ISSN 2220-3397

ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Программная инженерия

Том 14. № 8. 2023

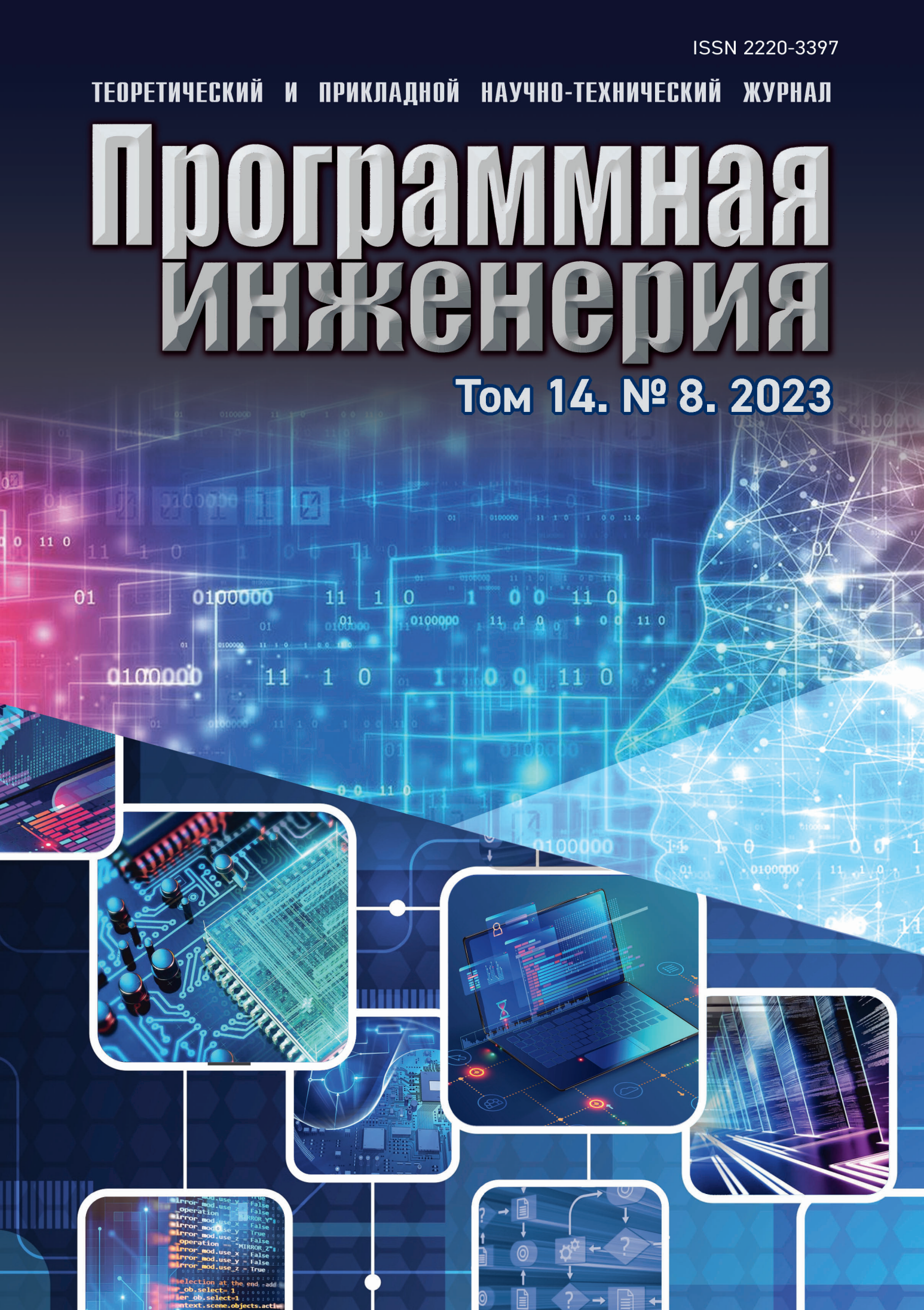


Рисунок к статье **Б. А. Позина, А. Е. Гараниной, Е. А. Ивановой**
«АВТОМАТИЗИРОВАННАЯ ОЦЕНКА ПРОГРЕССА И «ЗДОРОВЬЯ» ПРОЕКТА НА ОСНОВЕ СТАНДАРТА OMG ESSENCE»

