50, no. 1, article 14.

УДК 004.434 DOI: 10.17587/prin.8.396-406

А. О. Сухов, канд. физ.-мат. наук, доц., e-mail: ASuhov@hse.ru,
Национальный исследовательский университет "Высшая школа экономики", г. Пермь

Предметно-ориентированный язык описания трансформаций визуальных моделей

Рассмотрен подход к трансформации визуальных моделей, основанный на применении предметно-ориентированного языка для описания правил преобразования. Использование такого языка позволяет пользователям, не являющимся IT-специалистами, задавать такие трансформации в иные графические нотации и текстовое представление. Описаны основные конструкции языка, приведены примеры их использования. Интерпретатор языка реализован в инструментальном средстве MetaLanguage.

Ключевые слова: *предметно-ориентированный язык, трансформация моделей, визуальные модели, модельно-ориентированный подход, языковой инструментарий, система MetaLanguage, многоуровневое моделирование, метамодель, графовые грамматики*

Введение

Современная разработка программного обеспечения требует участия представителей заказчика и будущих пользователей в процессе создания программного продукта, поскольку именно они знают специфику предметной области, в которой будет функционировать система. Отсутствие учета этой специфики приведет к тому, что система не будет соответствовать реально выполняемым бизнес-процессам, а следовательно, в дальнейшем возникнет необходимость ее доработки или отказа от эксплуатации.

Привлечение к процессу создания и сопровождения программного обеспечения представителей заказчика выявляет еще один проблемный вопрос — эффективную коммуникацию между IT-специалистами и экспертами в предметной области. Большинство экспертов не имеет даже базовых знаний в области разработки программного обеспечения (ПО), поэтому им сложно понять и оценить предложенные программистами решения, описанные на языках программирования.

Использование модельно-ориентированного (*Model-Driven Development*, MDD) и предметно-ориентированного (*Domain-Specific Development*, DSD) подходов в процессе создания ПО способно разрешить этот вопрос. Основная идея MDD заключается в создании визуальных моделей, которые отражают проектные решения, и в автоматической генерации исходного кода реализации системы на целевом языке программирования. Такой подход значительно сокращает время разработки, минимизирует трудозатраты и число ошибок [1, 2]. Кроме того, визуальные модели наглядны, что позволяет экспертам в предметной области более глубоко понять проектные решения.

Предметно-ориентированная разработка предполагает применение в процессе построения моделей визуальных предметно-ориентированных языков (*Domain-Specific Language*, DSL), оперирующих терминами предметной области [3—5]. Такие языки способны сократить семантический разрыв между средствами разработки и объектами предметной области [6]. Оперировать конструкциями DSL могут даже представители заказчиков и будущие пользователи.

Если разрабатываемая система относится к системам, построенным на интерпретации метаданных, т. е. в процессе своего функционирования она способна на основе интерпретации моделей предметной области изменить свое поведение, то ее настройку и доработку могут выполнять не IT-специалисты, а эксперты в предметной области с помощью предметно-ориентированных языков.

Процесс применения MDD и DSD при создании ПО можно разделить на следующие этапы: построение моделей, отражающих проектные решения разработчиков, генерация исходного кода системы на основе построенных моделей, "ручная" доработка кода. Этап генерации кода на основе визуальных моделей должен выполняться автоматически. За выполнение такого преобразования отвечает специальный компонент среды разработки — трансформатор (генератор). Для большинства стандартных визуальных нотаций (IDEF0, ERD, UML, BPMN и др.) правила преобразования давно известны и реализованы во многих CASE-системах. Но для новых предметно-ориентированных языков, которые были разработаны для создания ПО, таких правил не существует.

Возникает необходимость определения правил преобразования моделей, созданных с использованием языков DSL, в стандартные визуальные нотации или в текстовое представление на целевом языке.

Для создания трансформаторов могут быть использованы различные подходы к преобразованию визуальных моделей. Некоторые из них базируются на теории графов и графовых грамматиках, другие применяют в процессе своей работы иные методы Computer Science [7—10].

Основным ограничением большинства подходов является то обстоятельство, что для описания правил преобразования они применяют языки высокого уровня, поэтому сложны для понимания экспертам в предметной области.

В инструментальном средстве создания визуальных предметно-ориентированных языков MetaLanguage [11] разработан компонент преобразования моделей, построенных с помощью DSL, в иные графические нотации или в код на целевом языке программирования. Однако реализованный в такой среде подход имеет ряд ограничений. В их числе отсутствие возможности:

— выполнения арифметических операций над значениями атрибутов в ходе их преобразования, например, невозможно вычислить разность значений атрибутов;

— описания ветвления потока выполнения трансформации при проверке заданного условия;

— навигации по элементам модели, их атрибутам, например, нельзя обойти все атрибуты сущности или отношения модели;

— вызова функций внешних библиотек, например, функции поиска строки в подстроке, математических функций.

Для устранения этих недостатков было решено разработать и интегрировать в MetaLanguage текстовый предметно-ориентированный язык (TDSL), позволяющий описывать правила преобразования визуальных моделей в иные графические нотации и текстовое представление на каком-либо целевом языке. Исходя из методов трансформации, применяемых в языковом инструментарии MetaLanguage, при разработке TDSL будут использоваться графовые грамматики и алгебраический подход к трансформации моделей.

Практическая значимость работы заключается в том, что разработанный язык TDSL позволит повысить функциональные возможности компонента трансформации визуальных моделей системы MetaLanguage.

Система MetaLanguage

Языковой инструментарий MetaLanguage — средство разработки визуальных предметно-ориентированных языков, создания моделей с использованием этих языков и трансформации созданных моделей.

В процессе своего функционирования MetaLanguage строит многоуровневую модель предметной области (рис. 1):

- модель состояния описывает текущее состояние предметной области и соответствует объектам предметной области;
- модель предметной области — метаданные, описывающие структуру объектов предметной области;
- метамодель — модель языка моделирования, применяемого для создания моделей, которая описывает основные конструкции языка, связи между ними, ограничения;

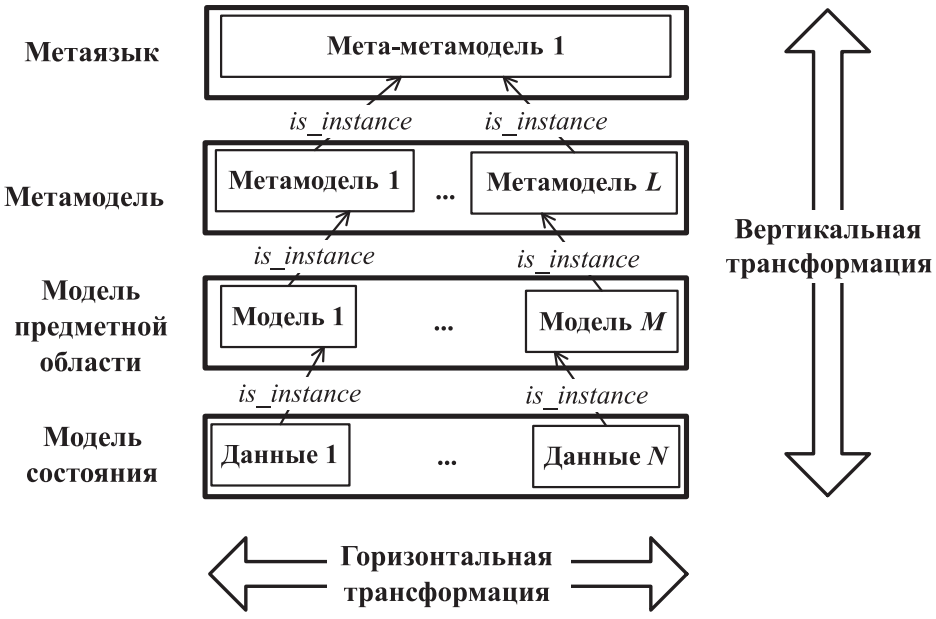

Рис. 1. Многоуровневая модель предметной области

- мета-метамодель (метаязык) — язык для построения метамоделей, т. е. для создания других языков моделирования.

Для построения предметно-ориентированных языков MetaLanguage использует метаязык, содержащий следующие конструкции [12]:

- сущность, которая позволяет описать конструкции DSL, представляемые в виде отдельных элементов метамодели;
- отношение, которое предназначено для определения связей между сущностями и может быть одного из трех типов — ассоциация, агрегация, наследование;
- ограничение, которое предоставляет возможность задать условия, налагаемые на сущности, отношения, значения их атрибутов.

По окончании разработки метамодели пользователь получает в свое распоряжение DSL, с помощью которого может выполнять построение визуальных моделей, создавая экземпляры сущностей и отношений метамодели и задавая значения их атрибутам.

Трансформации моделей в MetaLanguage

На настоящее время система MetaLanguage позволяет выполнять вертикальные и горизонтальные трансформации. *Вертикальная трансформация* представляет собой преобразование модели, находящейся на одном уровне иерархии, в модель, представленную на другом уровне иерархии (рис. 1). Такая трансформация выполняется, например, при построении модели на основе метамодели или при интерпретации элементов модели.

Горизонтальная трансформация — это преобразование модели, находящейся на одном уровне иерархии, в модель того же уровня (рис. 1). Такая трансформация осуществляется, например, в процессе преобразования модели, выполненной в нотации ERD, в диаграмму классов языка UML или в скрипт на языке SQL.

В зависимости от того, в визуальном или текстовом представлении будет выполнена целевая модель, все трансформации можно разделить на два вида: "модель—модель" и "модель—текст". *Трансформация вида "модель—модель"* позволяет выполнить преобразование визуальной модели в модель, представленную в другой графической нотации. *Трансформация вида "модель—текст"* предназначена для преобразования визуальных моделей в текстовое представление.

Собственно процесс трансформации описывается правилами, каждое из которых предназначено для преобразования некоторого подмножества элементов исходной метамодели в элементы целевой метамодели или в текстовые шаблоны. Все правила описываются на уровне метамоделей, а применяются на уровне моделей. Такой подход позволяет, определив однажды трансформацию для некоторого DSL, применять ее для любых моделей, созданных в этой нотации.

Правило состоит из имени, которое однозначно идентифицирует правило, левой и правой частей правила. *Левая часть* правила — это совокупность элементов исходной метамодели. *Правая часть* правила трансформации вида "модель—модель" — это подмножество элементов целевой метамодели. Правая часть правила трансформации вида "модель—текст" представляет собой текстовый шаблон, на основе которого выполняется генерация кода. *Шаблон* состоит из статической и динамической частей. Статическая часть не зависит от значений атрибутов элементов модели и остается постоянной для любой модели. Динамическая часть включает значения атрибутов элементов модели, поэтому изменяется от модели к модели. Для выделения динамической части используется специальный метасимвол "@" (коммерческое at). Данный метасимвол позволяет различать название элементов модели и статический текст шаблона. Для обращения к элементам модели используется префикс "М.", например, чтобы получить список всех экземпляров сущностей модели, следует написать "@М.Имя_сущности". Чтобы получить значение атрибута экземпляра сущности/отношения, следует указать префикс "С."/"О.", например, для получения значения атрибута экземпляра сущности необходимо написать "@С.Имя_сущности.Имя_атрибута".

Для создания правила трансформации пользователю необходимо указать название правила, определить его левую часть. Для этого с помощью мыши нужно: выделить фрагмент исходной метамодели, задать правую часть правила; указать фрагмент целевой метамодели (для трансформаций вида "модель—модель") или создать текстовый шаблон правила (для трансформаций вида "модель—текст"). Пример описания правила трансформации вида "модель—текст", которое позволяет на основе экземпляра сущности *Сущность* нотации Entity-Relationship Diagram сгенерировать команду создания таблицы на языке SQL, представлен на рис. 2.

Как видно на рис. 2, в левой части окна задания правила трансформации представлен фрагмент исходной метамодели — сущность *Сущность*. Правая часть окна содержит текстовое поле, в котором пользователь может определить текстовый шаблон на целевом языке (SQL), содержащий вставки по извлечению значений атрибутов элементов модели. В представленном примере динамическая часть правила "@С.Сущность.Имя" позволяет получить значение атрибута *Имя* каждого экземпляра сущности *Сущность*. После того как значение атрибута будет получено, система вставит его в код на целевом языке. Таким образом, для сущностей с именами *Студент*, *Преподаватель*, *Дисциплина* будет получен следующий фрагмент кода на целевом языке:

```
CREATE TABLE Студент(id INTEGER primary key)
CREATE TABLE Преподаватель(id INTEGER primary key)
CREATE TABLE Дисциплина(id INTEGER primary key)
```

После задания трансформации пользователь может применить ее к созданной модели. При этом система применяет каждое правило до тех пор пока это

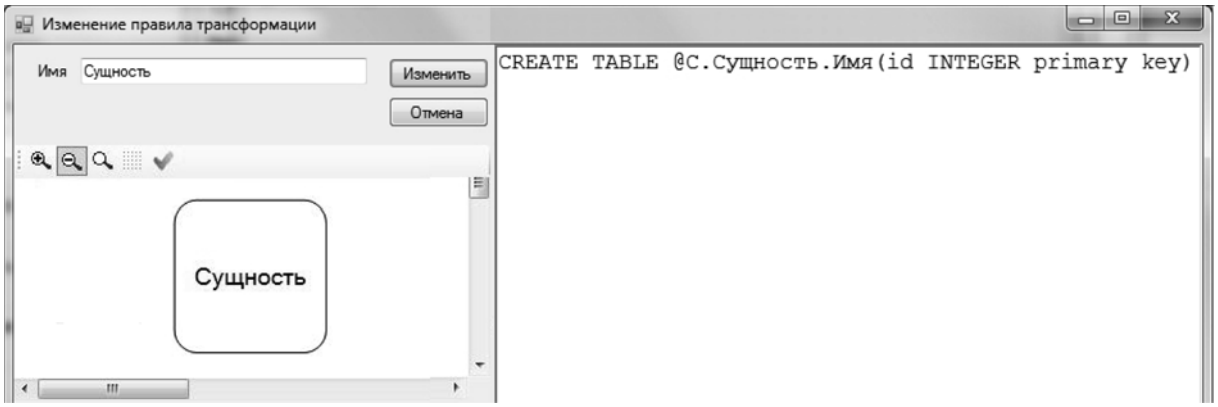

Рис. 2. Задание правила трансформации вида "модель—текст"

возможно. Для применения правила MetaLanguage ищет вхождение в исходную модель левой части правила. Для каждого такого вхождения в системе выполняется правая часть правила (генерация текстового представления или построение визуальной модели в целевой нотации). Подробное описание подхода к трансформации моделей в системе MetaLanguage приведено в работе [8].

Такой подход позволяет в декларативной форме полностью задать правила преобразования моделей. Он понятен экспертам в предметной области, однако имеет ряд ограничений, упомянутых во введении. Разрабатываемый язык TDSL позволит устранить эти ограничения. Такой язык, с одной стороны, не предъявляет высоких требований к уровню квалификации пользователей. С другой стороны, он содержит конструкции, позволяющие выполнять арифметические операции, проверку условий и навигацию по элементам моделей или пользовательским коллекциям, предоставляющие возможности вызывать заранее предопределенные функции, задавать значения атрибутов.

Анализ методов трансформации визуальных моделей

Существуют различные подходы к трансформации визуальных моделей и реализующие их инструментальные средства. Проанализируем языки, используемые в этих инструментальных средствах для описания трансформаций, с целью выявления возможностей, которые могут быть реализованы в компоненте трансформации системы MetaLanguage.

При сравнении возможностей языков будут использованы перечисленные далее критерии.

- *Полнота базового набора конструкций языка.* Язык описания правил преобразования должен обязательно включать конструкции условного оператора, оператора цикла, возможности выполнения арифметических операций над значениями атрибутов сущностей и отношений, вызова функций внешних библиотек. Это позволит устранить ряд ограничений, существующих на данный момент в компоненте трансформации системы MetaLanguage.
- *Виды описываемых трансформаций.* Язык должен поддерживать возможность задания трансформаций вида "модель—модель" и "модель—текст", это позволит выполнять преобразования моделей как в иные графические нотации, так и в текстовое представление.
- *Способ описания трансформаций.* Графический способ предоставляет возможность определять правила преобразования с помощью мыши, а текстовая нотация позволяет выполнять навигацию по элементам моделей, описывать арифметические операции над их атрибутами и условия применения правил.
- *Язык, используемый для описания трансформаций.* Средства, которые применяют для описания преобразований моделей языки высокого уровня, сложны для пользователей, не являющихся IT-специалистами.

Рассмотрим подходы к трансформации визуальных моделей подробнее.

Attributed Graph Grammar (AGG) — инструментальная среда для описания и выполнения трансформаций типизированных атрибутных графов, реализованная на платформе Java [13, 14]. Для задания преобразований графов используются конструкции языка Java, что предъявляет высокие требования к уровню квалификации пользователя. Для трансформации моделей необходимо, чтобы исходная и целевая модели были описаны в одной нотации. Вместе с тем это требование означает, что инструментальная среда не позволяет выполнять преобразования визуальной модели в иные графические нотации и текстовое представление, что является существенным ее ограничением.

Язык Graph REwriting And Transformation (GReAT), предназначен для описания преобразований моделей, базирующихся на тройных графовых грамматиках [15—17]. Исходная и целевая метамодели при его использовании объединяются в единый граф, на котором задаются правила отображения между их конструкциями. Правила трансформации описываются на языке C++, что затрудняет использование данного подхода пользователями, не являющимися IT-специалистами. Основным преимуществом GReAT является возможность объединения нескольких метамоделей в единый граф, что позволяет

выполнять трансформации моделей, описанных с использованием более одного языка.

Инструментальные средства VIsual Automated model TRAnsformations (VIATRA) основаны на правилах и паттернах преобразования графовых моделей [18, 19]. В рамках VIATRA для описания преобразований применяется текстовый язык VTL, базирующийся на логической парадигме. К преимуществам этого подхода следует отнести возможность определения правил преобразования в графическом и текстовом видах. Инструментарий VIATRA позволяет: проверять условия, налагаемые на объекты предметной области и связи между ними; выполнять выборку объектов, удовлетворяющих этим условиям. Основным его недостатком является то, что язык не содержит операторов цикла и возможности выполнения арифметических операций над значениями атрибутов.

Стандарт трансформации моделей *Query/View/Transformation* (QVT) разработан консорциумом OMG для реализации подхода MDA (Model-Driven Architecture) [20—22]. Для описания трансформации платформо-независимых моделей в платформо-зависимые в QVT предлагается использовать язык, близкий к высокоуровневым языкам программирования. Достоинством данного подхода является то обстоятельство, что он базируется на стандартизированных языках MOF и OCL. Однако язык QVT может быть использован лишь для описания трансформаций вида "модель—модель". К ограничениям этого подхода также следует отнести сложность синтаксиса языка, отсутствие возможности визуального определения правил преобразования, а также тот факт, что в языке отсутствуют оператор цикла, условный оператор и возможность вызова предопределенных функций.

Язык трансформации ATL (ATLAS Transformation Language) основан на стандарте QVT. Этот язык разработан как часть платформы ATLAS Model Management Architecture [23, 24]. Как и QVT, язык ATL базируется на стандартизованном языке задания ограничений OCL, что предоставляет возможность в ходе задания преобразований использовать стандартные типы, фильтры, помощники языка OCL. Язык предъявляет высокий уровень требований к уровню квалификации пользователя, так как кроме ATL необходимо знать еще OCL. В языке отсутствуют конструкции условного оператора, операторов цикла, возможность вызова предопределенных функций.

Подход к трансформации моделей *Model Transformation By-Example* (MTBE) не предполагает явное определение пользователем правил преобразования. Такие правила генерируются автоматически на основе анализа примеров — наборов отображений исходной модели в целевую, которые пользователь подал на вход системе [25—27]. Это значительно упрощает процесс описания преобразований моделей и снижает уровень требований к квалификации пользователя. Однако при применении данного подхода отсутствует уверенность в том, что были сгенерированы все правила и они корректны. Причина в том, что результат сильно зависит от исходного набора обучающих примеров.

С более подробным описанием проанализированных средств трансформации моделей можно ознакомиться в работе [28].

Как следует из представленного выше анализа, в языковом инструментарии MetaLanguage отсутствует возможность текстового определения правил преобразования. Кроме того, компонент трансформации не позволяет выполнять навигацию по элементам моделей, задавать арифметические операции над их атрибутами, определять условия применения правила, а также вызывать предопределенные функции. Разработка текстового предметно-ориентированного языка трансформации моделей позволит устранить эти ограничения. Результаты сравнения подходов с возможностями системы MetaLanguage, полученными после реализации в ней интерпретатора TDSL, приведены в таблице.

Рассмотрим описание конструкций TDSL и их использование на практике более подробно.

Результаты сравнения подходов к трансформации моделей

Подход к трансформации	Полнота базового набора конструкций языка	Виды описываемых трансформаций	Способ описания трансформаций	Язык, используемый для описания трансформаций
AGG	Полон	Модель—модель в рамках одной нотации	Текстовый	Java
GReAT	Полон	Модель—модель	Текстовый	C++
VIATRA	Неполон	Модель—модель, модель—текст	Графический, текстовый	MOF, VTL
QVT	Неполон	Модель—модель	Текстовый	MOF, OCL
ATL	Неполон	Модель—модель	Текстовый	MOF, OCL
MTBE	Текстовый язык отсутствует	Модель—модель	Правила генерируются автоматически	Правила генерируются автоматически
MetaLanguage	Полон	Модель—модель, модель—текст	Графический, текстовый	MetaLanguage

Описание синтаксиса языка

Как было отмечено выше, текстовый шаблон в правой части правила содержит статическую и динамическую части. Статическая часть описывается на целевом языке, на котором должен быть сгенерирован код. Такая часть может содержать динамические вставки, которые позволяют добавить в генерируемый код фрагменты текста, зависящие от модели.

Рассмотрим пример описания правой части правила трансформации вида "модель—текст". Пусть в метамодели определена сущность *Компания* со следующими атрибутами: *Название* — атрибут строкового типа, который позволяет идентифицировать компанию; *Доход* — атрибут вещественного типа, который хранит значение дохода компании в текущем месяце; *Расход* — атрибут вещественного типа, который хранит значение расхода компании в текущем месяце. Необходимо для каждой компании вывести текст, который содержит ее название и информацию о доходе. В этом случае правая часть правила будет иметь следующий вид:

```
Доход компании @C.Компания.Название составляет
                @C.Компания.Доход.
```

Для обращения к элементам графического языка используются префиксы "M.", "C." и "O.", после которых следует указать имя элемента графической модели и имя его атрибута.