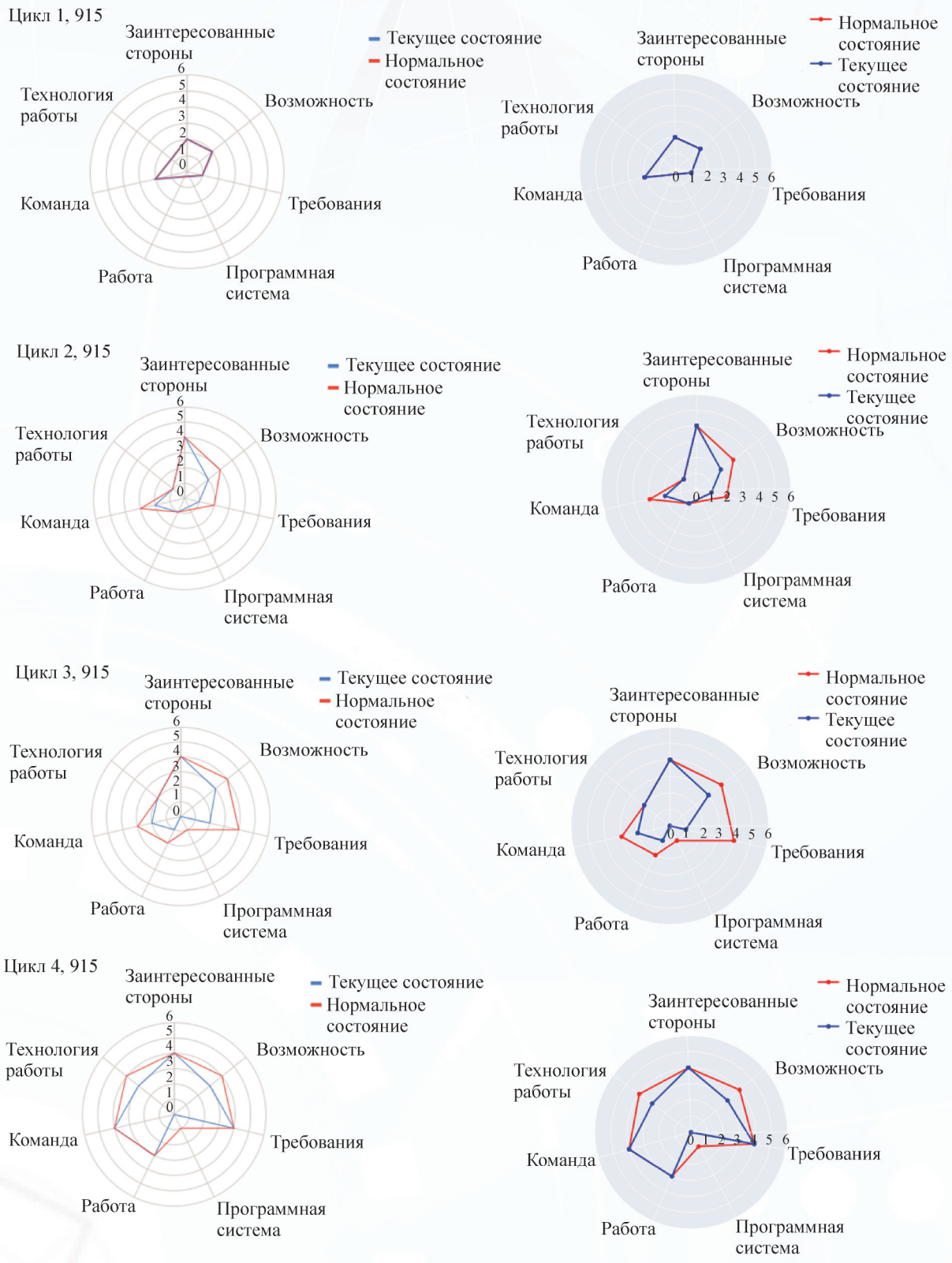


Рис. 3. Лепестковые диаграммы для циклов проекта (начало)

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

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 подписных агентствах (индекс по Объединенному каталогу "Пресса России" — **22765**) или непосредственно в редакции (для юридических лиц).
Тел.: (499) 270-16-52.
Http://novtex.ru/prin/rus E-mail: prin@novtex.ru
Журнал включен в Российский индекс научного цитирования (РИНЦ) и Russian Science Citation Index (RSCI).
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. 14
N 8
2023

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

CONTENTS

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

УДК 658.511.3
DOI: 10.17587/prin.14.367-376

Б. А. Позин[1,2,3], д-р техн. наук, проф., bpozin@ec-leasing.ru,
А. Е. Гаранина[1], магистр, aegaranina@edu.hse.ru,
Е. А. Иванова[1], магистр, eaivanova_23@edu.hse.ru
[1] Научно-исследовательский университет «Высшая школа экономики», Москва
[2] Институт системного программирования РАН, Москва
[3] ЗАО ЕС-лизинг, Москва

Автоматизированная оценка прогресса и «здоровья» проекта на основе стандарта OMG ESSENCE

Поступила в редакцию 15.05.2023
Принята к публикации 20.06.2023

Представлена разработанная авторами автоматизированная технология оценки состояния прогресса и «здоровья» в рамках выполнения программного проекта на основе стандарта OMG ESSENCE. В основе такой оценки лежит степень достижения нормальных значений определенных семи сущностей, являющихся основанием выполнения любого программного проекта. Значения сущностей определяются в стандарте ESSENCE. Предложен метод оценки состояния прогресса и «здоровья» проекта, который позволяет автоматизировать получение подобных оценок на каждой итерации его реализации на основе анализа чек-листов, заполненных руководителем проекта. В случае отклонения результатов от нормальных значений предложен метод выявления зависимостей на основании анализа графа взаимосвязей, построенного по ядру ESSENCE. Разработан сервис для автоматизации оценки, приведена статистика внедрения технологии, получены оценки ошибок первого и второго рода.

Ключевые слова: *ESSENCE, качество программного проекта, прогресс проекта, «здоровье» проекта, граф взаимосвязей, предикатная модель оценки результатов итерации, автоматизация оценки качества программного проекта, SEMAT*

Для цитирования:

Позин Б. А., Гаранина А. Е., Иванова Е. А. Автоматизированная оценка прогресса и «здоровья» проекта на основе стандарта OMG ESSENCE // Программная инженерия. 2023. Том 14, № 8. С. 367—376. DOI: 10.17587/prin.14.367-376.

Введение

При разработке программных приложений, особенно не очень опытными командами, возникают вопросы оценки качества не только окончательного, но и промежуточных результатов разработки, а также вопросы обучения команды тому, чего следует добиваться при создании такого приложения. Трудности начинающей (да и не только начинающей) команды состоят в том, что разработчиками еще не накоплен достаточный опыт и критерии для оценки состояния проекта как степени выполнения целевой задачи. Особенно это важно при итерационной технологии создания приложения. Команда может разработать довольно большое количество строк кода, но не проработать потребности заинтересованных сторон заказчика и требования к приложению. В результате такой код окажется не востребован, а ресурсы потрачены впустую. Существующие на настоящее время методики разработки в основном упускают этот момент.

Анализ различных методов и методологий оценки качества проекта [1—8] показал, что стандарт OMG ESSENCE [9, 10], разработанный международным сообществом SEMAT [9—12] под руководством Ивара Якобсона [13, 14], наиболее близок к решению задачи создания технологии командной разработки приложения, которая дает возможность оценить количество полученных ре-

зультатов и их правильность с позиций достижения ожидаемых результатов. В стандарте предусмотрено выделение семи сущностей, которые характеризуют участников процесса, основные промежуточные результаты и ожидаемые состояния сущностей при любой выбранной командой технологии ведения работ. В соответствии с работами [9, 10] сущность — это элемент, который для контроля успеха разработки требует отслеживания результатов работ по изменению последовательности состояний этих элементов. Эти изменения проверяются ответами участников разработки на контрольные вопросы.

Стандарт определяет два важных понятия:

- *прогресс проекта*, т. е. количественные изменения созданных в проекте сущностей;
- *«здоровье» проекта*, т. е. соответствие созданных в проекте сущностей ожидаемым значениям, правильное развитие проекта, отражаемое в ожидаемых значениях, принимаемых сущностями на каждом этапе (итерации) его реализации.

Работы [2, 3] рассматривают состояния сущностей как возможность на каждом этапе (итерации) его выполнения принимать общими усилиями команды согласованные решения о том, в каком состоянии находится разработка и как правильно развивать проект. Однако в промышленных условиях разработки такая методика может быть не основной, а только вспомогательной для сложных случаев. При заказной разработке, как правило, перед началом проекта выбирают модель жизненного цикла (ЖЦ), которую команда будет или должна использовать при разработке. Такой подход сразу упрощает способ применения стандарта OMG ESSENCE.