В соответствии с перечисленными выше требованиями динамическая часть правила, описываемая на TDSL, может содержать оператор цикла, условный оператор, позволять определять арифметические операции над значениями атрибутов, вызывать функции из внешних библиотек. Необходимо предусмотреть возможность построения вложенных операторов. Опишем синтаксис разрабатываемого языка с помощью расширенных форм Бэкуса-Наура:

```
правая _ часть _ правила = {статическая _ часть
        | динамическая _ часть};
статическая _ часть = {литера},
```

где *литера* — печатный символ таблицы ASCII;

```
динамическая _ часть = оператор;
оператор = {литера}(составной _ оператор |
        оператор _ цикла
| условный _ оператор | оператор _ присваивания |
выражение | вызов _ функции) {литера}.
```

Конструкция *составной оператор* используется для описания блока операторов. В качестве операторных скобок используются открывающая и закрывающая фигурные скобки:

```
составной _ оператор = "@{" оператор {оператор} "@}".
```

Для того чтобы интерпретатор языка мог отличать терминальные символы от статической части правила трансформации, в которой могут использоваться те же самые литеры языка, будем в начале некоторых терминальных символов указывать метасимвол "@" (коммерческое at).

Условный оператор аналогичен условным операторам высокоуровневых языков программирования. Он содержит условие и две ветви — "то" и "иначе". Ветвь "иначе" может быть опущена:

```
условный _ оператор = "@если" выражение "@то" оператор
        ["@иначе" оператор].
```

Выражение в условном операторе должно иметь логический тип:

```
выражение = простое _ выражение |
простое _ выражение операция _ отношения
        простое _ выражение.
```

Операция отношения позволяет сравнить два простых выражения, в качестве которых может выступать значение атрибута экземпляра сущности или отношения, константа, результат работы функции, параметр совместного цикла, обозначаемый терминалом "@value", или арифметические операции над этими элементами:

```
операция _ отношения = "@=" | "@!= " |
        "@<" | "@<="| "@>" | "@>=";
простое _ выражение = слагаемое {("@+" | "@-" | "@или")
        слагаемое};
слагаемое = множитель {("@*" | "@/"| "@и") множитель};
множитель = значение _ атрибута | константа | "@value"
        | вызов _ функции |"(" выражение ")";
значение _ атрибута = ("@C." имя _ сущности |
"@O." имя _ отношения | "@value") "." имя _ атрибута,
```

здесь *имя_сущности* — имя одной из сущностей метамодели, находящейся в левой части правила; *имя_отношения* — имя одного из отношений метамодели, находящегося в левой части правила; *имя_атрибута* — имя атрибута сущности, отношения или параметра совместного цикла, если он имеет ссылочный тип;

```
константа = ["+"|"—"] число | строка | логическая _
        константа | "@null";
число = целое | вещественное;
целое = цифра {цифра};
вещественное = целое "." целое "E" ["+"|"—"] целое |
        целое "." целое;
логическая _ константа = "@истина" | "@ложь".
```

Значение атрибута, константа и возвращаемое функцией значение могут иметь один из четырех типов: знаковый целый (4 байта), знаковый вещественный (одинарной точности), строковый, логический. Строковые константы состоят из печатных символов таблицы ASCII и заключаются в кавычки. Значение атрибута может иметь также ссылочный тип.

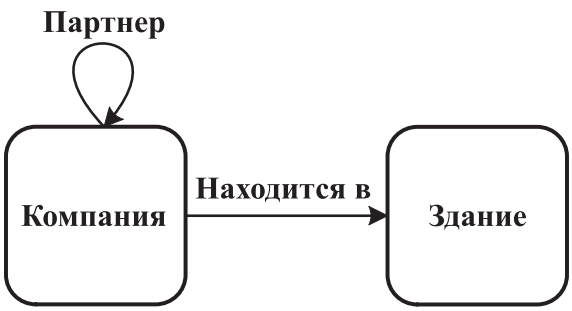

Рис. 3. Сущности *Компания* и *Здание*

Этот тип имеют те атрибуты, которые указывают на экземпляры сущностей или отношений. Например, атрибут *Адрес* сущности *Компания* указывает на экземпляр сущности *Здание*, связанный с *Компанией* ассоциацией (рис. 3). Для задания значения атрибута ссылочного типа необходимо присвоить ему имя соответствующего экземпляра сущности/отношения.

Арифметические операции могут выполняться лишь над операндами целого, вещественного, строкового, логического типов. Причем для выполнения операции необходимо, чтобы операнды были одного типа. Над операндами строкового типа может быть выполнена только операция конкатенации строк с помощью терминала "@+", над операндами логического типа — только логические операции с помощью терминалов "@и", "@или".

Терминал "@null" используется для обозначения пустого значения атрибута строкового или ссылочного типов. Например, если значение атрибута строкового типа равно пустой строке, или значение атрибута ссылочного типа не задано, то такой атрибут будет иметь пустое значение.

Рассмотрим пример использования условного оператора при описании правил трансформации. Пусть в рассмотренной ранее предметной области требуется определить, прибыль или убыток получила компания в текущем месяце. Для этого построим правило со следующей правой частью:

```
@если @С.Компания.Доход @- @С.Компания.Расход @> 0
    @то Компания работает в прибыль
    @иначе Компания работает в убыток.
```

Оператор *вызова функции* позволяет в правиле вызывать функции динамических библиотек, которые ранее были подключены к системе. Параметры функции указываются в круглых скобках через запятую, в случае их отсутствия скобки остаются пустыми:

```
вызов_функции = "@" имя_функции "(" [параметр
                {"," параметр}]")",
```

здесь *имя_функции* — имя, которое однозначно идентифицирует функцию во всех подключенных динамических библиотеках;

```
параметр = простое_выражение.
```

Оператор цикла разрабатываемого языка TDSL подобен совместному циклу языков программирования высокого уровня:

```
оператор_цикла = "@для value в" коллекция оператор.
```

Коллекция представляет собой множество элементов одного типа, для каждого из которых будет выполнено тело цикла. В качестве коллекции может выступать набор элементов, определенный пользователем (такие элементы заключаются в круглые скобки и перечисляются через точку с запятой), множество экземпляров сущностей и отношений модели, множество значений атрибутов и отношений экземпляра сущности, множество значений атрибутов и сущностей экземпляра отношения, значение атрибута типа данных "Массив":

```
коллекция = "@М.Сущности" | "@М.Отношения" |
 ("@С." Имя_сущности (".Атрибуты" | ".Отношения" |
 ".Имя_атрибута")) | ("@О." Имя_отношения (".Атрибуты" |
 ".Сущности" | ".Имя_атрибута")) | коллекция_пользователя.
коллекция_пользователя = "@("константа {";" константа} ")".
```

Параметр цикла value последовательно принимает значения всех элементов коллекции. Когда цикл начинает свою работу, параметру присваивается значение первого элемента, на следующей итерации — значение второго элемента и т.д. Для получения значения параметра цикла используется запись "@value".

Рассмотрим пример применения оператора цикла при создании правил трансформации. Пусть в рассмотренной ранее предметной области сущность *Компания* связана ассоциацией *Партнер* сама с собой (рис. 3). Данная ассоциация позволяет указать партнеров (поставщиков, клиентов) компании.

Требуется для каждой компании вывести список всех ее партнеров. Для этого необходимо построить правило со следующей правой частью:

```
Партнерами компании @С.Компания.Название являются
    @для value в @О.Партнер
@если @value.Источник.Название @= @С.Компания.Название
    @то @value.Приемник.Название @иначе
@если @value.Приемник.Название @= @С.Компания.Название
    @то @value.Источник.Название.
```

Таким образом, если название текущей компании совпадает с названием экземпляра сущности, который участвует в отношении Приемник, то в качестве партнера необходимо вывести название экземпляра сущности, находящегося на другом конце ассоциации.

Оператор присваивания позволяет задать значение атрибутам экземпляров сущностей и отношений либо параметра цикла *value*. Поскольку в процессе определения правила трансформации вида "модель—модель" может возникнуть необходимость присвоить значению атрибута экземпляра сущности/отношения экземпляр другой сущности/отношения (если атрибут имеет ссылочный тип), то в правой части

оператора присваивания может быть указано имя экземпляра сущности или отношения:

```
оператор _ присваивания = (значение _ атрибута |
                "@value") "@:="
(простое _ выражение | "@C." имя _ сущности |
                "@O." имя _ отношения).
```

Рассмотрим пример использования оператора присваивания при задании правила трансформации. Пусть в рассмотренной ранее предметной области сущность *Компания* также имеет еще один атрибута вещественного типа — *Выручка*, который должен вычисляться программно как разность дохода и расхода. Необходимо для каждого экземпляра сущности *Компания* вычислить значение атрибута *Выручка*. Для этого определим правило трансформации со следующей правой частью:

```
@C.Компания.Выручка @:=
@C.Компания.Доход @- @C.Компания.Расход.
```

Использование разработанного языка для описания трансформаций

Рассмотрим пример использования разработанного языка TDSL для описания трансформации моделей, созданных в нотации диаграмм классов UML, в код на языке SQL.

Упрощенная метамодель диаграмм классов UML представлена на рис. 4. Она содержит сущности *Класс* и *Атрибут*, а также три отношения ассоциации: *Ассоциация*, *Агрегация*, *Наследование*.

Сущность *Класс* имеет следующие атрибуты: *Имя*, *Поля*, *Методы*. Сущность *Атрибут* используется для описания типа полей класса. Она имеет три атрибута: *Имя*, *Тип*, *Значение*. Все отношения имеют такие атрибуты, как *Источник* (задает экземпляр сущности, из которого выходит экземпляр отношения) и

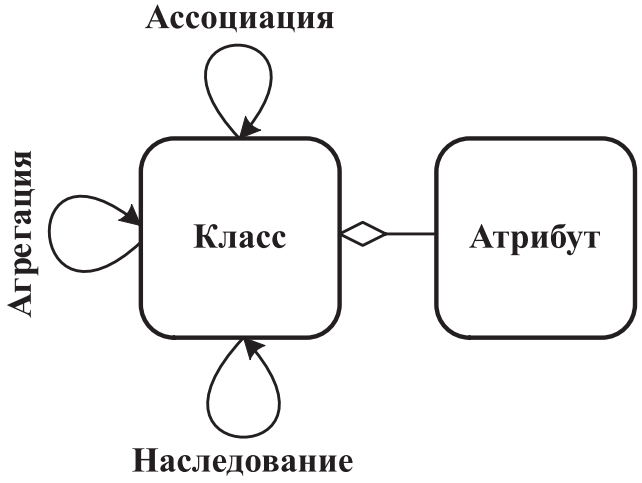

Рис. 4. Фрагмент метамодели диаграмм классов UML

Приемник (задает экземпляр сущности, в который входит экземпляр отношения). Отношения *Ассоциация* и *Агрегация* также имеют атрибут *Мощность*, который определяет мощность экземпляра ассоциации/агрегации.

Экземпляр класса в реляционной модели должен отображаться в таблицу, поля которой соответствуют атрибутам экземпляра класса.

Определим правило преобразования экземпляра класса в таблицу. Каждому полю класса должен соответствовать столбец создаваемой таблицы, поэтому правая часть правила примет следующий вид:

```
CREATE TABLE @C.Класс.Имя (id INTEGER primary key
@для value в @C.Класс.Поля, @value.Имя @value.Тип).
```

Отношения *Ассоциация* и *Агрегация* должны быть преобразованы в связь между таблицами. Причем, если мощность отношения один-ко-многим или многие-к-одному, и оно не имеет атрибутов, то для его преобразования необходимо добавить дополнительное поле в таблицу со стороны мощности один. Если же мощность отношения многие-ко-многим или один-к-одному, или оно имеет атрибуты, то такое отношение должно быть преобразовано в отдельную таблицу-связку, соединенную с таблицей-источником и таблицей-приемником. Таким образом, правая часть правила трансформации будет иметь следующий вид:

```
@если @O.Ассоциация.Мощность @= "1:M" @и
@O.Ассоциация.Атрибуты @= @null @то
@{
   ALTER TABLE @O.Ассоциация.Источник
   ADD @O.Ассоциация.Приемник @+ "ID" INTEGER
   ALTER TABLE @O.Ассоциация.Источник
   ADD FOREIGN KEY (@O.Ассоциация.Приемник @+ "ID")
   REFERENCES @O.Ассоциация.Приемник (id)
@}
@если @O.Ассоциация.Мощность @= "M:1" @и
@O.Ассоциация.Атрибуты @= @null @то
@{
   ALTER TABLE @O.Ассоциация.Приемник
   ADD @O.Ассоциация.Источник @+ "ID" INTEGER
   ALTER TABLE @O.Ассоциация.Приемник
   ADD FOREIGN KEY (@O.Ассоциация.Источник @+ "ID")
   REFERENCES @O.Ассоциация.Источник (id)
@}
@если @O.Ассоциация.Мощность @= "1:1" @или
@O.Ассоциация.Мощность @= "M:M" @или
@O.Ассоциация.Атрибуты @!= @null @то
@{
CREATE TABLE @O.Ассоциация.Источник @+
@O.Ассоциация.Приемник
(id INTEGER primary key, @O.Ассоциация.Источник
@+ "ID" INTEGER, @O.Ассоциация.Приемник @+ "ID" INTEGER)
ALTER TABLE @O.Ассоциация.Источник @+
@O.Ассоциация.Приемник
ADD FOREIGN KEY (@O.Ассоциация.Источник @+ "ID")
REFERENCES @O.Ассоциация.Источник (id)
ALTER TABLE @O.Ассоциация.Источник @+
@O.Ассоциация.Приемник
ADD FOREIGN KEY (@O.Ассоциация.Приемник @+ "ID")
REFERENCES @O.Ассоциация.Приемник (id)
@}
```

Для каждого экземпляра отношения *Наследование* необходимо в таблицу, соответствующую классу-потомку, добавить дополнительное поле — внешний ключ для связи с таблицей, соответствующей классу-родителю. В этом случае правая часть правила примет следующий вид:

```
ALTER TABLE @O.Наследование.Источник ADD
@O.Наследование.Приемник @+ "ID" INTEGER
ALTER TABLE @O.Наследование.Источник
ADD FOREIGN KEY (@O.Наследование.Приемник @+ "ID")
REFERENCES @O.Наследование.Приемник (id).
```

Заключение

Резюмируя представленные выше результаты исследований, можно констатировать, что разработанный текстовый предметно-ориентированный язык позволил устранить ограничения системы MetaLanguage, описанные в таблице. Язык TDSL позволяет выполнять арифметические операции над значениями атрибутов, навигацию по элементам моделей, ветвление потока управления, представляет возможность вызывать функции внешних библиотек. Перечисленные факторы существенно увеличивают выразительную мощность декларативных средств трансформации моделей. Несмотря на то что конструкции TDSL аналогичны соответствующим конструкциям языков высокого уровня, он не "загроможден" лишними элементами, что делает его простым в изучении и использовании для разных категорий пользователей.

Список литературы

1. **France R., Rumpe B.** Model-Driven Development of Complex Software: A Research Roadmap // Proceedings of the Workshop on the Future of Software Engineering. Washington: IEEE Computer Society, 2007. P. 37—54.

2. **Hutchinson J., Rouncefield M., Whittle J.** Model Driven Engineering Practices in Industry // Proceedings of the 33rd International Conference on Software Engineering. NY: ACM New York, 2011. P. 633—642.

3. **Литвинов Ю. В.** Методы и средства разработки графических предметно-ориентированных языков: дис. ... канд. техн. наук. Санкт-Петербург: СПбГУ, 2016. 193 с.

4. **Сухов А. О.** Классификация предметно-ориентированных языков и языковых инструментариев // Математика программных систем: межвузовский сборник научных статей. Пермь: Изд-во Пермского гос. нац. исслед. ун-та. 2012. № 9. С. 74—83.

5. **Терехов А. Н., Брыксин Т. А., Литвинов Ю. В.** QReal: платформа визуального предметно-ориентированного моделирования // Программная инженерия. 2013. № 6. С. 11—19.

6. **Кознов Д. В.** Программная инженерия и визуальное моделирование: воспитание культуры работы с информацией // Программная инженерия. 2015. № 10. С. 3—11.

7. **Атисков А. Ю.** Подходы к трансформации моделей проектирования информационных систем // Труды СПИИРАН. 2012. № 2 (21). С. 184—202.

8. **Sukhov A. O., Lyadova L. N.** Visual Models Transformation in MetaLanguage System // Proceedings of the 18th International Conference on Computers (part of CSCC'14). Vol. 2. Santorini Island: CSCC, 2014. P. 460—467.

9. **Cuong N. V.** Model Transformation Approach to Automated Model Driven Development: a thesis submitted in partial fulfillment of the requirements for the degree of Doctor. Prague: Czech Technical University in Prague, 2015. 122 p.

10. **Kalyanasundaram P., Ugale S. P.** Model Transformation: Concept, Current Trends and Challenges // International Journal of Computer Applications. 2015. Vol. 119, No. 14. P. 33—37.

11. **Сухов А. О.** Среда разработки визуальных предметно-ориентированных языков моделирования // Математика программных систем: межвузовский сборник научных статей. Пермь: Изд-во Пермского государственного университета. 2008. Вып. 5. С. 84—94.

12. **Лядова Л. Н., Сухов А. О.** Языковой инструментарий системы MetaLanguage // Математика программных систем: межвузовский сборник научных статей. Пермь: Изд-во Пермского гос. ун-та. 2008. № 5. С. 40—51.

13. **Ehrig H.** Fundamentals of Algebraic Graph Transformation. NY: Springer-Verlag, 2006. 388 p.

14. **Taentzer G.** AGG: A Graph Transformation Environment for Modeling and Validation of Software // Applications of Graph Transformations with Industrial Relevance. 2004. Vol. 3062/2004. P. 446—453.

15. **Balasubramanian D.** The Graph Rewriting and Transformation Language: GReAT // Electronic Communications of the EASST. 2006. Vol. 1. P. 1—8.

16. **Agrawal A., Karsai G., Neema S., Shi F., Vizhanyo A.** The Design of a Language for Model Transformations // Journal on Software and Systems Modeling. 2006. Vol. 5, No. 3. P. 261—288.

17. **Vizhanyo A., Neema S., Shi F., Balasubramanian D., Karsai G.** Improving the Usability of a Graph Transformation Language // Electronic Notes in Theoretical Computer Science. 2006. Vol. 152. P. 207—222.

18. **Varro D.** Automated Model Transformations for the Analysis of IT Systems: Ph.D. thesis. Budapest: Budapest University of Technology and Economics, 2003. 240 p.

19. **Varro D., Pataricza A.** VPM: A Visual, Precise and Multilevel Metamodeling Framework for Describing Mathematical Domains and UML // Journal of Software and Systems Modeling. 2003. Vol. 2. No. 3. P. 187—210.

20. **Кузнецов М. Б.** Трансформация UML-моделей и ее использование в технологии MDA // Программирование. 2007. Т. 33, № 1. С. 65—79.

21. **Markovic S., Baar T.** Semantics of OCL specified with QVT // Software and Systems Modeling. 2008. Vol. 7. P. 399—422.

22. **Stevens P.** Bidirectional Model Transformations in QVT: Semantic Issues and Open Questions // Model Driven Engineering Languages and Systems. 2007. Vol. 4735/2007. P. 1—15.

23. **Ступников С. А., Калиниченко Л. А.** Методы автоматизированного построения трансформаций информационных моделей // Системы и средства информатики. 2009. № 19. С. 34—62.

24. **Benelallam A.** Distributed Model-to-Model Transformation with ATL on MapReduce // Proceedings of 2015 ACM SIGPLAN International Conference on Software Language Engineering. NY: ACM New York, 2015. P. 37—48.

25. **Agirre J. A., Sagardui G., Etxeberria L.** Model Transformation by Example Driven ATL Transformation Rules Development Using Model Differences // Communications in Computer and Information Science. Springer, 2015. Vol 555. P. 113—130.

26. **Kappel G.** Model Transformation by-Example: a Survey of the First Wave // Conceptual Modelling and Its Theoretical Foundations. Berlin: Springer-Verlag, 2012. P. 197—215.

27. **Wimmer M.** Towards Model Transformation Generation By-Example // Proceedings of the 40th Annual Hawaii International Conference on System Sciences. Washington: IEEE Computer Society, 2007. P. 1—10.

28. **Сухов А. О.** Сравнение языков и инструментальных средств трансформации визуальных моделей // Математика программных систем: межвузовский сборник научных статей. Пермь: Изд-во Пермского гос. нац. исслед. ун-та. 2013. Вып. 10. С. 56—94.

Domain-Specific Language for Visual Models Transformation Creation

A. O. Sukhov, e-mail: ASuhov@hse.ru, National Research University Higher School of Economics, Perm, 614070, Russian Federation

Corresponding author:

Sukhov Alexander O., Associate Professor, National Research University Higher School of Economics, Perm, 614070, Russian Federation,
E-mail: ASuhov@hse.ru

Received on May 20, 2017
Accepted on June 14, 2017

Model-oriented approach to software development presumes the visual models usage at the process of software products creation. This approach can involve customer representatives, future users at the software development process. For automatic source code generation in the target language it is necessary to describe the rules for visual models transformation. There are various approaches to converting visual models: AGG, GReAT, VIATRA, QVT, ATL, MTBE, etc. However, they all make high requirements on the level of the user's skill, so their use is difficult. In addition, most of them use only text languages to describe transformations.

The language workbench MetaLanguage is a tool for developing visual domain-specific languages, creating models using these languages and transforming created models.

To increase the functionality of the transformation component of the MetaLanguage language workbench, a text-based domain-specific language was developed to describe visual model transformations into other graphical notations or textual representation. The language contains the conditional statement, the loop statement, the ability to specify arithmetic expressions over the values of the model elements attributes, and predefined function call. Despite the fact that TDSL's constructions are similar to the corresponding high-level languages constructions, the language is not "cluttered" with unnecessary elements, which makes it easy to use and understand for different categories of users. The language interpreter is implemented in the MetaLanguage tool.

Keywords: *domain-specific language, model transformation, horizontal transformation, visual models, model-oriented approach, language workbench, MetaLanguage system, multilevel modeling, metamodel, graph grammars*

For citation:

Sukhov A. O. Domain-Specific Language for Visual Models Transformation Creation**,** *Programmnaya Ingeneria,* 2017, vol. 8, no. 9, pp. 396—406.

DOI: 10.17587/prin.8.396-406

References

1. **France R., Rumpe B.** Model-Driven Development of Complex Software: A Research Roadmap, *Proceedings of the Workshop on the Future of Software Engineering*, Washington, IEEE Computer Society, 2007, pp. 37—54.

2. **Hutchinson J., Rouncefield M., Whittle J.** Model Driven Engineering Practices in Industry, *Proceedings of the 33rd International Conference on Software Engineering*, NY, ACM New York, 2011, pp. 633—642.