Для каждой модели ЖЦ можно выделить этапы (итерации) разработки, а для каждой итерации определить состав сущностей, которые на этой итерации должны приобрести определенные ожидаемые целевые значения/состояния. Эти значения предполагаются «правильными», а отклонения от них — признаки нарушения «здоровья» проекта.

1. Цель и задачи работы

Целью работы, результаты которой представлены в статье, является разработка на основе стандарта OMG ESSENCE метода и инструментальных средств для автоматизированной оценки качества (прогресса и «здоровья» проекта), позволяющего на каждой итерации/этапе проекта, основываясь на состояниях сущностей, автоматизировано оценивать отклонения текущих результатов от ожидаемых («правильных») и вырабатывать предложения по правильному развитию проекта.

Задачами исследования являются:

- разработка математического аппарата, позволяющего для различных моделей ЖЦ разработки программного обеспечения (ПО) по заполненным чек-листам автоматически получить оценку качества выполнения каждого этапа проекта по состоянию сущностей, а в случае отклонения сущности от правильных значений выявить ту (те) сущность (и), которые влияют на эти значения;
- разработка сервиса, который позволяет автоматически выработать рекомендации по улучшению качества проекта на последующих этапах для команды проекта любой квалификации.

Объект исследования. Объектом исследования является состояние выполнения программного проекта, разрабатываемого по некоторой модели ЖЦ, сводящейся к итерационной.

В качестве объекта исследования в данной работе выбраны программные проекты, которые разрабатывались в рамках проектной деятельности магистрантов МИЭМ НИУ ВШЭ. Проекты разрабатывались группами численностью до пяти участников, роли которых определялись перед началом проекта. В качестве заинтересованной стороны в проекте выступал руководитель проекта. При выполнении проекта использовалась итерационная модель ЖЦ. Особенностью такой модели для студенческих проектов являлось то обстоятельство, что в ней определены ожидаемые результаты проектной работы как содержательно (что должно быть реализовано), так и в виде модели ЖЦ.

Предмет исследования. Предметом исследования является возможность автоматической оценки состояния проекта на каждом его этапе (итерации), как интегрального состояния всех сущностей, а также автоматическое сопоставление этой оценки с эталонным состоянием, в результате которого выявляется причина отклонения.

2. Итерационная модель жизненного цикла

Ядро ESSENCE описывает состав сущностей любого программного проекта и правильные отношения между ними в процессе реализации проекта (Заинтересованные стороны, Возможность, Требования, Программная система, Работа, Команда, Технология работы). Для каждого проекта в стандарте OMG ESSENCE приведены правильные состояния сущностей; каждая сущность может принимать состояния, которые перечислены. Описаны и промежуточные состояния сущностей, по которым можно оценить, достигнуто правильное состояние сущности полностью или

частично. Для каждой сущности совокупность ее промежуточных состояний образует чек-лист. Пример модели ЖЦ приведен на рис. 1.

При итерационной модели ЖЦ для каждой итерации можно сформулировать задачу, которая решается на этой итерации и, соответственно, развитие каких сущностей в проекте должно приводить к решению задачи на этом этапе. Таким образом, определяются сущности, которые являются результатом этой итерации, а также их состояния на рассматриваемой итерации при правильном развитии проекта. При таком подходе для каждой итерации в модели ЖЦ можно определить множество сущностей, являющихся правильным состоянием результатов итерации, и подмножество правильных (далее — нормальных) значений сущностей для конкретной итерации.

Однако проекты могут развиваться не совсем так, как предписывает модель ЖЦ. При этом либо

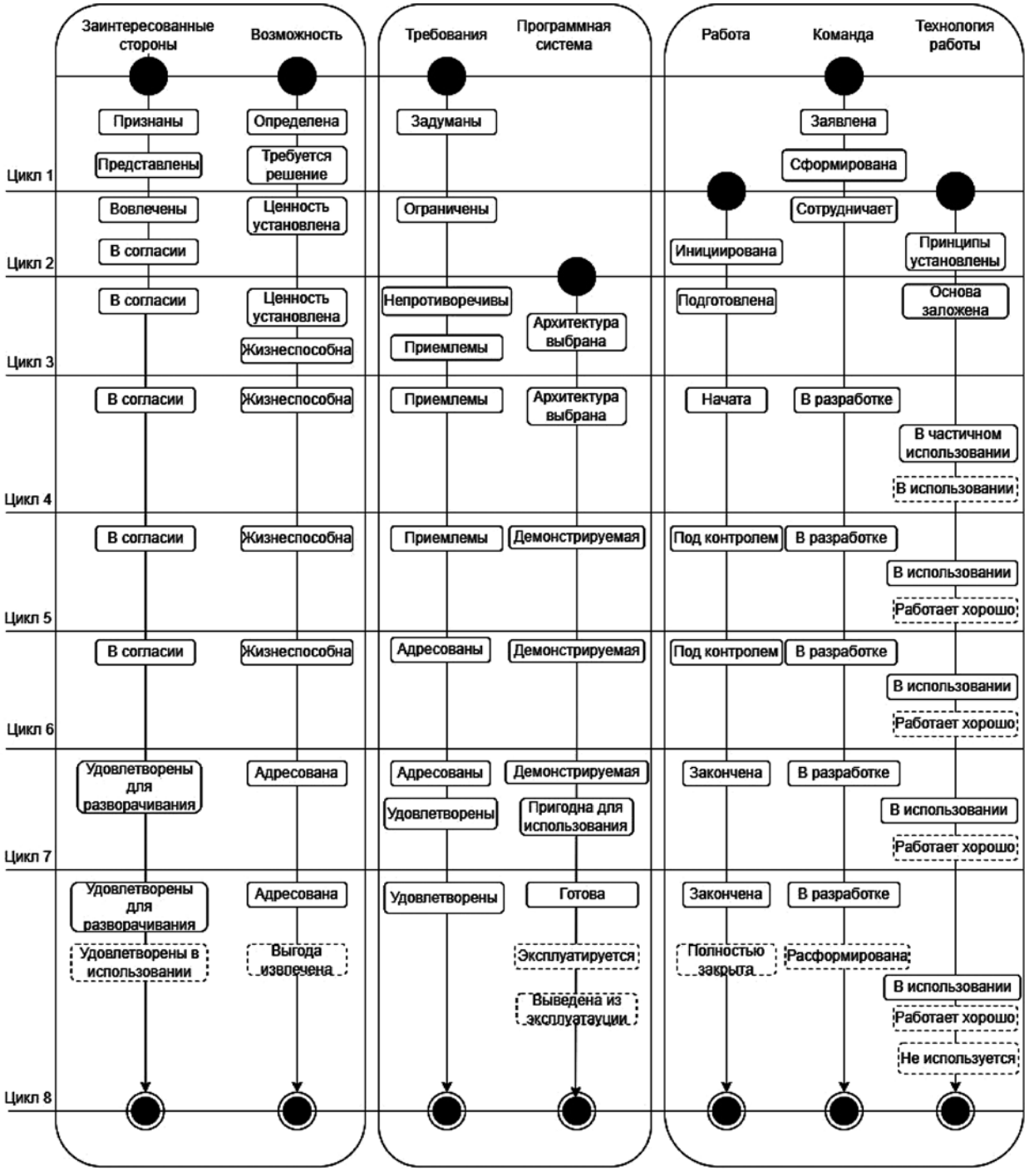


Рис. 1. Модель ЖЦ итерационного проекта (на рисунке итерации названы циклами)

сами изменившиеся сущности могут оказаться не в том состоянии, которые предусмотрены в модели, либо правильные состояния сущностей, описанные в чек-листах, могут по объективным причинам не достигаться. Такие отклонения свидетельствуют о нарушениях «здоровья» проекта на рассматриваемой итерации. Эти формализованно описанные нарушения показывают, что именно не так произошло в проекте, какие состояния сущностей не достигнуты. Описание достигнутых состояний является исходными данными руководителя и всей команды проекта для указания на то, какие недостатки следует исправить на последующих итерациях. Заметим, что проект может быть завершен с нарушениями «здоровья», что, как правило, свидетельствует о недостаточно высоком качестве разработки.