3. **Litvinov Ju. V.** Metody i sredstva razrabotki graficheskih predmetno-orientirovannyh jazykov (Graphical Domain-specific Languages Development Methods and Tools): dis. ... kand. tehn. nauk, Saint-Petersburg, SPbGU, 2016, 193 p. (in Russian).

4. **Suhov A. O.** Klassifikacija predmetno-orientirovannyh jazykov i jazykovyh instrumentariev (Domain-specific Languages and Language Workbench Classification), *Matematika programmnyh sistem: mezhvuzovskij sbornik nauchnyh statej*, Perm, Izd-vo Permskogo gosudarstvennogo nacionalnogo issledovatelskogo universiteta, 2012, no. 9, pp. 74—83 (in Russian).

5. **Terehov A. N., Bryksin T. A., Litvinov Ju. V.** QReal: platforma vizualnogo predmetno-orientirovannogo modelirovanija (QReal: Visual Domain-specific Modeling Platform), *Programmnaya Ingeneria*, 2013, no. 6, pp. 11—19 (in Russian).

6. **Koznov D. V.** Programmnaja inzhenerija i vizualnoe modelirovanie: vospitanie kultury raboty s informaciej (Software Engineering and Visual Modeling: Training of the Working with Information Culture), *Programmnaya Ingeneria*, 2015, no. 10, pp. 3—11 (in Russian).

7. **Atiskov A. Ju.** Podhody k transformacii modelej proektirovanija informacionnyh system (The Approaches to Information Systems Design Models Transformation), *Trudy SPIIRAN*, 2012, no. 2 (21), pp. 184—202 (in Russian).

8. **Sukhov A. O., Lyadova L. N.** Visual Models Transformation in MetaLanguage System, *Proceedings of the 18th International Conference on Computers (part of CSCC'14)*, Vol. 2. Santorini Island, CSCC, 2014, pp. 460—467.

9. **Cuong N. V.** Model Transformation Approach to Automated Model Driven Development: a thesis submitted in partial fulfillment

of the requirements for the degree of Doctor, Prague: Czech Technical University in Prague, 2015. 122 p.

10. **Kalyanasundaram P., Ugale S. P.** Model Transformation: Concept, Current Trends and Challenges, *International Journal of Computer Applications*, 2015, vol. 119, no. 14, pp. 33—37.

11. **Suhov A. O.** Sreda razrabotki vizualnyh predmetno-orientirovannyh jazykov modelirovanija (Visual Domain-specific Modeling Languages Development Tool), *Matematika programmnyh sistem: mezhvuzovskij sbornik nauchnyh statej*, Perm, Izd-vo Permskogo gosudarstvennogo universiteta, 2008, no. 5, pp. 84—94 (in Russian).

12. **Ljadova L. N., Suhov A. O.** Jazykovoj instrumentarij sistemy MetaLanguage (Language Workbench of the MetaLanguage System), *Matematika programmnyh sistem: mezhvuzovskij sbornik nauchnyh statej*, Perm, Izd-vo Permskogo gosudarstvennogo universiteta, 2008, no. 5, pp. 40—51 (in Russian).

13. **Ehrig H.** Fundamentals of Algebraic Graph Transformation, NY, Springer-Verlag, 2006, 388 p.

14. **Taentzer G.** AGG: A Graph Transformation Environment for Modeling and Validation of Software, *Applications of Graph Transformations with Industrial Relevance*, 2004, vol. 3062/2004, pp. 446—453.

15. **Balasubramanian D.** The Graph Rewriting and Transformation Language: GReAT, *Electronic Communications of the EASST*, 2006, vol. 1, pp. 1—8.

16. **Agrawal A., Karsai G., Neema S., Shi F., Vizhanyo A.** The Design of a Language for Model Transformations, *Journal on Software and Systems Modeling*, 2006, vol. 5, no. 3, pp. 261—288.

17. **Vizhanyo A., Neema S., Shi F., Balasubramanian D., Karsai G.** Improving the Usability of a Graph Transformation Language, *Electronic Notes in Theoretical Computer Science*, 2006, vol. 152, pp. 207—222.

18. **Varro D.** Automated Model Transformations for the Analysis of IT Systems: Ph.D. thesis. Budapest, Budapest University of Technology and Economics, 2003. 240 p.

19. **Varro D., Pataricza A.** VPM: A Visual, Precise and Multilevel Metamodeling Framework for Describing Mathematical Domains and UML, *Journal of Software and Systems Modeling*, 2003, vol. 2, no. 3, pp. 187—210.

20. **Kuznecov M. B.** Transformacija UML-modelej i ee ispolzovanie v tehnologii MDA (Transformation of UML-models and Its Use in MDA Technology), *Programmirovanie*, 2007, vol. 33, no. 1, pp. 65—79 (in Russian).

21. **Markovic S., Baar T.** Semantics of OCL specified with QVT, *Software and Systems Modeling*, 2008, vol. 7, pp. 399—422.

22. **Stevens P.** Bidirectional Model Transformations in QVT: Semantic Issues and Open Questions, *Model Driven Engineering Languages and Systems*, 2007, vol. 4735/2007, pp. 1—15.

23. **Stupnikov S. A., Kalinichenko L. A.** Metody avtomatizirovannogo postroenija transformacij informacionnyh modelej (Methods of Automated Creation of Information Model Transformations), *Sistemy i sredstva informatiki*, 2009, no. 19, pp. 34—62 (in Russian).

24. **Benelallam A.** Distributed Model-to-Model Transformation with ATL on MapReduce, *Proceedings of 2015 ACM SIGPLAN International Conference on Software Language Engineering*, NY, ACM New York, 2015, pp. 37—48.

25. **Agirre J. A., Sagardui G., Etxeberria L.** Model Transformation by Example Driven ATL Transformation Rules Development Using Model Differences, *Communications in Computer and Information Science*, Springer, 2015, vol. 555, pp. 113—130.

26. **Kappel G.** Model Transformation by-Example: a Survey of the First Wave, *Conceptual Modelling and Its Theoretical Foundations*, Berlin, Springer-Verlag, 2012, pp. 197—215.

27. **Wimmer M.** Towards Model Transformation Generation By-Example, *Proceedings of the 40th Annual Hawaii International Conference on System Sciences*, Washington, IEEE Computer Society, 2007, pp. 1—10.

28. **Suhov A. O.** Sravnenie jazykov i instrumentalnyh sredstv transformacii vizualnyh modelej (Comparison of Visual Models Transformation Languages and Tools), *Matematika programmnyh sistem: mezhvuzovskij sbornik nauchnyh statej*, Perm, Izd-vo Permskogo gosudarstvennogo nacionalnogo issledovatelskogo universiteta, 2013, no. 10, pp. 56—94 (in Russian).

ИНФОРМАЦИЯ

УДК 517.9, 004.9 DOI: 10.17587/prin.8.407-412

Н. Н. Светушков, канд. техн. наук, доц., e-mail: svetushkov@mai.ru,
Московский авиационный институт (национальный исследовательский университет),
Москва, Россия

Программные механизмы для моделирования процессов теплопередачи в геометрически сложных изделиях

Дано краткое описание функциональных возможностей разработанной автором программной среды на языке Visual C++, предназначенной для решения нестационарных задач теплопроводности. Для создания геометрических моделей использовался принцип кластерного объединения, позволяющий пользователю с помощью интуитивно понятных действий создавать довольно сложные двумерные фигуры. Представленное описание включает в себя собственно метод кластерного объединения и его программную реализацию. Для решения задачи теплопроводности использовался подход, основанный на интегральных уравнениях, описывающих процесс теплопередачи. Его использование позволило контролировать точность расчетов в каждой точке сеточного разбиения. В статье продемонстрированы возможности программной среды по заданию неоднородных материалов, имеющих как регулярное, так и нерегулярное строение. Результаты расчетов могут быть получены в виде числовых значений, в виде цветовых карт, а также как трехмерные поверхности, что позволяет получить наглядное представление о функции распределения температур на каждом временном слое и провести анализ критически важных тепловых процессов.

Ключевые слова: *программная среда, визуализация, сложные геометрические объекты, моделирование, уравнения теплопроводности, интегральные уравнения, численные методы, цветовые карты*

Введение

Задача расчета температурных полей в геометрически сложных изделиях при различных условиях внешнего теплового воздействия достаточно часто встречается в различных отраслях промышленности. К их числу относятся материаловедение, создание различных технических изделий для работы в экстремальных условиях и др. Отдельной, востребованной на практике задачей в связи с этим является расчет термических напряжений, возникающих в техническом изделии [1, 2]. При решении этой задачи можно выделить два основных вопроса, которые требуют разрешения. Первый из них связан с геометрическим представлением сложного изделия, а именно созданием его модели и с заданием начальных и граничных условий для задачи теплопроводности. Второй вопрос — выбор численного метода, обеспечивающего достаточно высокую скорость сходимости и требуемую точность результатов при численном решении поставленной задачи [3, 4].

Создание сложных геометрических моделей само по себе является непростой задачей даже в двумерном случае, и для ее решения требуется разрабатывать специальный программный интерфейс с необходимым набором функциональных возможностей. Даже использование простейших стандартных геометрических фигур приводит к необходимости при их взаимном расположении учитывать области контакта, обеспечить возможности выбора их ориентации и масштабирования каждой фигуры, а также ряд других функций. Неслучайно, что как в коммерческом, так и в свободном доступе существует большое число графических редакторов с различными функциональными возможностями. Применение таких редакторов эффективно и оправдано в одних случаях, но представляет значительные трудности в других [5].

В настоящей статье описана разработанная автором программная среда, в которой для построения двумерных геометрических моделей применялся новый подход, основанный на принципах кластерного объединения [6]. Программа написана на языке Visual C++, с подключенными библиотеками MFC. Она позволяет конечному пользователю простыми средствами создавать довольно сложные геометрические объекты. При этом поддерживаются механизмы автоматического выделения границ и предоставляется возможность задания на них различных граничных условий.

Из различных потенциально возможных методов численного решения задачи теплопроводности в разработанной программе был использован метод, основанный на интегральном описании процессов теплопередачи [7—9]. Такой подход обеспечивает достаточно высокую скорость сходимости, имеет потенциальные возможности для распараллеливания алгоритма решения задачи, а также, что является существенным, позволяет контролировать точность получаемых численных результатов в каждой точке сеточного разбиения.

Принципы кластерного объединения при создании двумерных геометрических объектов

Для того чтобы упростить программный интерфейс в части создания сложных двумерных моделей был реализован подход, основанный на возможности автоматического объединения указанных пользователем элементов [6]. Для создания геометрического объекта пользователю необходимо отметить лишь определенное число отображаемых на экране точек, а собственно объект будет сгенерирован программными средствами. На рис. 1 (см. вторую сторону обложки) показана пустая сеточная область модели, выбором точек из которой формируется определенная геометрическая фигура, причем не обязательно относящаяся к классическим.

Для того чтобы пояснить, как эти фигуры сгенерированы, рассмотрим более подробно принципы так называемого кластерного объединения на примере анализа четырех точек (кластерных элементов), расположенных в виде четырехугольника (рис. 2).

Пользователь с помощью указателя "мышь" может выбрать одну или несколько кластерных точек, входящих в состав данного четырехугольника. Если выбрана одна точка, то на экране отобразится фигура в виде квадрата (рис. 3).

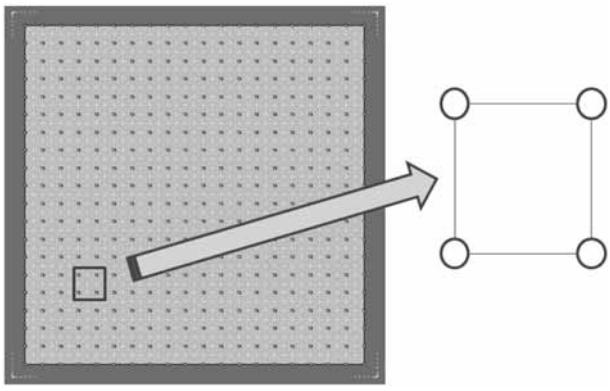

Рис. 2. Элементарный четырехугольник, на основе которого формируется общая фигура во всей сеточной области

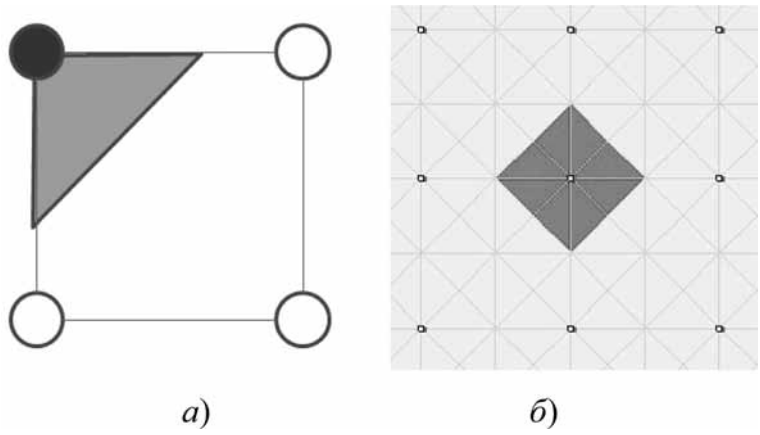

Рис. 3. Выбор единственной точки:

а — пользователь отметил одну точку в элементарном четырехугольнике; *б* — программа построила четыре треугольника — по одному для каждой элементарной области

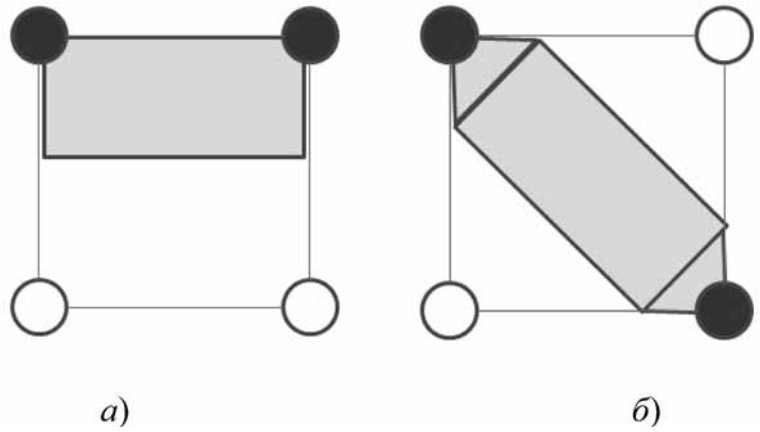

Рис. 4. Выбор двух точек в четырехугольнике:

а — выбраны две точки, расположенные на одном ребре; *б* — выбраны две точки, расположенные на диагонали

Программный анализатор построен таким образом, что вид построенной фигуры зависит только от числа выбранных пользователем точек. Такой подход сокращает общее число действий пользователя и позволяет получать достаточно сложные геометрические объекты.

На рис. 4 показаны элементарные фигуры в случае, когда пользователем выбраны две точки в элементарном четырехугольнике. На рис. 5 показаны фигуры, сформированные в программной среде на экране монитора.

Если пользователь выбрал три точки из четырехугольника, то формируется область, вид который является симметричным отражением ситуации в случае выбора одной точки. На рис. 6 приведены простейшие трехточечная и четырехточечная модели.

Реализованный в программной среде подход для создания двумерных моделей имеет интуитивно понятную основу. Он позволяет пользователю не заботиться о выборе геометрии классической фигуры и

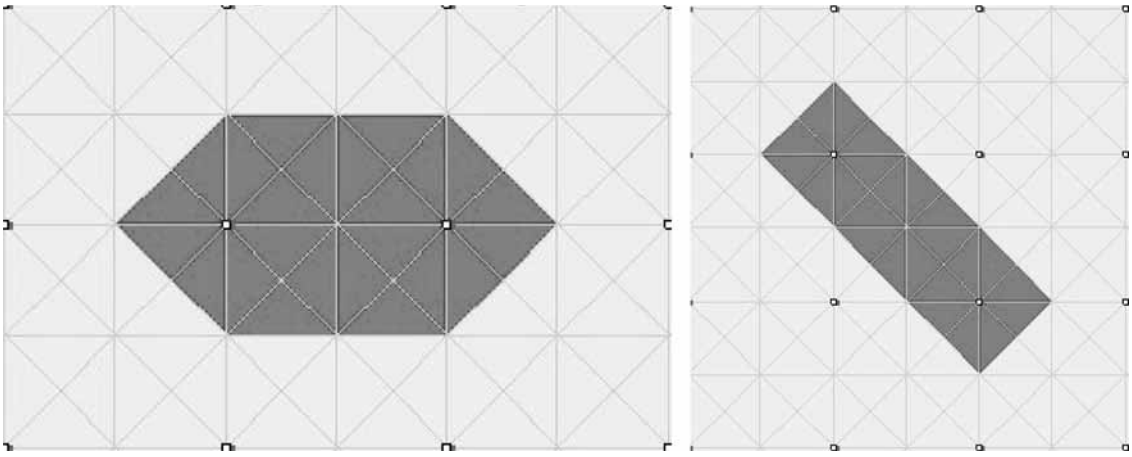

Рис. 5. Простейшие двухточечные фигуры, автоматически сформированные в разработанном программном комплексе

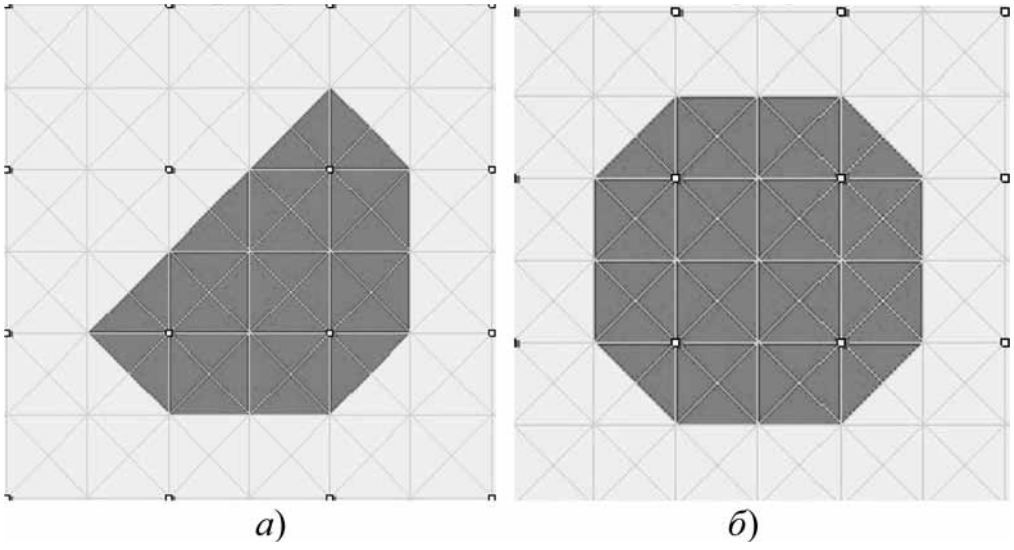

Рис. 6. Трехточечная (*а*) и четырехточечная (*б*) модели, построенные на принципах кластерного объединения

перебрать возможные варианты или отметить входящие в модель точки, чтобы получить геометрическую фигуру, достаточно хорошо описывающую исходную модель. На рис. 7 (см. вторую сторону обложки) приведены примеры сложных геометрических фигур, которые могут быть созданы неподготовленным пользователем в течение нескольких минут. Создание таких фигур другими способами представляет собой непростую задачу.

В программном комплексе для пользователя предусмотрена возможность выбора одного из инструментальных средств рисования с целью выделения или отмены выделения определенных точек — карандаш, ластик, заливка, прямоугольник, круг и др. Все эти дополнительные средства позволяют еще больше упростить процесс создания сложных геометрических фигур.

Задание теплофизических характеристик модели

В разработанной программной среде, кроме создания собственно модели изделия, предусмотрена возможность задания теплофизических характеристик не только для всего изделия целиком, но и для отдельных его частей. На рис. 8 (см. вторую сторону обложки) показана модель в виде обычного параллелепипеда, теплофизические характеристики которой отражают ее слоистую внутреннюю структуру.

В программе предусмотрена возможность связать определенный цвет с названием материала и с его теплофизическими характеристиками с помощью диалогового окна из библиотеки MFC. На рис. 9 (см. третью сторону обложки) показано

окно, которое открывается при нажатии левой клавиши мыши на выбранном цвете и с помощью которого можно задать теплоемкость, коэффициенты теплопроводности по осям и название материала.

Формирование геометрии внутреннего наполнителя или слоя в данном случае также происходит в соответствии с положениями кластерного объединения. В рамках такого подхода выделяются базовые четырехугольники и анализируется цвет входящих в них вершин. Затем используются стандартные правила объединения и построения элементарных фигур, описанные в предыдущем разделе.

Задание начальных условий для нестационарной задачи теплопроводности

Для задания функции распределения начальной температуры в программе предусмотрено отдельное окно, которое вызывается путем выбора соответствующего пункта меню из основного диалогового окна (рис. 10, см. третью сторону обложки).

Правило, по которому строится распределение температур в начальный момент времени, также основано на возможности пользователя задавать с помощью диалогового окна значение температуры в выбранной точке. Кроме этого, здесь предусмотрена цветовая индикация от темно-синего до ярко-красного цвета, позволяющая более наглядно визуализировать введенные значения. Температурный диапазон, отражающий минимальное и максимальное значения, может быть изменен пользователем. При этом меняется и вся цветовая карта, соответствующая заданному полю температур. В программе предусмотрена возможность одновременно отобразить на экране все введенные данные (рис. 11, см. третью сторону обложки).

Задание граничных условий для нестационарной задачи теплопроводности

При задании граничных условий пользователь может выбирать из двух типов условий: граничные условия первого рода, когда на поверхности задается температура; и граничные условия второго рода, когда на поверхности задаются тепловые потоки (возможно, зависящие от температуры). Важной особенностью разработанного программного обеспечения является поддержка механизмов, предоставляющих возможность задавать одновременно на одной части границы условия первого рода, а на остальной — условия второго рода. Здесь также предусмотрена цветовая индикация и диалоговые окна для внесения конкретных значений для температуры и тепловых потоков. Важно отметить, что для задания изменяющихся во времени граничных условий предусмотрена возможность перехода к определенному временному слою с помощью клавиш PgDn и PgUp, а также по введенному номеру слоя. После этого можно или внести новые значения, или дублировать данные с другого слоя, или тиражировать текущий слой в другие и т. д. Таким образом, пользователю предоставляется возможность достаточно гибко задавать различные типы граничных условий, в том числе, меняющиеся с течением времени (рис. 12, см. четвертую сторону обложки).

Отображение результатов расчета в виде трехмерных карт

В процессе расчета на каждом временном слое рассчитывается распределение температур с учетом граничных условий, которые можно просматривать в интерактивном режиме. Для "листания" предназначены клавиши PgDn, PgUp, а также предусмотрена возможность перехода по номеру выбранного слоя, который указывается в специальном диалоговом окне.

Для удобства анализа получаемых результатов предусмотрена как цветовая индикация функции температуры (с заданными нижним и верхним пределами), так и непосредственный вывод значений функции температуры в базовых точках и визуализация полученных результатов в виде трехмерной карты (рис. 13, см. четвертую сторону обложки).

Данное окно позволяет пользователю не обращаться к сторонним программам для построения графиков по полученным результатам и непосредственно в процессе работы подготовить необходимые рисунки для использования в технических отчетах.

Заключение

Представленный программный комплекс, созданный на языке Visual C++, обладает функциональными возможностями, достаточными для постановки и решения нестационарных задач теплопроводности в двумерном случае. Проведенные тестовые расчеты показали устойчивость используемого алгоритма. Полученные оценки точности решения в среднем совпадали с оценками для известных задач, имеющих аналитическое решение, например, остывание цилиндра или прогрев полубесконечной среды.

Подход, использованный при формировании геометрических моделей, который назван кластерным объединением [6], имеет интуитивно понятную основу и обладает достаточно гибкими возможностями при создании сложных объектов. Программные средства просмотра полученных результатов моделирования позволяют сформировать ясное понимание происходящих тепловых процессов в неоднородных материалах и изделиях, имеющих как регулярную, так и нерегулярную структуру.

Можно констатировать, что, несмотря на простой интерфейс, созданная программная среда является удобным и надежным средством как для решения инженерных задач, возникающих в различных технических областях, так и для дополнительного анализа получаемых в ходе исследований числовых результатов.

Список литературы

1. **Зарубин В. С.** Прикладные задачи термопрочности элементов конструкций. М.: Машиностроение, 1985. 296 с.
2. **Зарубин В. С.** Математическое моделирование в технике (3-е издание). М.: Изд-во МГТУ им. Н. Э. Баумана, 2009. 495 с.
3. **Пасконов В. М., Полежаев В. И., Чудов Л. А.** Численные методы в задачах тепло- и массобмена. М.: Наука, 1984. 288 с.
4. **Minkowycz W. J., Sparrow E. H., Schneider C. E., Plechter R. H.** Handbook of numerical heat transfer, New York: John Wiley & Sons, 1988. 1024 p.
5. **Большаков В. П., Бочков А. Л.** Основы 3D-моделирования (учебный курс). Санкт-Петербург: Питер, 2012. 304 с.
6. **Светушков Н. Н.** Модель объединенного кластера в трехмерной графике // Программная инженерия. 2014. № 7. С. 40—43.
7. **Светушков Н. Н.** Метод струн в задачах многомерной нестационарной теплопроводности // Информационные технологии. 2014. № 12. С. 14—19.
8. **Светушков Н. Н.** Параллельный метод струн для численного решения нелинейных задач теплопроводности // Информационные технологии. 2015. Т. 21, № 9. С. 689—693.
9. **Svetushkov N. N.** Iterative method for the numerical solution of a system of integral equations for the heat conduction initial boundary value problem//11th International Conference on "Mesh methods for boundary-value problems and applications" IOP Publishing IOP Conf. Series: Materials Science and Engineering. 2016. Vol. 158, N. 1, URL: http://iopscience.iop.org/1757-899X/158/1/012091

Software Facilities for Heat Transfer Modeling in Geometrically Complex Objects

N. N. Svetushkov, svetushkov@mai.ru, Moscow Aviation Institute (National Research University), Moscow, 125993, Russian Federation

Corresponding author:

Svetushkov Nikolaj N., Associate Professor, Moscow Aviation Institute (National Research University), Moscow, 125993, Russian Federation,
E-mail: svetushkov@mai.ru

Received on June 14, 2017
Accepted on June 28, 2017

The article contains a brief description of the developed software environment in the Visual C++ language intended for solving non-stationary heat conduction problems. To create geometric models, the cluster consolidation principle was used, which allows the user to create complex two-dimensional figures using intuitive operations. The presented description includes the actual method of clustering and its software implementation. To solve the heat conduction problem, we used an approach based on integral equations describing the heat transfer process. Its use made it possible to control the accuracy of calculations at each grid point. The paper demonstrates the capabilities of the software environment for assigning heterogeneous materials that have both a regular and an irregular structure. The results of the calculations can be represented both as numerical values and in the form of color maps, and also as three-dimensional surfaces, which makes it possible to get a visual representation of the temperature distribution function at each time layer and analyze critical thermal processes.

Keywords: *software environment, visualization, complex geometric objects, modeling, heat equation, integral equations, numerical methods, color maps*

For citation:

Svetushkov N. N. Software Facilities for Heat Transfer Modeling in Geometrically Complex Objects, *Programmnaya Ingeneria,* 2017, vol. 8, no. 9, pp. 407—412.

DOI: 10.17587/prin.8.407-412

References

1. **Zarubin V. S.** *Prikladnye zadachi termoprochnosti jelementov konstrukcij* (Applied problems of thermal strength of elements in constructions), Moscow, Mashinostroenie, 1985, 296 p. (in Russian).

2. 2.**Zarubin V. S.** *Matematicheskoe modelirovanie v tehnike* (Mathematical modeling in engineering), Moscow, Izd-vo MGTU im. N. Je. Baumana, 2009, 495 p. (in Russian).

3. **Paskonov V. M., Polezhaev V. I., Chudov L. A.** *Chislennye metody v zadachah teplo- i massobmena* (Numerical methods in heat and mass transfer problems), Moscow, Nauka, 1984, 288 p. (in Russian).

4. **Minkowycz W. J., Sparrow E. H., Schneider C. E., Plechter R. H.** *Handbook of numerical heat transfer,* New York, John Wiley & Sons, 1988, 1024 p.

5. **Bol'shakov V. P., Bochkov A. L.** *Osnovy 3D-modelirovanija, uchebnyj kurs* (Basics of 3D modeling, training course), St. Petersburg, Piter, 2012, 304 p. (in Russian).

6. **Svetushkov N. N.** Model' obedinennogo klastera v trehmernoj grafike (Joint cluster model in three-dimensional graphics), *Programmnaya Ingeneria,* 2014, no. 7, pp. 40—43 (in Russian).

7. **Svetushkov N. N.** Metod strun v zadachah mnogomernoj nestacionarnoj teploprovodnosti (The strings method in multidimensional unsteady heat conduction problems), *Informacionnye Tehnologii,* 2014, no. 12, pp. 14—19 (in Russian).

8. **Svetushkov N. N.** Parallel'nyj metod strun dlja chislennogo reshenija nelinejnyh zadach teploprovodnosti (Parallel strings method for the numerical solution of nonlinear heat conduction problem), *Informacionnye Tehnologii,* 2015, vol. 21, no. 12, pp. 689—693 (in Russian).

9. **Svetushkov N. N.** Iterative method for the numerical solution of a system of integral equations for the heat conduction initial boundary value problem, *Proc. of 11th International Conference on "Mesh methods for boundary-value problems and applications" IOP Publishing IOP Conf. Series: Materials Science and Engineering,* 2016, vol. 158, no. 1, available at: http://iopscience.iop.org/1757-899X/158/1/012091

ИНФОРМАЦИЯ

10—13 октября 2017 г.
Москва, МГУ им. М. В. Ломоносова
XIX Международная конференция

АНАЛИТИКА И УПРАВЛЕНИЕ ДАННЫМИ В ОБЛАСТЯХ С ИНТЕНСИВНЫМ ИСПОЛЬЗОВАНИЕМ ДАННЫХ (DAMDID/RCDL'2017)

Тематика конференции

Треки сферы анализа данных, решения задач, организации экспериментов в DID

- Особенности данных в DID
- Постановки и решение задач в DID
- Организация экспериментов в DID
- Гипотезы и модели (как составная часть исследовательских экспериментов в DID)
- Развитые методы и процедуры анализа данных в DID
- Концептуальное моделирование предметных областей в DID
- Применения анализа с интенсивным использованием данных в исследованиях в DID

Треки сферы управления данными в DID

- Интеграция данных в DID
- Извлечение данных из текстов
- Исследовательские инфраструктуры данных и их применение в DID
- Роль Семантического Веба в DID

Подробную информацию о конференции см. на сайте:

http://damdid2017.frccsc.ru/

УДК 004.75 DOI: 10.17587/prin.8.413-421

Е. А. Курако, науч. сотр., e-mail: kea@ipu.ru, **В. Л. Орлов,** канд. техн. наук, вед. науч. сотр., e-mail: ovl@ipu.ru, Федеральное государственное бюджетное учреждение науки Институт проблем управления им. В. А. Трапезникова РАН, г. Москва

Сервис-браузеры для информационных систем

Рассмотрено использование различных типов клиентов в информационных системах. Введено понятие сервис-браузера, который взаимодействует с сервером путем вызова сервисов различных типов и отображения ответов. Дана общая схема проектирования сервис-браузеров. Проведены эксперименты для сравнения сервис-браузеров типа REST и SOAP с традиционным HTML-браузером.

Ключевые слова: *информационная система, web-клиент, браузер, сервис-браузер, REST, SOAP, HTML*

Введение

В настоящее время проблема проектирования и разработки информационных систем в части взаимодействия с пользователем стоит очень остро. Если раньше на серверах в основном размещались данные, а вся обработка шла на клиенте, который получил название "толстый" клиент, то сейчас наблюдается явная тенденция перемещения всего объема вычислений на сервер. При этом функционально компьютер-клиент становится похож на терминал мейнфреймов. Это, безусловно, удобно ввиду того, что не возникает необходимость устанавливать программное обеспечение клиента на множество рабочих станций. Достаточно чтобы на каждой из них был установлен универсальный web-клиент, например, браузер MS Internet Explorer, оператор которого по заданному URL инициировал бы обращение к серверу. Сервер в соответствии с URL отправляет клиенту HTML-страницу, содержащую формы, изображения, тексты и поля ввода, с которыми работает оператор.

Основные достоинства и недостатки, выявляющиеся при использовании "толстого" клиента и web-клиента, приведены в таблице.

"Толстый" клиент получает от сервера и возвращает только данные, подлежащие обработке. Основной контроль данных осуществляется на месте. Для ввода могут использоваться сложноорганизованные формы, если это удобно для оператора и процесса проведения контроля. Но за это приходится платить индивидуальной установкой клиента на каждом рабочем месте, что создает организационные проблемы. Нужно также иметь в виду то обстоятельство, что сопровождение здесь усложняется, так как при модернизации продукта необходимо проводить обновление на всех рабочих местах. Кроме того, если изменения должны проводиться на клиенте и на сервере одновременно, то это означает, что при обновлении рабочих мест нужно обеспечивать блокировку.

Все эти недостатки исчезают при переходе на web-клиент. Действительно, здесь зачастую не требуется установка клиентского программного обеспечения, достаточно только сообщить тройку (URL, а затем логин и пароль) для начала работы. Также отпадают вопросы с обновлением рабочих мест, так как все изменения происходят на сервере. Вместе с тем нужно учитывать то обстоятельство, что по сети передаются не только данные, но и вспомогательная информация для отображения, т. е. общий объем передаваемой информации возрастает. Для того чтобы координировать ввод в разные поля коррелирующего содержимого, что часто необходимо, обычно используется JavaScript. Текст JavaScript включается в HTML-страницу, следовательно, перемещается по сети и исполняется на клиенте в режиме интерпретации. Классические web-информационные системы с увеличением кода на JavaScript трансформируются в подобие "толстого" клиента.

Построение современных информационных систем невозможно без криптозащиты. Но соответству-

Сравнение "толстого" клиента и web-клиента

Клиент	Достоинства	Недостатки
"Толстый" клиент	• Быстрая обработка данных • Возможность использования сложных форм	• Сложность установки • Трудности сопровождения и обновления
Web-клиент	• Простота установки • Нет необходимости сопровождения клиентов	• Большой объем информации, передаваемой по сети • Необходимость дополнительных расширений браузера для криптозащиты

ющие функции либо должны быть встроены в браузер, либо включаться в его расширения. Это требует разработки и установки, т. е. "утолщает" клиентскую часть. Можно также использовать технологию ActiveX, но здесь возникает необходимость регистрации соответствующего компонента на каждом клиенте.

Таким образом, каждый подход не лишен отрицательных моментов. Поэтому было бы целесообразно ориентироваться на такого клиента, который интегрировал бы положительные стороны имеющихся решений и не повторял, по возможности, их недостатки. В настоящей статье рассмотрены основные методы построения подобного клиента, при этом необходимо решение следующих задач:

- клиент должен быть универсальным для различных информационных систем, но по линии сервер — клиент в рабочем режиме должны передаваться только данные;
- клиент должен уметь обновляться в автоматическом режиме;
- средства защиты универсального клиента должны быть идентичными для различных информационных систем.

Построение универсального клиента для обмена данными с сервером

Для решения этой задачи, прежде всего, следует определиться со способами передачи данных. На современном этапе в качестве такого транспортного средства целесообразно использовать электронные сервисы.

Вообще говоря, сервисы могут быть различных типов. Более того, каждый разработчик может придумать свой собственный сервис. Для этого достаточно только описать типы форматов данных, обрабатываемых этим сервисом, и способы транспортировки данных от клиента к сервису и в обратном направлении. При этом нет необходимости отказываться от надежно работающего протокола TCP/IP. Можно использовать на серверном пространстве web-сервер и организовать связь с ним на базе протокола HTTP, тогда многие вопросы будут уже решены. В группе сервисов, использующих этот подход, можно выделить два: SOAP-сервис и REST-сервис, которые широко распространены.

Спецификации SOAP-сервисов, часто называемых web-сервисами, разработаны фирмой Microsoft [1]. Архитектурный стиль REST (*Representational state transfer*) представлен в 2000 г. Роем Филдингом [2]. На нем базируется понятие REST-сервиса. Как тот, так и другой сервисы используют web-сервер и протокол HTTP в качестве базового (хотя теоретически существует и возможность использования других протоколов). Разница заключается в прослойке между системными программами, реализующими на сервере HTTP-протокол, и прикладными программами, написанными разработчиком информационной системы. Важным достоинством SOAP-сервиса, которое обеспечивается данной прослойкой, является самодокументируемость. Действительно, сервис позволяет вызвать WSDL-структуру, в которой приводится подробное описание методов сервиса. Это обстоятельство обеспечивает возможность оперативного и корректного подключения к сервису различных клиентов. REST-сервис не имеет таких средств, но гибкость использования дает ему определенные преимущества.

Отсюда следует, что универсальный браузер для передачи данных должен иметь аппарат вызова сервисов по крайней мере одного типа. То есть в отличие от обычного браузера, который ориентирован на запросы и получение HTML-страниц, предлагаемый браузер должен передавать вызовы сервисов и получать в ответ данные. Поэтому в дальнейшем будем называть его *сервис-браузером*. Следует отметить, что сервис-браузер в силу указанных особенностей (работы только с сервисами) представляет собой отдельный программный продукт.

Динамически обновляемые модули в сервис-браузере

В сервис-браузере данные передаются клиенту через сервисы. Для передачи форм, изображений и программ, которые обрабатывают действия оператора, возможно использование двух способов:

1) передача информации для отображения и скриптов вместе с данными;

2) предварительная установка всех форм и программ заранее на каждую клиентскую машину.

Нетрудно заметить, что в первом случае идет возвращение к классическому web-браузеру, хотя структура его организации будет несколько иной. Но в силу распространенности и отлаженности web-браузеров, имеющих немалую историю, вряд ли на этом пути можно будет получить сколь-либо значительный эффект.

Во втором случае реализуется вариант "толстого" клиента, о недостатках которого говорилось выше. И при этом, что важно, полностью теряется универсальность клиента.

Возможен третий путь, заключающийся в использовании все более широко применяющегося механизма обновления. Однако обычно обновление — это весьма сложная процедура, требующая затрат как времени, так и ресурсов. Вместе с тем достаточно организовать модульную структуру, где каждый модуль представляется отдельным файлом сравнительно небольшого объема, содержащим программы и элементы дизайна, делать короткие проверки (например, сравнение версий модуля на сервере и на клиенте) и обновлять соответствующие модули при запуске сервис-браузера. Иными словами, можно использовать метод динамически обновляемых модулей [3]. Опыт применения этого

метода для информационных систем говорит о том, что обычно оператор практически не замечает процедуру обновления при перезапуске клиента, а сам перезапуск он проводит по крайней мере раз в сутки или чаще. Следует отметить, что метод динамически обновляемых модулей в качестве инструмента для обновления использует сервисы, что дает возможность применять его в системах, где прямой доступ к файлам запрещен [3, 4]. Однако в сервис-браузер этот метод встраивается естественно.

Следует также выделить еще один важный момент. Программное обеспечение меняется не только на клиенте, но и на сервере. Для синхронизации изменений возможно применение нескольких методов. Если изменения на сервере не требуют корректировки клиента (например, исправлена обработка какой-либо ошибки), то клиент не ставится в известность, а начинает работать с новым сервисом после его активизации. Если сервис меняется более радикально и это изменение требует изменения клиента, то новая версия сервиса публикуется под другим именем и обновившийся клиент начинает работать с вновь опубликованной версией сервиса. Не обновленные по каким-либо причинам "старые" клиенты в это время продолжают работать со "старой" версией сервиса (метод постепенного перехода).

Таким образом, универсальный сервис-браузер должен взаимодействовать по крайней мере с одним служебным сервисом — сервисом обновления. Благодаря этому сервису могут динамически обновляться как служебные модули, из которых и составляется сервис-браузер, так и прикладные, входящие в состав информационной системы. Причем служебные модули и соответствующие им служебные сервисы обеспечивают собственно работу сервис-браузера, а прикладные модули и сервисы для каждой информационной системы должны быть разработаны индивидуально.

По этой причине при первом запуске браузера на машине клиента отсутствуют модули, входящие в прикладную информационную систему, которая определяется по URL, заданному при запуске сервис-браузера. Модуль обновления сервис-браузера совместно с сервисом обновления проводят первоначальную загрузку прикладных модулей на клиентскую машину, после чего она готова к началу работы.

Вместе с тем очевидно, что для современной информационной системы недостаточно наличия в универсальном сервис-браузере только лишь модуля обновления, взаимодействующего с сервисом обновления. Необходимы также другие служебные модули и сервисы.

Аутентификация и авторизация в сервис-браузере

Если сервис-браузер может вызывать различные системы (или подсистемы), естественным представляется обеспечение унифицированных методов доступа. В этом случае целесообразно применять средство управления доступом, использующее механизмы единой аутентификации, такие как Single Sign On (SSO) [5].

Решения на основе SSO обеспечивают аутентификацию и авторизацию пользователя в различных системах, и при этом исключают необходимость запоминания множества паролей. Это особенно актуально для больших корпоративных структур [6], где после входа в большую систему пользователь должен иметь доступ к тем или иным ее фрагментам на основе единых правил авторизации.

Нужно отметить, что при использовании web-сервисов (как служебных, так и прикладных) должны соблюдаться технологии защиты информации [7], в основном базирующиеся на спецификациях OASIS. Для работы в рамках единой структуры безопасности разработчик сервис-браузера должен предоставлять прикладным программистам соответствующие библиотеки и описание интерфейса взаимодействия.

Для обеспечения защищенного входа в систему сервис-браузер должен кроме модуля обновления иметь также следующие служебные модули, взаимодействующие с соответствующими сервисами:

- модуль аутентификации;
- модуль авторизации.

Модуль аутентификации либо обеспечивает считывание информации "логин—пароль", либо читает ключевую информацию с USB-токена [5], предоставленного пользователем, и PIN-код, вводимый в данном случае пользователем. Сочетание "логин—пароль" направляется сервису аутентификации, где проверяется, и при положительном исходе проверки пользователю возвращается идентификатор сеанса. PIN-код не передается на сервер и хранится на компьютере пользователя обычно в преобразованном виде. Преобразование выполняется с использованием односторонней хеш-функции. Такой PIN-код может размещаться и на USB-токене. Если используется схема с USB-токеном и успешно пройдена проверка PIN-кода, то на сервер аутентификации посылается запрос на получение кодовой комбинации. Полученная комбинация подписывается электронной подписью с использованием ключевой информации, хранящейся на USB-токене, и вновь отправляется на сервер аутентификации, где проверяется электронная подпись и откуда получается идентификатор сеанса, так же как и в случае "логин—пароль".

С использованием этого идентификатора сеанса далее могут вызываться все доступные сервисы до следующей процедуры аутентификации. Например, при перезапуске сервис-браузера.

Следует отметить, что описанные процедуры являются наиболее распространенными, но при проектировании сервис-браузера могут использоваться и другие, например, с применением биометрической аутентификации. Общим моментом здесь является ориентация на технологию SSO и следующее отсюда обстоятельство, в соответствии с которым прикладн-

ные модули и сервисы не используют свою процедуру аутентификации, а пользуются той, которую им предоставляет сервер-браузер.

Авторизация в системах, использующих сервер-браузер, проводится с использованием матрицы доступа [8]:

$$\mathbf{D} = \begin{bmatrix} R_{11} & \cdots & R_{1n} \\ \vdots & \ddots & \vdots \\ R_{m1} & \cdots & R_{mn} \end{bmatrix}$$

Матрица **D** имеет элементы R_{ij}; $i \leqslant m, j \leqslant n$. Столбцы соответствуют субъектам, т. е. пользователям, а строки — объектам, т. е. прикладным модулям. Таким образом, в системе, обслуживаемой сервис-браузером, каждый элемент R_{ij} принимает либо значение, равное единице, означающее, что пользователь *j* может иметь доступ к прикладному модулю *i*, либо ноль, что запрещает пользователю *j* иметь доступ к прикладному модулю *i*.

Обычно в матрице доступа элемент R_{ij} может принимать большее число значений, например, дополнительно чтение/запись. Но здесь необходимо учитывать, что модули являются программами и главная их характеристика — возможность выполнения модуля данным пользователем.

Если сервер-браузер обслуживает одновременно несколько систем (подсистем), то для каждой подсистемы должна быть своя матрица доступа, связанная с URL начального вызова, либо объединенная матрица, по которой можно определять доступ к модулям разных систем.

Когда модуль аутентификации получает идентификатор сеанса, открытого данным клиентом в процессе аутентификации, то этот идентификатор передается модулю авторизации, который вызывает сервис авторизации. Сервис авторизации по идентификатору сеанса определяет пользователя и устанавливает список прикладных модулей, которые доступны данному пользователю. Список возвращается модулю авторизации, который отображает его на экране. Далее пользователь может активизировать любой из отображенных модулей, т. е. модулей, к которым ему разрешен доступ.

Важно, что активизируемый модуль получает идентификатор пользователя и уже по внутренним правилам конкретной информационной системы может в соответствии с ролью, назначенной данному пользователю для работы с данным модулем, расширять или сужать его полномочия. Однако в задачи собственно сервер-браузера это не входит.

Общая схема взаимодействия сервис-браузера с прикладными модулями, сервисами и данными

Общая схема взаимодействия сервис-браузера с прикладными модулями, сервисами и данными приведена на рис. 1.

Нужно отметить, что перед первым запуском сервис-браузера на рабочем месте клиента (рассматриваем вариант Windows) ничего не установлено, кроме модулей провайдера безопасности, например, "КриптоПро CSP". Сервис-браузер может быть инсталлирован по сети с использованием обычного браузера. Практика показывает, что инсталляция проходит за время, не превышающее несколько десятков секунд, так как сервис-браузер достаточно компактен и, по существу, предназначен только для

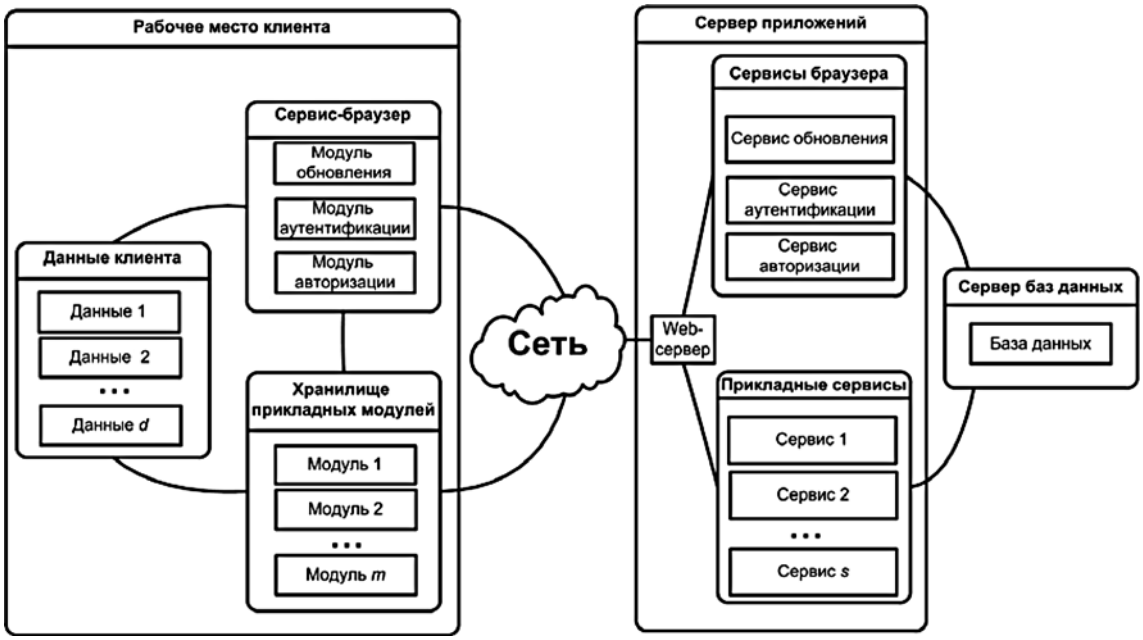

Рис. 1. Общая схема взаимодействия сервис-браузера с прикладными модулями, сервисами и данными

обеспечения работы модулей обновления, аутентификации и авторизации.

Далее в соответствии с выбранным URL происходит аутентификация пользователя и его авторизация. После авторизации определяется перечень модулей, с которыми будет работать абонент. В процессе обновления эти модули копируются с сервера на машину клиента в хранилище прикладных модулей. Также копируются на машину клиента данные, необходимые для работы.

Затем пользователь с использованием средств сервис-браузера выбирает прикладной модуль, с которым он хочет работать, и запускает его. Прикладной модуль без обращения к web-серверу осуществляет отрисовку элементов пользовательского интерфейса, предоставляет пользователю возможность ввода данных, а затем вызывает с использованием web-сервера соответствующий прикладной сервис. Сервис обрабатывает полученный запрос, обращаясь, при необходимости, к серверу баз данных. Клиентский модуль получает ответ и отображает его в соответствии с формой, которую он задает. То есть в данном случае ответ в виде данных (без оформления) получается от сервера, а форму его отображения определяет на рабочем месте клиента заранее скачанный модуль, который ранее вызывал сервис. При этом форма вывода может динамически изменяться в соответствии с составом и объемом данных, что определяется программным обеспечением клиентского модуля. Отметим, что оператором может быть проведен запуск нескольких различных прикладных модулей. При этом сервис-браузер отслеживает состояние модулей и может прекратить их работу.

В процессе работы передача данных между модулем и сервисом проводится в защищенном режиме. В простейшем случае для этого может использоваться технология HTTPS.

Отметим, что согласованную работу с элементами пользовательского интерфейса берут на себя сервис-браузер и прикладные модули, а множество прикладных модулей и прикладных сервисов, заранее не известных сервис-браузеру, обеспечивают функциональное наполнение той или иной системы. Здесь важно только то, что разрабатываемые прикладные модули должны отвечать правилам, задаваемым сервис-браузером, иначе возникнут конфликты как при отображении элементов пользовательского интерфейса, так и при отработке функций защиты и в ряде других случаев. Конечно, нужно иметь в виду, что прикладные модули информационных систем пишутся, как правило, для определенной операционной среды, что является естественным ограничением при использовании сервис-браузера.

Экспериментальные оценки

Для сравнения предложенной организации клиентских мест с использованием сервис-браузеров с клиентскими местами на основе традиционных web-сервисов были рассмотрены три варианта построения клиент-серверных архитектур (рис. 2).

Для проведения эксперимента были разработаны два макета сервис-браузеров, использующие REST-сервисы и SOAP-сервисы, которые будем называть REST-браузером и SOAP-браузером соответственно. Для проведения сравнения с ними в качестве универсального HTML-браузера использовался MS Internet Explorer v.11. Язык программирования — C#.

При проведении эксперимента использовали компьютеры с процессором Core-i5 (x64), оперативной памятью 4 Гб и операционными системами Windows Server 2012 — для серверов и Windows10 — для клиентских мест. Причем web-сервер и сервер приложений совмещались на одном физическом сервере. Использовалась локальная сеть 100 Мбит/с.

Для тестирования передаваемая информация создавалась случайным образом. На клиентском месте в результате должна отобразиться таблица, содержащая следующие столбцы:

- порядковый номер;
- случайный текст заданной длины;
- изображение.

Вызов web-сервиса, базирующегося на SOAP, происходил через функцию DataSet GetData(int countRow, int TextLength), где countRow — число строк в возвращаемой таблице; TextLength — размер случай-

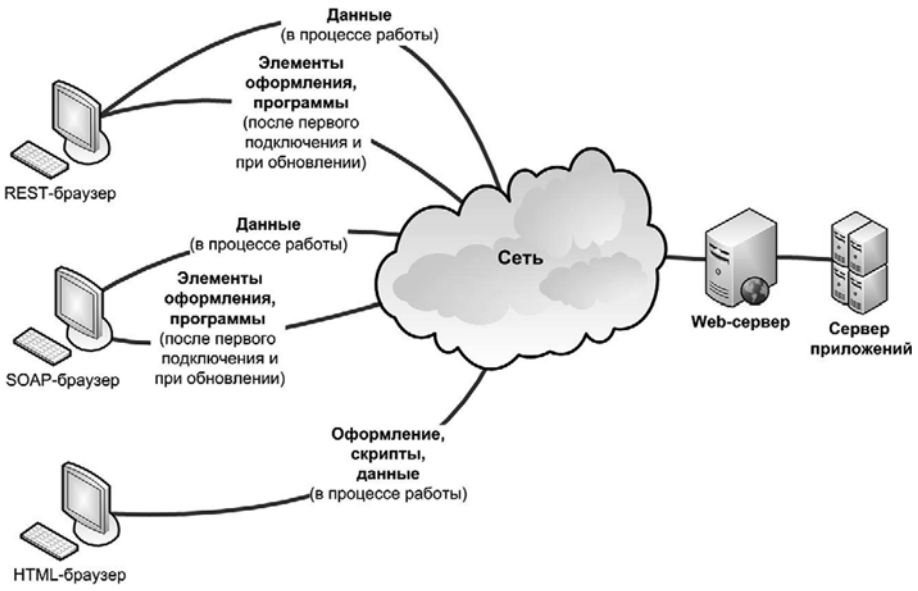

Рис. 2. Три браузера, работающие с web-сервисом

ного текста в каждой строке. Функция возвращала таблицу, которая отображалась пользователю:

```
TestReceivingTimeSoapClient s = new TestReceivingTimeSoapClient();
DataSet ds = s.GetData(countRow, TextLength);
if (ds == null) return;
foreach (DataRow s in ds.Tables[0].Rows) {
        int index = tblView.Rows.Add();
        tblView.Rows[index].Cells[0].Value = s[0].ToString();
        tblView.Rows[index].Cells[1].Value = s[2].ToString();
        tblView.Rows[index].Cells[2].Value =
              Image.FromFile(@"\TestReceivingTime\" + s[1].ToString());
        tblView.Rows[index].Height = 257;
}
```

Метод REST-сервиса, базирующегося на формате JSON, вызывался с помощью POST-запроса с двумя параметрами: count — число строк в возвращаемой таблице; textLength — размер случайного текста в каждой строке. Полученный результат преобразовывался в массив объектов и выдавался на экран пользователю:

```
string json;
string url = "http://application3/SiteForTestArticle/Home/GetData";
using (var client = new WebClient()) {
       client.Headers[HttpRequestHeader.ContentType] =
             "application/x-www-form-urlencoded";
       client.Headers[HttpRequestHeader.CacheControl] = "max-age = 0";
       client.Headers[HttpRequestHeader.Accept] =
             "text/html,application/xhtml + xml,application/xml;
       q=0.9,image/webp,*/*;q = 0.8";
       json = client.UploadString(url, "POST", "count=" + countRow.ToString()
+ "&textLength=" + TextLength.ToString());
}
List<RowWithImage> cl = Newtonsoft.Json.JsonConvert.
                    DeserializeObject<List<RowWithImage>>(json);
foreach (var s in cl) {
       int index = tblView.Rows.Add();
       tblView.Rows[index].Cells[0].Value = s.Number.ToString();
       tblView.Rows[index].Cells[1].Value = s.RandomText;
       tblView.Rows[index].Cells[2].Value =
             Image.FromFile(@"\TestReceivingTime\", s.ImageFileName);
       tblView.Rows[index].Height = 257;
}
```

HTML-страница подготавливалась с помощью технологии ASP.NET MVC 4 и формировала готовую таблицу с изображениями в ответ на POST-запрос с двумя параметрами, аналогичными REST-запросу:

```
@if (Model! = null)
{
        <table>
        <tr><th>№ </th><th>Картинка</th><th>Описание</th></tr>
        @foreach (var s in Model)
        {
            <tr>
            <td>@s.Number</td>
            <td><img id = "list_id" src = "@Url.Content("~/Content/Images/"+
        @s.ImageFileName)"/></td>
            <td>@s.RandomText</td>
            </tr>
        }
        </table>
}
```

Рис. 3. Время загрузки страницы в зависимости от типа браузера

Результаты измерений для первого варианта, когда объем оформления каждой страницы существенно больше объема передаваемых данных, приведены на рис. 3.

По данным рис. 3 можно сделать вывод, что время, затрачиваемое на загрузку страницы HTML-браузером, включающее ее передачу с сервера и отображение, больше времени REST-браузера и SOAP-браузера, использовавшихся для выполнения аналогичных действий. Причем результаты REST- и SOAP-браузеров в принципе идентичны, и их расхождение находится в пределах погрешности измерений. В то же время универсальный HTML-браузер в среднем работает приблизительно в 5 раз медленнее. Это естественно, так как HTML-браузер большую часть времени затрачивает на транспортировку данных оформления по сети, а сервис-браузеры переносят только смысловые данные, объем которых существенно меньше. Для оформления, использующего рисунки с фиксированным содержимым в виде файлов, включается механизм кеширования, и время будет существенно сокращаться, приближаясь ко времени REST- и SOAP-браузеров.

Нужно отметить, что все данные получены для случая, когда объем оформления существенно больше объема передаваемых данных. По мере того как размер данных (динамическая составляющая) относительно объема оформления (статическая составляющая) будет увеличиваться, разница по времени между сервис-браузерами и HTML-браузером должна сокращаться. Это подтверждают результаты эксперимента, приведенные на рис. 4.

Здесь в основном даны значения для SOAP-браузера. Отметим, что результаты для REST-браузера имеют близкие значения, и поэтому здесь не рассматриваются. Также на рис. 4 представлен для сравнения один график для HTML-браузера с соотношением динамических и статических данных, равным единице.

Графики для SOAP-браузера различаются между собой объемом динамических данных. Видно, что время загрузки по мере увеличения этого объема возрастает. HTML-браузер все равно использует больше времени для загрузки, хотя временные показатели сервис-браузеров начинают приближаться к показателям HTML-браузера по мере увеличения соотноше-

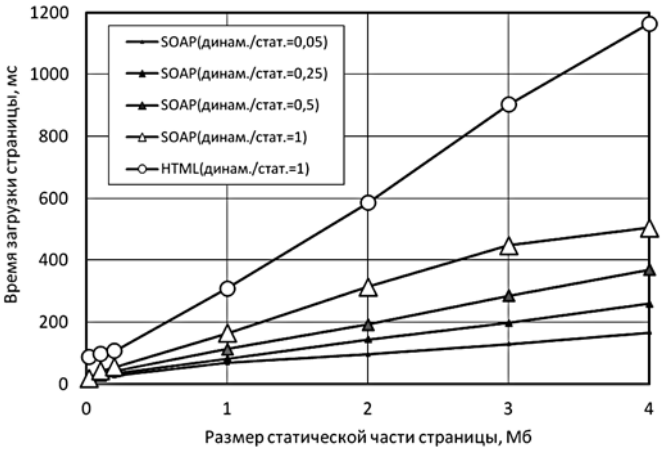

Рис. 4. Время загрузки страницы по мере увеличения объема передаваемых данных при фиксированном соотношении динамической и статической составляющих (динам./стат.)

ния динамические/статические данные. Так, SOAP-браузер при малом количестве динамических данных (см. рис. 3) работает быстрее в среднем в 4,47 раза по сравнению с HTML-браузером. При соотношении, равном единице (см. рис. 4), среднее превышение равно 2,43.

Также были проведены проверки работы браузеров в локальной сети и сети Интернет. При этом обеспечивался вход во всех точках со скоростью 100 Мбит/с. То есть по параметрам входа глобальная сеть соответствовала локальной. Результаты проверки приведены на рис. 5.

Здесь на диаграмме показаны времена загрузки страницы размером 0,6 Мб в различных сетях для трех типов браузеров. Измерения в сети Интернет проводились на территории Москвы. Значения 3 км и 15 км от клиента до сервера нужно считать условными, так как время передачи напрямую не зависит от расстояния, а определяется многими факторами. Данные точки были выбраны случайно вне пределов локальной сети, чтобы проверить скорость работы браузеров. Никакого специального оборудования для полигонов не требовалось. Достаточно было в каждой точке иметь компьютер, присоединенной к сети Интернет, с установленным сервис-браузером. Сервис-браузер при первоначальном запуске определял, какие прикладные модули и данные нужно скачать, скачивал их, показывал оператору перечень, а оператор уже выбирал нужный прикладной модуль и активизировал его. Модуль выбирал настройки, предоставленные ему сервис-браузером, и вызывал соответствующий сервис.

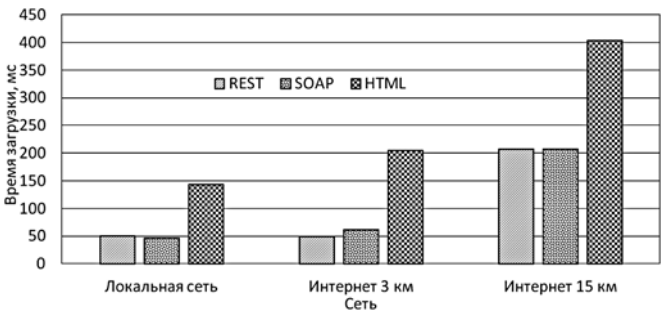

Рис. 5. Время загрузки в зависимости от организации сети

HTML-браузер (Internet Explorer) обращался к тому же самому web-серверу, вызывая HTML-страницы с идентичным оформлением и данными.

Из полученных в результате эксперимента диаграмм, представленных на рис. 5, следует, что в различных точках сети с разным обеспечением доступа браузеры типа SOAP и REST показывают каждый раз приблизительно одинаковые значения времени загрузки страницы относительно друг друга, а HTML-браузер требует в 2—3 раза больше времени, что объяснимо, ведь SOAP- и REST-браузеры свои программы и оформление скачали заранее.

Практическую проверку данная технология проходила при создании информационной системы, включенной в структуру Интерпола [9, 10]. Основной браузер, разработанный для этой системы, представляет собой SOAP-браузер. Это определялось скоростными характеристиками, хорошим инструментарием, наработанным ранее для создания SOAP-сервисов, использованием функций самодокументирования, необходимостью обеспечения высокого уровня защиты в распределенной системе. Важным обстоятельством также является то, что на удаленных расстояниях вероятность сетевых сбоев увеличивается. Поэтому важно заполнение сложных форм данными проводить практически в автономном режиме, подключая сервис только в момент передачи. Это повышает устойчивость процесса заполнения таких форм и обеспечивает резервное хранение вводимых данных клиента, что обычно просто запрещено в HTML-браузерах, так как по определению внешние программы (например, скрипты HTML-страниц) не должны вторгаться в пространство клиента. В сервис-браузерах клиента работает своя программа, ранее скачанная и проверенная. Поэтому она имеет возможность использовать хранилище данных клиента. Кстати, это оказалось удобным не только при передаче, но и при приеме данных, так как клиент естественно и непрерывно имеет возможность сохранять результаты своей (например, дневной) работы и в любое время их анализировать, печатать, не тратя сетевых ресурсов.

Заключение

Использование HTML-браузеров в настоящее время растет. С увеличением быстродействия и надежности работы сетей их применение в качестве клиентских мест информационных систем становится все более привлекательным. Вместе с тем сервис-браузеры, рассмотренные в настоящей статье, также могут найти свое место для построения распределенных корпоративных структур, обеспечивающих постоянную обработку информации на рабочих местах. Эти браузеры, оставаясь универсальными, могут

аккумулировать на начальном этапе работы клиентские программы и данные методом динамически обновляемых модулей, подкачивая их с сервера приложений конкретной системы. Они обеспечивают аутентификацию и авторизацию пользователей общим для нескольких программных комплексов способом. Они дают возможность в процессе текущей работы передавать по сети только данные, без оформления и скриптов, за счет чего увеличивается быстродействие. И наконец, они обеспечивают хранение необходимого минимального объема данных на рабочем месте, что разгружает сервер. Как правило, web-клиент больше ориентирован на просмотр и анализ информации, в то время как сервис-браузер позволяет не только просматривать, но и обеспечивать потоковый ввод в систему большого количества данных.

Следует отметить, что предложенное решение хорошо подходит для реализации как в обычных клиентских системах, так и в сфере мобильных приложений. Как правило, необходимы небольшие вычислительные мощности для сервис-браузера и его прикладных модулей на рабочем месте пользователя, ведь вся нагрузка по обработке данных возлагается на сервисы и базу данных.

Список литературы

1. **Шапошников И. В.** Web-сервисы Microsoft.net. СПб.: БХВ-Петербург, 2002. 336 с.

2. **Fielding R. T.** Architectural Styles and the Design of Network-based Software Architectures. Dissertation doctor of philosophy. University of California, Irvine. 2000.

3. **Лебедев В. Н., Курако Е. А., Москальков В. Е., Орлов В. Л.** Построение и развитие распределенных информационных систем на основе динамически обновляемых модулей // Материалы 2-й Международной конференции "Управление развитием крупномасштабных систем". MLSD-2008. Москва, 2008. М.: ИПУ РАН, 2008. Т. 2. С. 114—115.

4. **Курако Е. А., Орлов В. Л.** Защищенный SOA-браузер в системах с сервис-ориентированной архитектурой // Труды 24-й Международной научной конференции "Проблемы управления безопасностью сложных систем". Москва, 2016. М.: Издательский Центр РГГУ, 2016. С. 175—177.

5. **Шаньгин В. Ф.** Защита информации в компьютерных системах и сетях. М.: ДМК Пресс. 2012. 592 с.

6. **Асратян Р. Э., Курако Е. А., Лебедев В. Н.** и др. Архитектура и методы построения автоматизированных информационно-аналитических систем предприятий ракетно-космической промышленности. М.: ИПУ РАН, 2016. 120 с.

7. **Шепелявый Д.** Обеспечение безопасности Web-сервисов // Информационная безопасность. 2008. № 1. С. 28—29.

8. **Щеглов А. Ю.** Защита компьютерной информации от несанкционированного доступа. СПб.: НиТ, 2004. 384 с.

9. **Апульцин В. А., Курако Е. А., Орлов В. Л.** Развитие информационных технологий в НЦБ Интерпола МВД России // Труды 25-й Всероссийской конференции "Информатизация и информационная безопасность правоохранительных органов", Москва, 2016. М.: Академия управления МВД России, 2016. С. 10—15.

10. **Курако Е. А., Орлов В. Л.** Системы объектно-связанного документооборота и организация их взаимодействия с бизнес-процессами // Проблемы управления. 2017. № 2. С. 42—49.

Service Browsers for Information Systems

E. A. Kurako, kea@ipu.ru, **V. L. Orlov,** e-mail: ovl@ipu.ru, V. A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences, Moscow, 117997, Russian Federation

Corresponding author:

Kurako Evgeny A., Researcher, V. A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences, Moscow, 117997, Russian Federation,
E-mail: kea@ipu.ru

Received on April 24, 2017
Accepted on May 12, 2017

Use of various types of clients in information systems is considered. The concept service browser, which interacts with the server by a call of services of various types and display of answers, is introduced. The general design scheme of service browsers is given. Experiments for comparison of REST and SOAP service browsers with the traditional HTML browser are made.

Keywords: *information system, web-client, browser, service-browser, REST, SOAP, HTML*

For citation:

Kurako E. A., Orlov V. L. Service Browsers for Information Systems, *Programmnaya Ingeneria,* 2017, vol. 8, no. 9, pp. 413—421.

DOI: 10.17587/prin.8.413-421

References

1. **Shaposhnikov I. V.** *Web-servisy Microsoft.net* (Microsoft.net web services), Saint Petersburg, BHV-St. Petersburg, 2002, 332 p. (in Russian).
2. **Fielding R. T.** Architectural Styles and the Design of Network-based Software Architectures. Dissertation doctor of philosophy, Irvine: University of California, 2000, 162 p.
3. **Lebedev V. N., Kurako E. A., Moskalkov V. E., Orlov V. L.** Postroenie i razvitie raspredelennyh informacionnyh system na osnove dinamicheski obnovlyaemyh modulej (Construction and development of the distributed information systems on the basis of dynamically updated modules), *Materials of the 2nd International conference "Management of Development of Large-scale Systems" (MLSD-2008, Moscow)*, Moscow, ICS, RAS, 2008, vol. 2, pp. 114-115. (in Russian).
4. **Kurako E. A., Orlov V. L.** Zaschischennyj SOA-brauser v sistemah s servis-orientirovannoy arhitekturoj (The protected SOA browser in systems about service), *The 24th International scientific conference "Problems of Management of Safety of Difficult Systems"*, Moscow, RGGU Publishing Center, 2016, pp. 175—177 (in Russian).
5. **Shan'gin V. F.** *Zaschita informacii v komp'yuternyh sistemah i setyah* (Information security in computer systems and networks), Moscow, DMK Press, 2012, 592 p. (in Russian).
6. **Asratyan R. E., Kurako E. A., Lebedev V. N., Marakanov I. N., Noga N. L., Orlov V. L.** *Arhitektura i metody postroeniya avtomatizirovannyh informacionno-analiticheskih system predpriyatij raketno-kosmicheskoj otrasli* (Architecture and the methods of construction automated informational and analytical systems of the enterprises of the space-rocket industry), Moscow, ICS RAS, 2016, 120 p. (in Russian).
7. **Shepelyavyi D.** Obespechenie bezopasnosti Web-servisov (Safety of Web services), *Information Security*, 2008, no. 1, pp. 28—29 (in Russian).
8. **Scheglov A. Yu.** *Zaschita komp'yuternoj informacii ot nesankcionirovannogo dostupa* (Protection of computer information against unauthorized access), Saint Petersburg, NiT, 2004, 384 p. (in Russian).
9. **Apultsin V. A., Kurako E. A., Orlov V. L.** Razvitie informacionnyh tehnologij v NCB Interpola MVD Rossii (Development of information technologies in NCB of the Interpol of the Ministry of Internal Affairs of Russia), *Works of the 25th All-Russian conference "Informatization and Information Security of Law Enforcement Agencies"*, Moscow, Academy of the Department of MIA of Russia, 2016, pp. 10—15 (in Russian).
10. **Kurako E. A., Orlov V. L.** Sistemy ob'ektno-svyazannogo dokumentooborota i organizacia ih vzaimodejstviya s biznes-processami (Object-associated document flow systems and organization of their interaction with business processes), *Problemy upravlenija*, 2017, no. 2, pp. 42—49 (in Russian).

ИНФОРМАЦИЯ

SECR 2017 впервые пройдет в Санкт-Петербурге

13-я международная научно-практическая конференция "Software Engineering Conference Russia/Разработка программного обеспечения" впервые за свою историю пройдет в Санкт-Петербурге. Это ключевое профессиональное событие в индустрии разработки программного обеспечения России. Основные дни мероприятия: 20—21 октября (пятница, суббота), а 22 октября (воскресенье) — дополнительный день мастер-классов.

Уже известны имена ключевых спикеров, ими станут: Ивар Якобсон — основоположник компонентной архитектуры, UML и RUP, Эрик Райс — автор знаменитой книги "Usable Usability" и Андрей Николаевич Терехов — заведующий кафедрой системного программирования СПбГУ и генеральный директор Ланит-Терком.

Результаты рецензирования заявок и статей авторов были опубликованы 31 августа. Сейчас Программным комитетом формируется основная программа конференции, но уже известны самые популярные темы заявок SECR 2017. Традиционно ими стали темы, относящиеся к programming technologies. Интересно, что на втором месте заявки, посвященные тестированию, верификации и анализу ПО. Доклады по архитектуре программных систем и управлению проектом и продуктом также с отрывом впереди остальных. Завершает пятерку лидеров тема Programming Tools, которая собрала 18 работ.

Из 165 заявок 32 позиционируются как научные работы, а значит, могут побороться за премию Бертрана Майера в размере 40 000 руб., как лучший исследовательский доклад в области программной инженерии. Научные статьи в случае принятия будут опубликованы в электронной библиотеке ACM, сборник которых, как правило, индексируется Scopus и Web Of Science.

Независимость SECR позволяет отсекать рекламные доклады. Программа включает весь спектр технологий, методологий и инструментов современного ПО и общие вопросы, связанные с подготовкой кадров, управлением проектов, развитием и продвижением сферы и бизнеса. Начиная с 2015 г., отдельно рассматриваются главные ИТ-тренды прошедшего года:

- технологии и инструменты программирования;
- языки программирования;
- наука о данных, большие данные;
- дизайн-мышление;
- облачные модели использования ресурсов;
- тестирование, верификация и анализ ПО;
- взаимодействие человека и компьютера;
- цифровая трансформация;
- ИИ в разработке, тестировании; ИИ и Big Data;
- разумные приложения и разумные вещи;
- DevOps.

И это лишь некоторые из тем, представленных на конференции.

Программа мероприятия идет в 4—5 потоков одновременно, и участники свободно выбирают между несколькими темами и спикерами. Кроме конкурсных докладов в нее входят дискуссии, короткие мастер-классы и презентации от иностранных экспертов — лидеров мирового уровня.

Для тех, кто зарегистрируется на конференцию до 1 октября, действует специальная льготная цена. Также предусмотрены скидки для студентов, аспирантов и преподавателей, групп от трех человек и сотрудников компаний — членов ассоциации РУССОФТ (скидки суммируются).

Вся подробная информация о мероприятии на сайте: www.secr.ru

УДК 517.5, 004.4 DOI: 10.17587/prin.8.422-432

И. С. Астапов, канд. физ.-мат. наук, ст. науч. сотр., e-mail: velais@imec.msu.ru,
НИИ механики МГУ им. М. В. Ломоносова,
Н. С. Астапов, канд. физ.-мат. наук, доц., ст. науч. сотр., Новосибирский государственный университет, Институт гидродинамики им. М. А. Лаврентьева СО РАН, г. Новосибирск

Алгоритмы символьного решения алгебраических уравнений

Предложены два новых алгоритма символьного представления корней уравнения третьей степени. Описан способ решения уравнения четвертой степени с помощью возвратного уравнения. Построены алгоритмы для символьного решения алгебраических уравнений пятой и шестой степеней специальных видов. Особое внимание уделено уравнениям, разрешимым в квадратных радикалах. Указаны разложения на множители некоторых полиномов высоких степеней. Эффективность предложенных способов проиллюстрирована сравнением с решениями, генерируемыми пакетом прикладных программ Mathematica.

Ключевые слова: *возвратные уравнения, решение в радикалах, формулы Кардано, резольвента, полином Муавра, модулярное уравнение, компьютерная алгебра, программное обеспечение*

Введение

В физико-технических расчетах часто возникает необходимость явного символьного (не численного) выражения корней алгебраических уравнений через буквенные коэффициенты этих уравнений для последующего исследования физических закономерностей. Для уравнений третьей и четвертой степеней обычно используют так называемые формулы Кардано. Однако эти формулы оказываются неконструктивными для уравнений с произвольными буквенными коэффициентами, так как не существует алгоритма извлечения кубического корня из произвольного комплексного числа. По этой причине формулы Кардано являются компьютерно трудно реализуемыми в случае, когда исходное кубическое уравнение или резольвента уравнения четвертой степени имеют три различных действительных корня.

Алгебраические уравнения пятой или более высокой степеней в общем виде не имеют решения в радикалах. Однако специальные виды уравнений, например, двучленные уравнения $x^n - a = 0$, решаются в радикалах, а уравнение пятой степени общего вида решается с помощью тэта-функций [1]. Более того, корни любого алгебраического уравнения могут быть выражены в виде обобщенных гипергеометрических функций от коэффициентов этого уравнения [1, 2]. В пакете прикладных программ Mathematica для решения алгебраических уравнений используют базисы Гребнера [3]. В работе [4] приведены аналитические выражения для представления всех корней произвольного алгебраического уравнения в виде отношения бесконечных определителей. Поэтому в результате значения корней выражаются в конечном виде численно и приближенно. В работе [5] дан способ выражения корней произвольного алгебраического уравнения, использующий интегральную формулу Меллина. В работе [6] представлен способ поиска корней уравнения в поле алгебраических чисел. Несмотря на то что предлагаемые в работах [4—6] способы пригодны для решения уравнений произвольных степеней, реализующие их алгоритмы оказываются трудоемкими даже для уравнений невысоких степеней, либо, например, в силу ограничений на коэффициенты уравнений, позволяют получить значения корней не в символьном виде.

В предисловии к монографии [2] отмечена актуальность поиска простых алгоритмов для решения алгебраических уравнений, приведены примеры инженерно-технических задач, в которых возникает необходимость решения алгебраических уравнений третьей—восьмой степеней. В работе [2] приведена обширная библиография, содержащая более 280 ссылок, подтверждающая актуальность настоящей работы.

В частных случаях, когда коэффициенты исходных уравнений связаны какими-либо дополнительными соотношениями, иногда удается выразить в конечном виде корни уравнений через коэффициенты существенно более просто, чем это реализовано, например, в системе Mathematica 8.0 [7].

Настоящая работа является естественным продолжением работы [7].

Кубическое уравнение

Уравнение третьей степени вида

$$z^3 + az^2 + bz + b^2/(3a) = 0 \qquad (1)$$

имеет корень $z = -b/\left(a + \sqrt[3]{3ab - a^3}\right)$, что проверяется подстановкой этого значения непосредственно в уравнение. С помощью замены переменной

$z = x - a/3$ приведем уравнение (1) к каноническому виду $x^3 + (3b - a^2)/3x + (3b - a^2)(3b - 2a^2)/(27a) = 0$. Обозначим коэффициент при первой степени неизвестного x через p, а свободный член — через q. Из полученной системы уравнений найдем a и b:

$$a = 9\left(-q \pm 2\sqrt{D}\right)/(2p),\ \ b = \left(3p + a^2\right)/3, \qquad D = p^3/27 + q^2/4. \tag{2}$$

Итак, справедливо утверждение: одним из корней кубического уравнения в канонической форме $x^3 + px + q = 0$ является $x = a/3 - b/\left(a + \sqrt[3]{3ab - a^3}\right)$, где a и b определяются равенствами (2), причем в выражении для a можно взять любой знак. Остальные корни находятся с помощью формул Виета. Несмотря на то что этот способ решения кубического уравнения отличается от известных способов [7], при его использовании в неприводимом случае возникают те же затруднения. Однако изложенный способ решения произвольного кубического уравнения построен на разложении на множители кубического уравнения (1) частного вида. Ниже, пользуясь разложением на множители полинома шестой степени специального вида, будут даны новые, более плодотворные алгоритмы решения произвольных кубических уравнений.

Выясним, при каких условиях кубическое уравнение

$$z^3 + az^2 + bz + c = 0 \tag{3}$$

с произвольными комплексными коэффициентами a, b и c умножением обеих частей уравнения на линейный двучлен $z + t$ приводится к возвратному уравнению четвертой степени, которое разрешимо в квадратных радикалах [7]. Приравнивая коэффициенты при одинаковых степенях z в равенстве $(z^3 + az^2 + bz + c)(z + t) = z^3 + mz^2 + nz^2 + mlz + l^2$, имеем систему уравнений $(t + a)l = bt + c$, $l^2 = ct$. Если $t + a = 0$ и $c = -bt = ab$, то получим биквадратное уравнение $(z^3 + az^2 + bz + ab)(z - a) = (z^2 - a^2)(z^2 + b)$. Если $t + a \neq 0$, то, выбирая любой корень l резольвенты $l^3 - bl^2 + acl - c^2 = 0$, найдем значение $t = l^2/c$, с помощью которого уравнение (3) приводится умножением на линейный двучлен $z + t$ к возвратному уравнению.

С помощью средств компьютерной алгебры нетрудно проверить, что если коэффициенты кубического уравнения (3) связаны соотношением

$$2a^6 - 18a^4b + 4a^3c + 53a^2b^2 - 18abc - 50b^3 + 27c^2 = 0,$$

или, другими словами, соотношением

$$c = \left(a(9b - 2a^2) \pm 5(3b - a^2)\sqrt{2(3b - a^2)}\right)/27,$$

то уравнение (3) приводится умножением на линейный двучлен $z + t$ к разрешимому в квадратных радикалах уравнению четвертой степени

$$\left(z^3 + az^2 + bz + c\right)(z + t) = (z + m)^4 + \left((3b - a^2)/2\right)^2 = 0,$$

$$t = \left(a \mp \sqrt{8(3b - a^2)}\right)/3,$$

$$m = (t + a)/4 = \left(a \mp \sqrt{(3b - a^2)/2}\right)/3.$$

Для кубического уравнения в каноническом виде эти формулы упрощаются следующим образом. Так, кубическое уравнение вида

$$x^3 + px \pm 5p\sqrt{6p}/9 = 0, \tag{4}$$

где p — произвольное комплексное число, имеет корни $x_1 = -2\sqrt{p/6}$, $x_{2,3} = \sqrt{p/6} \pm 3\sqrt{-p/6}$, если в уравнении (4) выбран знак "+"; и имеет корни $x_1 = 2\sqrt{p/6}$, $x_{2,3} = -\sqrt{p/6} \pm 3\sqrt{-p/6}$, если в уравнении (4) выбран знак "−". Таким образом, кубическое уравнение специального вида (4) разрешимо в квадратных радикалах. Заметим, что система прикладных программ Mathematica 8.0 выдает для корней уравнений вида (4), например, для уравнения $x^3 + \left(\sqrt{2} + \sin 3\right)x + 5\left(\sqrt{2} + \sin 3\right)\sqrt{6\left(\sqrt{2} + \sin 3\right)}/9 = 0$ громоздкие выражения. Решение, построенное для кубического уравнения (4), можно использовать и для решения алгебраических уравнений высоких степеней специального вида, подставляя вместо величины p квадрат многочлена от x. Так, например, при $p = \left(ax^2 + bx + c\right)^2/6$ уравнение $x^3 + \left(ax^2 + bx + c\right)^2 x/6 \pm 5\left(ax^2 + bx + c\right)^3/54 = 0$ оказывается возвратным уравнением шестой степени частного вида. Корни этого уравнения выражаются через квадратные радикалы и находятся среди корней квадратных уравнений $x = \pm\left(ax^2 + bx + c\right)/3$, $x = \left(ax^2 + bx + c\right)(1 \pm 3i)/6$ и $x = -\left(ax^2 + bx + c\right)(1 \pm 3i)/6$, что проверяется непосредственной подстановкой.

Далее, при рассмотрении возвратных уравнений шестой степени будут даны два новых способа решения кубических уравнений.

Уравнение четвертой степени

В работе [7] доказана теорема о том, что любое уравнение четвертой степени

$$z^4 + az^3 + bz^2 + cz + d = 0 \tag{5}$$

с произвольными комплексными коэффициентами a, b, c и d приводится к возвратному относительно x уравнению подстановкой $z = x + t$, где t — какой-либо корень кубического уравнения (резольвенты)

$$t^3\left(a^3 - 4ab + 8c\right) + t^2\left(a^2b + 2ac - 4b^2 + 16d\right) + t\left(a^2c + 8ad - 4bc\right) + a^2d - c^2 = 0. \tag{6}$$

Так, для корней уравнения $z^4 + \left(4 + \sqrt{3}\right)z^3 + \left(6 + 3\sqrt{3}\right)z^2 + \left(4 + 2\sqrt{3}\right)z + 2 = 0$ Mathematica 8.0 генерирует длинные выражения с использованием кубических корней. Однако резольвента (6) этого уравнения имеет корень $t = -1$. Поэтому с помощью подстановки $z = x - 1$ это уравнение приводится

к возвратному уравнению $x^4 + \sqrt{3}x^3 - \sqrt{3}x + 1 = 0$, корни которого $x_{1,2} = \left(\sqrt{15} - \sqrt{3} \pm i\left(\sqrt{5} - 1\right)\right)/4$, $x_{3,4} = \left(-\sqrt{15} - \sqrt{3} \pm i\left(\sqrt{5} + 1\right)\right)/4$ выражаются через квадратные радикалы.

Рассмотрим несколько случаев, когда уравнение (6) легко разрешимо. Если коэффициенты исходного уравнения (5) связаны равенством $a^3 - 4ab + 8c = 0$, то резольвента принимает вид $\left(d - (a^2 - 4b)^2/64\right)(4t + a)^2 = 0$ и, следовательно, $t = -a/4$ является двукратным корнем резольвенты. В этом случае уравнение (5) подстановкой $z = x - a/4$ приводится к биквадратному уравнению (частному случаю обобщенного возвратного уравнения)

$$x^4 - (3a^2 - 8b)x^2/8 + (5a^4 - 16a^2b + 256d)/256 = 0.$$

Таким образом, уравнение $z^4 + az^3 + bz^2 + (a^3 - 4ab)z/8 + d = 0$ с произвольными комплексными коэффициентами a, b и d разрешимо в квадратных радикалах.

При выполнении равенства $d = (a^2 - 4b)^2/64$ резольвента (6) принимает вид

$$(a^3 - 4ab + 8c)\times$$
$$\times\left(t^3 + at^2/4 + (a^2 - 4b)t/8 + (a^3 - 4ab - 8c)/64\right) = 0$$

и при дополнительном условии $b = 5a^2/24$ имеет корень $t = -a/12 + \sqrt[3]{c/8 - 7a^3/3456}$. Следовательно, уравнение

$$z^4 + az^3 + 5a^2z^2/24 + cz + a^4/2304 = 0$$

приводится подстановкой $z = x = t$ к возвратному относительно x уравнению.

Если выполняется равенство $d = (a^2 - 4b)^2/64$ и равенство $a^3 - 4ab - 8c = 0$, то уравнение (5) является возвратным и, следовательно, разрешимо в квадратных радикалах.

Далее, при рассмотрении возвратных уравнений шестой степени будет представлен новый способ решения уравнения четвертой степени.

Уравнение пятой степени

При построении алгоритмов для решения алгебраических уравнений пятой степени здесь будем использовать известные теоретические результаты, подробно изложенные в монографии [1].

Рассмотрим уравнение пятой степени специального вида

$$x^5 + ax^3 + a^2/5\,x + b = 0, \qquad (7)$$

с многочленом Муавра в левой части (см. [1], с. 226). Уравнение (7) имеет корни

$$x_j = w^j\sqrt[5]{t_1} + w^{4j}\sqrt[5]{t_2}, \quad j = 0, 1, 2, 3, 4, \qquad (8)$$

где t_1 и t_2 — корни квадратного уравнения $t^2 + bt - (a/5)^5 = 0$, а w — любой примитивный корень пятой степени из единицы (см. [1], с. 211), например, $w = e^{2\pi i/5} = \left(\sqrt{5} - 1 + i\sqrt{2\sqrt{5} + 10}\right)/2$. То есть уравнение (7) решается в радикалах. Если коэффициенты a и b уравнения (7) связаны соотношением $a = -5\sqrt[5]{(b/2)^2}$, то $t_1 = t_2 = -b/2$ и $x = -2\sqrt[5]{b/2}$ является корнем уравнения (7). Например, уравнение $x^5 - 10x^3 + 20x + 8\sqrt{2} = 0$ имеет корень $x = -2\sqrt{2}$.

Тригонометрический способ Виета. По аналогии со способом Виета для решения кубических уравнений построим, пользуясь тождеством $\sin 5\alpha = 16\sin^5\alpha - 20\sin^3\alpha + 5\sin\alpha$, алгоритм решения алгебраических уравнений пятой степени. Умножая это тождество на $2R^5$ и обозначая $x = 2R\sin\alpha$, получим тождество $x^5 - 5R^2x^3 + 5R^4x + b = 0$, где $b = -2R^5\sin 5\alpha$. Следовательно, уравнение

$$x^5 - 5R^2x^3 + 5R^4x + b = 0 \qquad (9)$$

при условии $R > 0$, $-2R^5 \leqslant b \leqslant 2R^5$ имеет пять действительных корней

$$x_k = 2R\sin\left[\left(\arcsin\left(b/-2R^5\right) + 2\pi k\right)/5\right],$$

$k = 0, 1, 2, 3, 4$. Если выполняется равенство $b = \pm 2R^5$, то уравнение (9) имеет два двукратных корня. Проведем в уравнении (9) замену $-5R^2 = a$, получим уравнение (7). Следовательно, для уравнения пятой степени специального вида (7) с многочленом Муавра в левой части имеем два различных представления корней уравнения: через радикалы и с помощью тригонометрических функций. Заметим, что оба способа пригодны и для уравнений с комплексными коэффициентами. Аналогичный результат получим и в том случае, если воспользуемся тождеством $\cos 5\alpha = 16\cos^5\alpha - 20\cos^3\alpha + 5\cos\alpha$. Интересно отметить, что с помощью своего способа Ф. Виет решил алгебраическое уравнение сорок пятой степени специального вида (см. [8], с. 106—108).

Для уравнения пятой степени общего вида английским математиком А. Кэли были найдены условия, при выполнении которых уравнение разрешимо в радикалах, а также получены в явном виде корни этих разрешимых уравнений. Оказывается, любое уравнение пятой степени без кратных корней можно привести к виду

$$x^5 + ax + b = 0, \qquad (10)$$

решая при этом лишь уравнения второй и третьей степени (см. [1], с. 222—225), а это уравнение в случае $a \neq 0$ с помощью линейной подстановки $x = (a/5)^{1/4}t$ приводится к виду $t^5 + 5t + c = 0$.

Для уравнения (10) с рациональными коэффициентами a и b в монографии [1], приведено конструктивное доказательство следующей теоремы, которую можно использовать для построения алгоритма решения уравнения (10) и тогда, когда коэффициенты a и b не являются рациональными.

Теорема. ([1, с. 228—231]) *Пусть a и b — такие рациональные числа, что многочлен пятой степени $x^5 + +ax + b$ неприводим. Тогда уравнение (10) разрешимо в радикалах в том и только том случае, когда существуют такие рациональные числа $\varepsilon = \pm 1$, $c \geqslant 0$ и $f \neq 0$, что*

$$a = \frac{5f^4(3 - 4\varepsilon c)}{c^2 + 1}, \quad b = \frac{-4f^4(11\varepsilon + 2c)}{c^2 + 1}.$$

В этом случае корни уравнения имеют вид

$$x_j = f\left(w^j u_1 + w^{2j} u_2 + w^{3j} u_3 + w^{4j} u_4\right), \quad j = 0, 1, 2, 3, 4, \tag{11}$$

где $w = e^{2\pi i/5} = \left(\sqrt{5} - 1 + i\sqrt{2\sqrt{5} + 10}\right)/2$,

$$u_1 = \left(v_1^2 v_3 / D^2\right)^{1/5}, \quad u_2 = \left(v_3^2 v_4 / D^2\right)^{1/5}, \quad u_3 = \left(v_2^2 v_1 / D^2\right)^{1/5}, \quad u_4 = \left(v_4^2 v_2 / D^2\right)^{1/5}, \tag{12}$$

$$v_1 = \sqrt{D} + \sqrt{D - \varepsilon\sqrt{D}}, \quad v_2 = -\sqrt{D} - \sqrt{D + \varepsilon\sqrt{D}}, \quad v_3 = -\sqrt{D} + \sqrt{D + \varepsilon\sqrt{D}}, \quad v_4 = \sqrt{D} - \sqrt{D - \varepsilon\sqrt{D}}, \tag{13}$$

$$D = c^2 + 1.$$

Резольвентой уравнения (10) является уравнение шестой степени

$$r^6 + 8ar^5 + 40a^2r^4 + 160a^3r^3 + 400a^4r^2 + \left(512a^5 - 3125b^4\right)r + \left(256a^6 - 9375ab^4\right) = 0, \tag{14}$$

которое имеет рациональные корни, если выполнены условия теоремы. Окончательно для решения уравнения (10) с рациональными коэффициентами предлагается следующий алгоритм.

Шаг 1. Находим какой-либо рациональный корень r уравнения (14) с рациональными коэффициентами [7] (несколько новых способов символьного решения уравнений шестой степени специального вида будут приведены в следующем пункте).

Шаг 2. Если $(3r - 16a)/(4(r + 3a)) > 0$, то полагаем $\varepsilon = 1$, иначе полагаем $\varepsilon = -1$.

Шаг 3. Вычисляем $f = -5b\varepsilon/(2(r + 2a))$.

Шаг 4. Вычисляем значение $D = c^2 + 1 = 25(r^2 + 16a^2)/(16(r + 3a)^2)$. Заметим, что в работе [1] выражение для $c^2 + 1$ дано с опечаткой.

Шаг 5. По формулам (13) вычисляем значения v_1, v_2, v_3 и v_4.

Шаг 6. По формулам (12) вычисляем значения u_1, u_2, u_3 и u_4.

Шаг 7. По формулам (11) вычисляем все пять корней уравнения (10).

Замечание. Если коэффициенты a и b связаны равенством

$$256a^5 - 9375b^4 = 0, \tag{15}$$

то резольвента (14) имеет корень $r = 0$, и все формулы алгоритма упрощаются. Например, для уравнения

$$x^5 + 15x + 12 = 0 \tag{16}$$

последовательно получим: $r = 0$, $\varepsilon = -1$, $c = 4/3$, $D = 25/9$, $f = 1$,

$$u_1 = -\left(\frac{21\sqrt{10}}{125} + \frac{3}{5}\right)^{1/5}, \quad u_2 = -\left(\frac{72\sqrt{10}}{125} - \frac{9}{5}\right)^{1/5}, \quad u_3 = \left(\frac{72\sqrt{10}}{125} + \frac{9}{5}\right)^{1/5}, \quad u_4 = \left(\frac{21\sqrt{10}}{125} - \frac{3}{5}\right)^{1/5},$$

и по формуле (11) при $j = 0$ получим единственный ([1], с. 236) действительный корень $x_0 = u_1 + u_2 + u_3 + u_4$.

Равенство (15) можно записать в виде $(a/5)^5 - 3(b/4)^4 = 0$. Это равенство выполняется в том и только том случае, когда $a = 3 \cdot 5^5 k^4$, $b = 3 \cdot 4 \cdot 5^5 k^5$, т. е. $b = 4ka$, где k — любое комплексное число. Так как для $r = 0$ значения величин ε, c, D, v_i и u_i зависят только от коэффициента a и лишь значение величины f зависит от двух коэффициентов a и b, то корни уравнения (10) при условии (15) получаются из корней уравнения (16) умножением на f, для определения которого достаточно выполнить шаг 3 алгоритма. Например, уравнение $x^5 + cx + c = 0$, не разрешимое в радикалах для произвольного c ([9], с. 17), при $c = 3 \cdot 5^5 \cdot 4^{-4}$ имеет своими корнями умноженные на $f = 5/4$ корни уравнения (16). Уравнение $x^5 + \sqrt{3}x - 4\sqrt[8]{75^3}/25 = 0$ имеет своими корнями умноженные на $f = -1/\sqrt[8]{75}$ корни уравнения (16) потому, что для его коэффициентов также выполнено равенство (15). Для уравнения $x^5 + 9375x + 37500i = 0$ с комплексным коэффициентом b по формуле шага 3 получим множитель $f = 5i$.

Еще несколько разложений на множители многочленов пятой степени специального вида будут приведены в следующем пункте.

Уравнение шестой степени

Найдем условия, при которых полином шестой степени в каноническом виде можно представить в виде произведения полиномов третьей степени в каноническом виде

$$x^6 + cx^4 + dx^3 + ex^2 + fx + g = \left(x^3 + px + q\right)\left(x^3 + rx + k\right), \tag{17}$$

где коэффициенты c, d, e, f, g — произвольные комплексные числа. В этом случае уравнение шестой степени

$$x^6 + cx^4 + dx^3 + ex^2 + fx + g = 0 \qquad (18)$$

решается в радикалах, например, одним из способов, изложенных в работе [7]. Оказывается справедлива следующая теорема.

Теорема 1. *Для того чтобы выполнялось равенство (17) необходимо и достаточно, чтобы выполнялось равенство*

$$g\left(c^2 - 4e\right) - cdf + d^2e + f^2 = 0\,. \qquad (19)$$

Доказательство. *Достаточность.* Пусть выполняется равенство (19). Рассмотрим два возможных случая. Если кроме равенства (19) дополнительно выполняется равенство $c^2 - 4e = 0$, то, подставляя в равенство (19) выражение $e = c^2/4$, найдем $f = cd/4$. В этом случае разложение (17) имеет вид

$$\begin{gathered} x^6 + cx^4 + dx^3 + \frac{c^2}{4}x^2 + \frac{cd}{2}x + g = \\ = \left(x^3 + \frac{c}{2}x + \frac{d+v}{2}\right)\left(x^3 + \frac{c}{2}x + \frac{d-v}{2}\right), \end{gathered} \qquad (20)$$

где $v = \sqrt{d^2 - 4g}$. Если выполняется неравенство $c^2 - 4e \neq 0$, то, выражая из равенства (19) коэффициент g, получим разложение

$$\begin{gathered} x^6 + cx^4 + dx^3 + ex^2 + fx + \frac{cdf - d^2e - f^2}{c^2 - 4e} = \\ = \left(x^3 + \frac{c+w}{2}x + \frac{d}{2} + \frac{cd - 2f}{2w}\right)\times \\ \times\left(x^3 + \frac{c-w}{2}x + \frac{d}{2} - \frac{cd - 2f}{2w}\right), \end{gathered} \qquad (21)$$

где $w = \sqrt{c^2 - 4e}$. Справедливость разложения (21) легко проверяется перемножением скобок.

Необходимость. Пусть выполняется равенство (17). Следовательно, множество корней уравнения (18) распадается на две группы, в каждой из которых сумма корней равна нулю. Обозначим корни так: m, n, $-(m+n)$ и s, t, $-(s+t)$ и запишем равенство (17) в виде

$$\begin{gathered} x^6 + cx^4 + dx^3 + ex^2 + fx + g = \\ = (x-m)(x-n)(x+m+n)(x-s)(x-t)(x+s+t). \end{gathered}$$

Раскрывая скобки и приравнивая коэффициенты при одинаковых степенях x, выразим коэффициенты c, d, e, f, g через корни m, n, s и t. С помощью средств компьютерной алгебры получим следующие выражения:

$$\begin{gathered} c = -(m^2 + mn + n^2 + s^2 + st + t^2), \\ d = m^2n + mn^2 + s^2t + st^2, \\ e = (m^2 + mn + n^2)(s^2 + st + t^2), \\ c^2 - 4e = (m^2 + mn + n^2 - s^2 - st - t^2)^2, \\ f - cd/2 = \\ = (m^2 + mn + n^2 - s^2 - st - t^2)(m^2n + mn^2 - s^2t - st^2)/2. \end{gathered} \qquad (22)$$

Рассмотрим два возможных случая. Если выполняется равенство $c^2 - 4e = 0$, то, используя выражения (22), имеем $m^2 + mn + n^2 - s^2 - st - t^2 = 0$, следовательно, выполняется равенство $f - cd/2 = 0$. Из этих двух равенств получим равенство (19). Если выполняется неравенство $c^2 - 4e \neq 0$, то, используя выражения (22), с помощью средств компьютерной алгебры получим равенство $\left(cdf - d^2e - f^2\right)/(c^2 - 4e) = mn(m+n)st(s+t) = g$, т. е. равенство (19).

Следствие 1. При $g = d^2/4$ разложение (20) принимает вид

$$x^6 + cx^4 + dx^3 + \frac{c^2}{4}x^2 + \frac{cd}{2}x + \frac{d^2}{4} = \left(x^3 + \frac{c}{2}x + \frac{d}{2}\right)^2.$$

В этом случае уравнение (18) имеет три двукратных корня.

Следствие 2. График функции

$$y = x^6 + cx^4 + dx^3 + ex^2 + fx + \frac{cdf - d^2e - f^2}{c^2 - 4e}$$

при $e \neq c^2/4$ получается из графика функции $\varphi(x) = x^6 + cx^4 + dx^3 + ex^2 + fx + g$ параллельным переносом вдоль оси Oy. И хотя в общем случае уравнение (18) в радикалах неразрешимо, при $e \neq c^2/4$ можно выразить в радикалах абсциссы точек пересечения графика функции $\varphi(x) = 0$ с графиком прямой линии $y = g - \dfrac{cdf - d^2e - f^2}{c^2 - 4e}$, где коэффициенты c, d, e, f, g — действительные числа.

Следствие 3. Разрешая уравнение (19) относительно переменной f, получим вместо разложений (20) и (21) разложение

$$\begin{gathered} x^6 + cx^4 + dx^3 + ex^2 + \frac{cd \pm vw}{2}x + g = \\ = \left(x^3 + \frac{c+w}{2}x + \frac{d \mp v}{2}\right)\left(x^3 + \frac{c-w}{2}x + \frac{d \pm v}{2}\right), \end{gathered} \qquad (23)$$

которое при $w = \sqrt{c^2 - 4e} = 0$ совпадает с разложением (20), а при $w = \sqrt{c^2 - 4e} \neq 0$ совпадает с разложением (21).

Пример (модулярное уравнение). Рассмотрим уравнение шестой степени

$$x^6 - t^5x^5 + 4tx + t^6 = 0, \qquad (24)$$

симметричное относительно переменных x и t. Это уравнение возникает при решении уравнений пятой

степени с помощью тэта-функций ([1], с. 264) и называется модулярным. Найдем значения параметра t, при которых левую часть этого уравнения можно представить в виде (17) и тем самым выразить корни уравнения через радикалы от параметра t. С помощью подстановки $x = z + t^5/6$ приведем уравнение (24) к каноническому виду (18) и найдем, что условие (19) выполняется только для таких t (кроме $t = 0$), когда $t^{24} = 256$ или $t^{24} = -64$:

$$cdf - d^2e - f^2 - g(c^2 - 4e) =$$
$$= t^2\left(t^3 - 2\right)\left(t^3 + 2\right)\left(t^6 + 4\right)\left(t^8 + 4\right)\left(t^{12} + 16\right)\left(t^{16} - 4t^8 + 16\right)/1024 =$$
$$= t^2\left(t^{24} - 256\right)\left(t^{24} + 64\right)/2^{10} = 0.$$

Это уравнение имеет всего четыре действительных корня: $t_{1,2} = 0$, $t_{3,4} = \pm\sqrt[3]{2}$. По формуле (21) найдем разложение при $t = \sqrt[3]{2}$ и, возвращаясь с помощью подстановки $z = x - t^5/6$ к исходным переменным, для уравнения (24) получим искомое разложение

$$x^6 - 2\sqrt[3]{4}x^5 + 4\sqrt[3]{2}x + 4 =$$
$$= \left(x^3 - \sqrt[3]{4}x^2 + \sqrt[3]{2}\left(\sqrt{5} - 1\right)x - 2\right)\times$$
$$\times\left(x^3 - \sqrt[3]{4}x^2 - \sqrt[3]{2}\left(\sqrt{5} + 1\right)x - 2\right) = 0.$$

Следовательно, уравнение (24) при $t = \sqrt[3]{2}$ разрешимо в радикалах. Теперь с помощью средств компьютерной алгебры находим корни этого уравнения. Приведем здесь выражение в радикалах одного из двух действительных корней

$$x_1 = \left(2\sqrt[3]{4} + \sqrt[3]{50} + 4\sqrt[3]{5} - 3\sqrt[6]{20}\right)/6 \approx 1,45935.$$

Заметим, что в работе [1], с. 267, приведено решение модулярного уравнения (24) при $t = \sqrt[4]{2}$ с помощью тэта-функций.

Следствие 4. Некоторые коэффициенты уравнения (18) могут быть равны нулю. Так, разрешая уравнение (19) относительно переменной e, при $c = 0$ и $d^2 - 4g \neq 0$ получим разложение

$$x^6 + dx^3 - \frac{f^2}{u^2}x^2 + fx + g =$$
$$= \left(x^3 - \frac{f}{u}x + \frac{d+u}{2}\right)\left(x^3 + \frac{f}{u}x + \frac{d-u}{2}\right),$$

где $u = \sqrt{d^2 - 4g}$.

Следствие 5. Если в разложении (21) положим $g = \dfrac{cdf - d^2e - f^2}{c^2 - 4e} = 0$ и разрешим уравнение $cdf - d^2e - f^2 = 0$ относительно переменной f, то получим разложение на множители полинома пятой степени

$$x^5 + cx^3 + dx^2 + ex + \frac{d(c \pm w)}{2} =$$
$$= \left(x^2 + \frac{c \pm w}{2}\right)\left(x^3 + \frac{c \mp w}{2}x + d\right), \qquad (25)$$

где $w = \sqrt{c^2 - 4e}$. Разложение (25) проще получить из разложения (23), полагая $g = 0$. Если разрешить уравнение $cdf - d^2e - f^2 = 0$ относительно переменной e, то при $d \neq 0$ получим разложение

$$x^5 + cx^3 + dx^2 + \frac{cdf - f^2}{d^2}x + f =$$
$$= \left(x^2 + \frac{f}{d}\right)\left(x^3 + \frac{cd - f}{d}x + d\right).$$

Следствие 6. В приведенных выше разложениях в качестве коэффициентов можно выбирать произвольные функции, в том числе функции нескольких переменных, в частности, полиномы. Например, если в разложении (21) положить $c = 3$, $e = 2$, а затем в полученном разложении

$$x^6 + 3x^4 + dx^3 + 2x^2 + fx - 2d^2 + 3df - f^2 =$$
$$= \left(x^3 + x - d + f\right)\left(x^3 + 2x + 2d - f\right) \qquad (26)$$

сделать замену коэффициентов

$$d = x^4 - 2x^3 + (k+1)x^2 + (p-3)x + m + n,$$
$$f = x^4 - 3x^3 + (k+2)x^2 + (p-4)x + m + 2n,$$

где $k = p(rs - pt)/s^2$, $m = pt/s$ и $n = s/p$, то с помощью компьютерной алгебры получим разложение на множители полинома шестой степени частного вида

$$x^6 + \frac{s^3 + p^2(rs - pt)}{ps^2}x^4 + px^3 + rx^2 + sx + t =$$
$$= \left(x^2 + \frac{s}{p}\right)\left(x^4 + \frac{p(rs - pt)}{s^2}x^2 + px + \frac{pt}{s}\right), \qquad (27)$$

в котором коэффициенты p, r, s, t — произвольны $(ps \neq 0)$. Для сравнения этого разложения с разложением (21) обозначим в разложении (27) $p = d$, $r = e$, $s = f$. Коэффициент при x^4 обозначим через c и разрешим полученное равенство $c = \left(s^3 + p^2(rs - pt)\right)/ps^2$ относительно t. После указанных подстановок разложение (27) примет вид

$$x^6 + cx^4 + dx^3 + ex^2 + fx + \frac{f\left(f^2 - cdf + d^2e\right)}{d^3} =$$
$$= \left(x^2 + \frac{f}{d}\right)\left(x^4 + \frac{cd - f}{d}x^2 + dx + \frac{f^2 - cdf + d^2e}{d^2}\right). \qquad (28)$$

Если для коэффициентов уравнения (18) выполняется равенство $g = f\left(f^2 - cdf + d^2e\right)/d^3$ и равенство $g = -\left(f^2 - cdf + d^2e\right)/(c^2 - 4e)$, то справедливы раз-

ложения (28) и (21), причем $w = \sqrt{c^2 - 4e} = \sqrt{-d^3/f}$. Легко проверить, что в этом случае $x = +\sqrt{-f/d}$ является корнем уравнения $x^3 + \frac{c-w}{2}x + \frac{d}{2} - \frac{cd-2f}{2w} = 0$, а $x = -\sqrt{-f/d}$ является корнем уравнения $x^3 + \frac{c+w}{2}x + \frac{d}{2} + \frac{cd-2f}{2w} = 0$, следовательно, уравнение (18) решается в квадратных радикалах.

Следствие 7. Сделаем следующую подстановку в разложении (26): $d = x^6 + (c-2)x^3 + mx^2 + nx + j$, $f = x^6 + (c-3)x^3 + (m+1)x^2 + (a+n-1)x + b + j$, где $m = d - ac + 1$, $n = a^2c + a - ad - bc + e - 3$ и $j = v - a^3c + a^2d + 2abc - ae + b - bd$. В результате с помощью средств компьютерной алгебры получим разложение на множители полинома восьмой степени частного вида

$$x^8 + ax^7 + bx^6 + cx^5 + dx^4 + ex^3 + vx^2 + gx + bh = \\ = (x^2 + ax + b)\times \\ \times\left(x^6 + cx^3 + (d - ac)x^2 + (a^2c - ad - bc + e)x + h\right), \quad (29)$$

где $g = ah + b(a^2c - ad - bc + e)$, $h = v - a^3c + a^2d + 2abc - ae - bd$, а коэффициенты a, b, c, d, e, v — произвольны.

Если в разложении (26) положить

$$d = x^6 + kx^4 - (k+2)x^3 + mx^2 + nx + j,$$
$$f = x^6 + kx^4 - (k+3)x^3 + (m+1)x^2 + nx + j - k,$$

то получим разложение на множители полинома восьмой степени специального вида

$$x^8 + x^7 - \left(k^2 + k - m + 1\right)x^4 + \left(k^2 + m + n + 1\right)x^3 + \\ + (j + 2k - km + n + 2)x^2 + (j - kn - k)x - jk - k^2 = \\ = (x^2 + x - k)\times \\ \times\left(x^6 + kx^4 - kx^3 + (m-1)x^2 + (n+2)x + j + k\right),$$

отличающееся от разложения (29). Из последнего разложения при $k = -1$, $m = 1$, $n = -3$, $j = 2$ получим разложение $x^8 + x^7 + 1 = (x^2 + x + 1)(x^6 - x^4 + x^3 - x + 1)$, которое является решением олимпиадной задачи (см. [10], стр. 74, № 854).

Теорема 2. *Обобщенное возвратное уравнение шестой степени*

$$x^6 + bx^5 + cx^4 + dx^3 + chx^2 + bh^2x + h^3 = 0 \quad (30)$$

имеет корни $\alpha\beta$, $\alpha\gamma$, $\alpha\delta$, $\beta\gamma$, $\beta\delta$, $\gamma\delta$, *где* α, β, γ, δ — *корни уравнения четвертой степени*

$$z^4 + sz^3 - bz^2 + qz + h = 0, \quad (31)$$

коэффициенты которого s *и* q *находятся из системы уравнений*

$$sq - c - h = 0, \quad q^2 + d + h(s^2 + 2b) = 0. \quad (32)$$

Доказательство. Запишем с помощью формул Виета коэффициенты уравнения (31) через корни: $s = -\alpha - \beta - \gamma - \delta$, $b = -\alpha\beta - \alpha\gamma - \alpha\delta - \beta\gamma - \beta\delta - \gamma\delta$, $q = -\alpha\beta\gamma - \alpha\beta\delta - \alpha\gamma\delta - \beta\gamma\delta$, $h = \alpha\beta\gamma\delta$. Теперь так же выразим с помощью средств компьютерной алгебры коэффициенты следующего уравнения шестой степени

$$(x - \alpha\beta)(x - \alpha\gamma)(x - \alpha\delta)(x - \beta\gamma)(x - \beta\delta)(x - \gamma\delta) = \\ = x^6 + px^5 + mx^4 + nx^3 + kx^2 + lx + r = 0 \quad (33)$$

через его корни. Легко видеть, что коэффициенты p и r разложения (33) выражаются через коэффициенты уравнения четвертой степени (31) так: $p = b$, $r = h^3$. Кроме того, с помощью средств компьютерной алгебры убеждаемся, что коэффициенты m, n, k и l уравнения (33) также однозначно находятся по коэффициентам s, b, q и h уравнения (31):

$$m = sq - h, \quad n = -q^2 - h(s^2 + 2b), \quad k = hm, \quad l = h^2b. \quad (34)$$

Сравнивая выражения (34) с равенствами (32) и коэффициентами уравнения (30), находим, что $c = m$, $d = n$, $ch = k$, $bh^2 = l$, т. е. коэффициенты уравнений (30) и (33) равны. Следовательно, уравнение (30) имеет те же корни, что и уравнение (33). Теорема доказана.

Замечание. Система двух уравнений (32) с двумя неизвестными s и q сводится к биквадратному уравнению, например, относительно s:

$$hs^4 + (d + 2bh)s^2 + (c + h)^2 = 0. \quad (35)$$

Таким образом, если выполнены условия теоремы, то уравнение шестой степени (30) решается в радикалах. Отметим, что идея этой теоремы навеяна олимпиадной задачей (см. [11], стр. 63, № 21.7).

Заметим, что исходное уравнение шестой степени (30) восстанавливается по уравнению четвертой степени (31) однозначно. Однако уравнение (31), которое строится по уравнению (30), в общем случае может иметь два существенно различных набора корней в зависимости от корней квадратного относительно s^2 уравнения (35). Например, для уравнения шестой степени

$$x^6 - 35x^5 + 476x^4 - 3220x^3 + 11\,424x^2 - \\ - 20\,160x + 13\,824 = 0$$

получим уравнение (31) в виде $z^4 + sz^3 + 35z^2 + qz + 24 = 0$, где s — любой корень биквадратного уравнения (35), а q находится из первого соотношения (32). Так, из уравнения $24s^4 - 4900s^2 + 250000 = 0$ находим $s_{1,2} = \pm 10$, $s_{3,4} = \pm 25/\sqrt{6}$, а из соотношения $q = (c + h)/s$ находим $q_{1,2} = \pm 50$, $q_{3,4} = \pm 20\sqrt{6}$. Поэтому для определения корней α, β, γ, δ можно выбрать уравнение (31) в одном из следующих четырех видов:

$z^4 \pm 10z^3 + 35z^2 \pm 50z + 24 = 0$, $z^4 \pm 25/\sqrt{6}\, z^3 + 35z^2 \pm \pm 20\sqrt{6}z + 24 = 0$. Корни первых двух уравнений находятся просто: $\alpha_{1,2} = \mp 1$, $\beta_{1,2} = \mp 2$, $\gamma_{1,2} = \mp 3$, $\delta_{1,2} = \mp 4$. Однако для корней последних двух уравнений получаем громоздкие символьные выражения, приближенные численные значения которых равны: $\alpha_{1,2} \approx \mp 1,225$; $\beta_{1,2} \approx \mp 1,633$; $\gamma_{1,2} \approx \mp 2,449$; $\delta_{1,2} \approx \mp 4,899$. В этом случае для попарных произведений этих приближенных значений, т. е. для корней исходного уравнения шестой степени, находим приближенные значения: 2,000; 3,000; 3,999; 6,001; 8,000; 12,00.

Следствие 1. *Возвратное уравнение шестой степени*

$$x^6 + bx^5 + cx^4 + dx^3 + cx^2 + bx + 1 = 0$$

имеет корни $\alpha\beta$, $\alpha\gamma$, $\alpha\delta$, $\beta\gamma$, $\beta\delta$, $\gamma\delta$, *где* α, β, γ, δ — *корни уравнения четвертой степени*

$$z^4 + sz^3 - bz^2 + (c+1)/sz + 1 = 0,$$

а коэффициент $s \neq 0$ *находится из биквадратного уравнения*

$$s^4 + (d + 2b)\, s^2 + (c+1)^2 = 0.$$

Так как теорема справедлива для любых h, то, полагая в уравнении (30) $h = 0$, получим следующее утверждение.

Следствие 2 (новый способ решения кубического уравнения). *Уравнение третьей степени*

$$x^3 + bx^2 + cx + d = 0 \qquad (36)$$

тогда и только тогда имеет корни $\alpha\beta$, $\alpha\gamma$, $\beta\gamma$, *когда* α, β, γ *являются корнями уравнения*

$$z^3 \pm c/\sqrt{-d}\, z^2 - bz \pm \sqrt{-d} = 0. \qquad (37)$$

Интересно отметить, что дискриминант D_2 уравнения (37) связан с дискриминантом D_1 уравнения (36) равенством $D_2 = -D_1/d$, поэтому при положительных d дискриминанты уравнений (36) и (37) имеют разные знаки. Данное следствие предлагает новый способ [7] решения уравнений третьей степени, так как в некоторых случаях решить уравнение (37) оказывается проще, чем решить исходное уравнение (36). Например, уравнение $x^3 + 7/\sqrt{15}\, x^2 + x + \sqrt{15}/9 = 0$ имеет два комплексных и один действительный корень, для которого пакет прикладных программ Mathematica 8.0 вырабатывает громоздкое точное выражение (приближенно равное $-1,291$) с использованием корней шестой степени из двучленов, содержащих квадратные корни. Связанное с ним уравнение (37) с комплексными коэффициентами $z^3 - i\sqrt[4]{27/5}z^2 - 7/\sqrt{15}\, z + i\sqrt[4]{5/27} = 0$ имеет три комплексных корня $z_1 = i/\sqrt[4]{15}$, $z_{2,3} = (i \pm 2)/\sqrt[4]{15}$, а произведение двух последних корней равно точному значению действительного корня исходного уравнения $z_2 z_3 = -\sqrt{15}/3 \approx -1,291$, т. е. выражается всего лишь через один квадратный радикал. Рассмотрим кубическое уравнение $x^3 - 3x^2 - 2x + \sqrt{129}/9 + 1 = 0$, имеющее три действительных корня. После приведения связанного с ним уравнения (37) к каноническому виду получим уравнение $t^3 + t\left(\sqrt{129} + 3\right)/4 = 0$, с помощью которого находим точные выражения корней исходного уравнения через квадратные радикалы: $x_1 = \left(\sqrt{129} + 9\right)/6$, $x_{2,3} = \left(9 - \sqrt{129} \pm 3\sqrt{34 - 2\sqrt{129}}\right)/12$.

Теорема 3 (еще один новый способ решения кубического уравнения). *Уравнение третьей степени*

$$t^3 + bt^2 + ct + d = 0 \qquad (38)$$

тогда и только тогда имеет корни $\alpha\beta + 1/\alpha\beta$, $\alpha\gamma + 1/\alpha\gamma$, $\alpha\delta + 1/\alpha\delta$, *когда* α, β, γ, δ *являются корнями уравнения четвертой степени*

$$z^4 + sz^3 - bz^2 + (c+4)/s\, z + 1 = 0, \qquad (39)$$

где коэффициент $s \neq 0$ находится из биквадратного уравнения

$$s^4 + (d + 4b)\, s^2 + (c+4)^2 = 0. \qquad (40)$$

Для доказательства этой теоремы сделаем в уравнении (38) подстановку $t = x + 1/x$ и умножим результат на x^3. Получим возвратное уравнение шестой степени $x^6 + bx^5 + (c+3)x^4 + (d+2b)x^3 + (c+3)x^2 + bx + 1 = 0$. Затем применим следствие 1 теоремы 2. Заметим, что уравнение (39) является возвратным тогда и только тогда, когда $d + 4b = 2(c+4)$ или $d + 4b = -2(c+4)$. В этом случае один из корней уравнения (38) равен 2 или, соответственно, -2, т. е. корни уравнения (38) выражаются через квадратные радикалы. Примерами таких кубических уравнений являются уравнения $x^3 + cx \pm 2(c+4) = 0$, где c — произвольное комплексное число, в том числе уравнения $x^3 - x - 6 = 0$ и $x^3 - 19x + 30 = 0$, подробно рассмотренные в работе [7].

Предлагаемый способ может оказаться полезным, когда решение кубических уравнений стандартными способами затруднительно [7]. Например, для кубического уравнения

$$t^3 - \sqrt{3}t^2 + \left(\sqrt{3} - 9/2\right)t + 9\sqrt{3}/2 - 29/8 = 0,$$

имеющего отрицательный дискриминант (неприводимый случай), находим из биквадратного уравнения $s = \sqrt{2}$ и, приведя уравнение (39) с помощью подстановки $z = x - \sqrt{2}/4$ к каноническому виду, получим биквадратное уравнение $x^4 + \left(\sqrt{3} - 3/4\right)x^2 - \sqrt{3}/8 + 69/64 = 0$. Наконец, для исходного кубического уравнения находим выражения корней $t_1 = \sqrt{3} - 1/2$, $t_{2,3} = 1/4 \pm \sqrt{69/16 - \sqrt{3}/2}$, записанные лишь через квадратные радикалы.

С помощью теоремы 3 получим следующий способ решения алгебраического уравнения четвертой степени.

Следствие. Новый алгоритм решения уравнения четвертой степени.

Произвольное уравнение четвертой степени можно привести к виду (39) делением на свободный член и заменой переменной, поэтому будем считать, что исходное уравнение имеет вид (39). Из уравнения (40) находим $d = -s^2 - (c+4)^2/s^2 - 4b$ и решаем уравнение (38), т. е. получим значения корней $w_1 = \alpha\beta + 1/\alpha\beta$, $w_2 = \alpha\gamma + 1/\alpha\gamma$, $w_3 = \alpha\delta + 1/\alpha\delta$. Затем, для вычисления значений величин $\alpha\beta$, $\alpha\gamma$, $\alpha\delta$ решаем три квадратных уравнения $v + 1/v = w_i$, $i = 1, 2, 3$. Тогда корни уравнения (39) находятся среди корней следующих восьми квадратных уравнений: $z^2 + sz + v_1 + v_2 + v_3$, где v_i — один из корней квадратного уравнения $v + 1/v = w_i$, $i = 1, 2, 3$. Заметим, что если корни кубического уравнения (38) выражаются через квадратные радикалы, то корни уравнения четвертой степени (39) также выражаются через квадратные радикалы.

Несколько разложений частного вида

1. Уравнение третьей степени вида $z^3 + az^2 + bz + a(bk^2 - ak + a)/k^3 = 0$ имеет корень $z = -a/k$, т. е. разрешимо в квадратных радикалах.

2. Уравнение третьей степени вида $z^3 + az^2 + bz + b^2(b - ak + k^2)/k^3 = 0$ имеет корень $z = -b/k$, т. е. разрешимо в квадратных радикалах.

3. Для любых комплексных a и b справедливо разложение

$$x^3 - 3a^2bx \pm a^3(b^3 + 1) = \\ = [x \pm a(b+1)]\left[x^2 \mp a(b+1)x + a^2(b^2 - b + 1)\right]. \quad (41)$$

Отсюда при $b = 1$ получим разложение $x^3 - 3a^2x \pm 2a^3 = (x \pm 2a)(x \mp a)^2$. Из разложения (41) при $a = \sqrt[3]{q/3}$, $b = \sqrt[3]{2}$ получим разложение на множители кубического многочлена $x^3 - \sqrt[3]{6q^2}x \pm q$, а при $a = \sqrt{-p/\sqrt[3]{54}}$, $b = \sqrt[3]{2}$ получим разложение на множители многочлена $x^3 + px \pm \sqrt{-p^3/6}$.

4. Для любых комплексных a и r справедливо разложение

$$x^4 + 2ax^2 + r\sqrt{\pm r}x \pm ra + a^2 = \\ = \left(x^2 \pm \sqrt{\pm r}x + a\right)\left(x^2 \mp \sqrt{\pm r}x \pm ra + a\right).$$

5. Если в многочлене $x^6 + cx^4 + dx^3 + ex^2 + fx + g$ коэффициенты связаны соотношениями $f = d(4c-1)/4 - k(16e-1)/16$, $g = dk/4 + (16e-1)/64$, где $k = \pm\sqrt{1/2 - c}$, то справедливо разложение

$$x^6 + cx^4 + dx^3 + ex^2 + fx + g = \\ = \left(x^2 - kx + 1/4\right)\left(x^4 + kx^3 + x^2/4 + dx + 4g\right). \quad (42)$$

6. Легко проверить, что разложение (42) остается справедливым, если коэффициенты c, f и k выразить через свободные коэффициенты d, e и g так: $c = 1/2 - k^2$, $f = d/4 - 4gk$, $k = (1 - 16e + 64g)/16d$. Тогда, полагая $g = 0$, получим разложение на множители многочлена пятой степени:

$$x^5 + cx^3 + dx^2 + ex + f = \\ = \left(x^2 - kx + 1/4\right)\left(x^3 + kx^2 + x/4 + d\right),$$

где $c = 1/2 - k^2$, $f = d/4$, $k = (1 - 16e)/16d$.

7. Если в многочлене $x^6 + bx^5 + cx^4 + dx^3 + ex^2 + fx + g$ коэффициенты связаны соотношениями $f = mc/2$, $g = nk$, где $m = -(bc - 2d)/2$, $n = (c^2 - 4e)/(4b)$, $k = (2b^2c - 4bd - c^2 + 4e)/(4b)$, то справедливо разложение

$$x^6 + bx^5 + cx^4 + dx^3 + ex^2 + mcx/2 + nk = \\ = \left(x^3 + cx/2 - n\right)\left(x^3 + bx^2 + cx/2 - k\right). \quad (43)$$

Если $e = c^2/4$, то $n = 0$, $g = 0$ и из разложения (43) получим разложение на множители многочлена пятой степени

$$x^5 + bx^4 + cx^3 + dx^2 + c^2x/4 + mc/2 = \\ = \left(x^2 + c/2\right)\left(x^3 + bx^2 + cx/2 + m\right),$$

где $m = -(bc - 2d)/2$.

Заключение

В практически важных задачах часто возникают именно такие кубические уравнения, все три корня которых являются вещественными числами. А в этом случае известные способы решения (формулы Ферро-Тартальи, формулы Феррари-Кардано и др.) оказываются малопродуктивными [7]. В работе [12], например, доказывается, что однотипные величины треугольника (стороны, высоты, синусы углов и т. п.) являются корнями соответствующего кубического уравнения, коэффициенты которого выражаются через полупериметр p треугольника, радиус r вписанной в треугольник и радиус R описанной около треугольника окружности. Так, стороны треугольника являются корнями кубического уравнения $x^3 - 2px^2 + (p^2 + r^2 + 4R)x - 4pRr = 0$, а тангенсы половинных углов являются корнями уравнения $px^3 - (4R + r)x^2 + px - r = 0$, и стандартными способами эти действительные величины выражаются через комплексные числа. Решение уравнений четвертой степени обычно сводится к решению уравнений третьей степени, а к решению этих уравнений приводит решение уравнений более высоких степеней.

Основная часть данной работы посвящена разложениям на множители полиномов пятой и, особенно, шестой степени специального вида. Исследование символьных решений возвратного уравнения шестой степени привело к построению новых алгоритмов для символьных решений уравнений третьей и четвертой степеней. Особое внимание в работе уделялось символьному выражению корней уравнений через квадратные радикалы. Найденные способы

решения уравнений третьей степени (сведением к решению уравнений четвертой степени) и четвертой степени (сведением к решению уравнений третьей степени) отличаются дискриминантами и резольвентами от дискриминантов и резольвент известных способов [7]. Поэтому интересно построить итерационный процесс из различных способов решения этих уравнений и выяснить, приводит ли он за конечное число итераций к легко разрешимому уравнению. Таким образом, появляется возможность построить более универсальный алгоритм решения таких уравнений, чем известные алгоритмы.

Представленные результаты могут служить основой при проектировании программ, дополняющих имеющиеся пакеты прикладных программ простыми алгоритмами в части точного символьного решения алгебраических уравнений третьей, четвертой степеней и некоторых специальных видов уравнений более высоких степеней.

Список литературы

1. **Прасолов В. В., Соловьев Ю. П.** Эллиптические функции и алгебраические уравнения. М.: Факториал, 1997. 288 с.
2. **Кутищев Г. П.** Решение алгебраических уравнений произвольной степени: теория, методы, алгоритмы. М.: Изд-во URSS, 2010. 232 с.
3. **Прасолов В. В.** Многочлены. М.: МЦНМО, 1999. 336 с.
4. **Шмойлов В. И., Хисамутдинов М. В., Кириченко Г. А.** Решение алгебраических уравнений методом Рутисхаузера-Никипорца // Вестник НГУ. Серия: Математика, механика, информатика. 2015. Т. 15. № 1. С. 63—79.
5. **Михалкин Е. Н.** О решении общих алгебраических уравнений с помощью интегралов от элементарных функций // Сиб. мат. журн. 2006. Т. 47. № 2. С. 365—371.
6. **Зеленова М. Е.** Решение полиномиальных уравнений в поле алгебраических чисел // Вестн. Моск. ун-та. Серия 1. Математика, механика. 2014. № 1. С. 25—29.
7. **Астапов И. С., Астапов Н. С.** Решение алгебраических уравнений третьей и четвертой степеней с помощью компьютерной алгебры // Программная инженерия. 2014. № 10. С. 33—42.
8. **Никифоровский В. А.** Из истории алгебры XVI—XVII вв. М.: Наука, 1979. 208 с.
9. **Математика в** понятиях, определениях и терминах. Часть I. М.: Просвещение, 1978, 320 с.
10. **Вышенский В. А., Карташов Н. В., Михайловский В. И., Ядренко М. И.** Сборник задач киевских математических олимпиад. Киев: Вища школа, Изд-во при Киев. ун-те, 1984. 240 с.
11. **Зарубежные** математические олимпиады. / Под ред. И. Н. Сергеева. М.: Наука, 1987. (Б-ка мат. кружка). 416 с.
12. **Солтан В. П., Мейдман С. И.** Тождества и неравенства в треугольнике. Кишинев: Штиинца, 1982. 60 с.

Algorithms for Symbolic Solving of Algebraic Equations

I. S. Astapov, velais@imec.msu.ru, Institute of Mechanics Lomonosov Moscow State University, Moscow, 119192, Russian Federation, **N. S. Astapov,** nika@hydro.nsc.ru, Lavrentyev Institute of Hudrodynamics of Siberian Branch of Russian Academy of Sciences, Novosibirsk, 630090, Russian Federation

Corresponding author:

Astapov Nikolay St., Senior Researcher, Lavrentyev Institute of Hudrodynamics SB RAS, Novosibirsk, 630090, Russian Federation,
E-mail: nika@hydro.nsc.ru

Received on June 08, 2017
Accepted on June 19, 2017

In physical and technical simulations there is often a need to express symbolically (not numerically) the roots of algebraic equations as functions of the symbolic coefficients of these equations for a subsequent study of the phenomena of interest. In many technical problems we have a cubic equation with three real roots. In this case the well-known solutions (Ferro-Tartaglia`s and Ferrari-Cardano`s formulas) are difficult to implement accurately, since there is no efficient algorithm for finding the cube roots of a complex number. The general polynomial equations of degree five or higher have no solutions in radicals, although they can be expressed in terms of generalized hypergeometric functions of the coefficients of these equations. However, some special types of equations, for example, $x^n - a = 0$, can be solved in radicals. Therefore, it is important to find simple algorithms to solve algebraic equations, in particular, of degree three to eight.

The bulk of this work is devoted to factorizations of special polynomials of degree five and six. A modular equation is considered. A polynomial of degree six with a single coefficient parametrically dependent of the other arbitrary coefficients is factorized. Factorizations are found for some polynomials of higher degree. A study of symbolic solutions to a reciprocal equation has resulted in some new algorithms for solving equations of degree three and four. These algorithms are based on solutions to equations with discriminants and resolvents different from those of the original equations. New methods of reducing the equations of degree three and four to reciprocal equations are proposed. Particular attention is given to equations solved by square radicals. The performance of these methods is shown by a comparison with solutions generated in the software system Mathematica.

These results can be used in the design of simple computer programs to be used as supplementary to the programs of exact symbolic solving of algebraic equations available in the conventional software.

Keywords: *reciprocal equations, solution in radicals, Cardano's formula, resolvent, De Moivre's polynomial, modular equation, computer algebra, software*

For citation:

Astapov I. S., Astapov N. S. Algorithms for Symbolic Solving of Algebraic Equations, *Programmnaya Ingeneria,* 2017, vol. 8, no. 9, pp. 422—432.

DOI: 10.17587/prin.8.422-432

References

1. **Prasolov V. V., Solov'ev Iu. P.** *Ellipticheskie funktsii i algebraicheskie uravneniia* (Elliptic functions and algebraic equations), Moscow, Faktorial, 1997, 288 p. (in Russian).

2. **Kutishchev G. P.** *Reshenie algebraicheskikh uravnenii proizvol'noi stepeni: teoriia, metody, algoritmy* (Solution of algebraic equations of arbitrary degree: theory, methods, algorithms), Moscow, Izd-vo URSS, 2010, 232 p. (in Russian).

3. **Prasolov V. V.** *Mnogochleny* (Polynomials), Moscow, MCNMO, 1999, 336 p. (in Russian).

4. **Shmoilov V. I., Khisamutdinov M. V., Kirichenko G. A.** Reshenie algebraicheskikh uravnenii metodom Rutiskhauzera-Nikiportsa (Solution of algebraic equations by the Rutischauser-Nikiportz method), *Vestnik NGU. Series: Mathematics, mechanics, computer science,* 2015, vol. 15, no. 1, pp. 63—79 (in Russian).

5. **Mikhalkin E. N.** O reshenii obshchikh algebraicheskikh uravnenii s pomoshch'iu integralov ot elementarnykh funktsii (On the solution of general algebraic equations by means of integrals of elementary functions), *Sib. mat. zhurnal,* 2006, vol. 47, no. 2, pp. 365—371 (in Russian).

6. **Zelenova M. E.** Reshenie polinomial'nykh uravnenii v pole algebraicheskikh chisel (Solving polynomial equations over the field of algebraic numbers), *Vestn. Mosk. Un-ta. Series 1. Mathematics, mechanics,* 2014, no. 1, pp. 25—29 (in Russian).

7. **Astapov I. S., Astapov N. S.** Reshenie algebraicheskikh uravnenii tret'ei i chetvertoi stepenei s pomoshch'iu komp'iuternoi algebry (Solving third- and fourth-order algebraic equations by methods of competer algebra), *Programmnaya Ingeneria,* 2014, no. 10, pp. 33—42 (in Russian).

8. **Nikiforovskii V. A.** *Iz istorii algebry XVI—XVII vv.* (From the history of algebra XVI — XVII centuries), Moscow, Nauka, 1979, 208 p. (in Russian).

9. ***Matematika*** *v poniatiiakh, opredeleniiakh i terminakh* (Mathematics in terms, definitions and terms). Part I. Moscow, Prosveshchenie, 1978. 320 p. (in Russian).

10. **Vyshenskii V. A., Kartashov N. V., Mikhailovskii V. I., Iadrenko M. I.** *Sbornik zadach kievskikh matematicheskikh olimpiad* (Collection of problems of the Kiev Mathematical Olympiads) Kiev, Vishcha shkola, Izdatel'stvo pri Kievskom universitete, 1984, 240 p. (in Russian).

11. ***Zarubezhnye*** *matematicheskie olimpiady* (Foreign Mathematical Olympiads) / Editor I. N. Sergeev. Moscow, Nauka, 1987, 416 p. (in Russian).

12. **Soltan V. P., Meidman S. I.** *Tozhdestva i neravenstva v treugol'nike* (Identities and inequalities in the triangle), Kishinev, Shtiinca, 1982, 60 p. (in Russian).

ООО "Издательство "Новые технологии". 107076, Москва, Стромынский пер., 4
Технический редактор *Е. М. Патрушева.* Корректор *И. Е. Назарова*

Сдано в набор 10.07.2017 г. Подписано в печать 18.08.2017 г. Формат 60×88 1/8. Заказ PI917
Цена свободная.

Оригинал-макет ООО "Адвансед солюшнз". Отпечатано в ООО "Адвансед солюшнз".
119071, г. Москва, Ленинский пр-т, д. 19, стр. 1. Сайт: **www.aov.ru**