3. Метод оценки прогресса и «здоровья» проекта

Согласно стандарту, все программные проекты описываются с помощью семи сущностей α_m, $m = 1...7$. Каждая сущность может принимать $\{\pi_{mr}\}$ состояний, где r — это номер состояния для каждой сущности, все состояния перечислены в стандарте в явном виде. В ходе выполнения проекта состояния сущностей изменяются в зависимости от того, каких результатов достигает команда проекта. В стандарте определены признаки, с помощью которых оценивается, достигнуто или не достигнуто то или иное состояние конкретной сущности π_{mr}. Совокупность всех $\{\pi_{mr}\}$ для всех сущностей сведена в чек-листы, которые заполняет руководитель команды проекта при завершении каждой итерации проекта. Каждая итерация i характеризуется определенным набором сущностей (рис. 1), которые значимы для достижения целей именно этой итерации и должны принять правильные состояния α_{mi}.

Руководитель команды проекта при завершении каждой итерации (цикла) заполняет чек-листы в части достигнутых состояний значимых сущностей.

Пример. На рис. 1 показано, что на первой итерации (цикле) значимыми являются четыре сущности из семи:

Сущность Заинтересованные стороны (α_1), принимает два состояния: π_{11} — *Признаны* и π_{12} — *Представлены*;

Сущность Возможность (α_2) принимает два состояния: π_{21} — *Определена* и π_{22} — *Требуется решение*;

Сущность Требования (α_3) принимает состояние: π_{31} — *Задуманы*;

Сущность Команда (α_6) принимает два состояния: π_{61} — *Заявлена* и π_{62} — *Сформирована*.

Правильные состояния сущностей приведены на рис. 1 и отражены в чек-листе. По завершении цикла 1 руководитель проекта заполняет поле чек-листа, в котором указаны достигнутые на цикле 1 состояния сущностей. Например, если команда проекта заявлена, но не сформирована после первой итерации, то значение π_{62} будет соответствовать тому, что команда НЕ сформирована.

Для автоматизации получения оценки прогресса и «здоровья» проектных работ предложена математическая модель, построенная на основе предикатов первого порядка для всех итераций (циклов).

Результаты выполнения каждого цикла описываются предикатом первого порядка. По модели ЖЦ можно определить, какие сущности значимы на данном цикле и должны достигнуть нормального состояния. Множество значимых сущностей для цикла i обозначим $\{\alpha_i\}$. Состояния сущности r для цикла i обозначим как $<\pi_{ir}>$.

Значения предиката первого порядка $\varphi(\{\alpha_i < \pi_{ir} >\})$ для цикла i определяются как результат проверки соответствия достигнутых значений сущности, значимых для цикла i, правильным значениям из чек-листов:

$$\varphi\left(\left\{\alpha_i \left\langle \pi_{ir} \right\rangle\right\}\right) = \begin{cases} 1, \text{ если } \cap\{\alpha_i \left\langle \pi_{ir} \right\rangle\} = 1 \text{ и} \\ 0 \text{ в противном случае.} \end{cases}$$

Иными словами, если сущность на данном этапе достигла необходимого (нормального) состояния, то она принимает значение 1, в противном случае, т. е. если хотя бы одна сущность не соответствует нормальному состоянию, то ее значение равно 0.

Заметим, что при начальном формировании чек-листов для нормальных состояний значимых сущностей сразу устанавливаются значения «все 1».

Таким образом, описание результатов каждого цикла можно проводить с помощью предикатов первого порядка, так же как и итоговое описание результатов проекта по всем циклам.

Рассмотрим на примере. На первом цикле сущность «заинтересованные стороны» должна принимать значение «признаны», для этого должен выполняться список условий [14]:

- выявлены все различные группы заинтересованных сторон, которые затронуты или будут затронуты разработкой и эксплуатацией системы ПО;
- существует соглашение о группах заинтересованных сторон, которые будут представлены;
- рассмотрены, как минимум, заинтересованные группы, которые финансируют, используют, поддерживают систему;

• определены обязанности представителей заинтересованных сторон.

Состояние «признаны» будет достигнуто (примет значение 1) только в том случае, если чек-лист будет иметь значения (1, 1, 1, 1), т. е. все четыре условия из списка выполнены, в противном случае 0.

Для состояния «признаны» функция будет выглядеть следующим образом:

$$\varphi_{\text{признаны}} = \begin{cases} 1, & \text{если } P(1, 1, 1, 1), \\ 0 & \text{в противном случае.} \end{cases}$$

Аналогичным образом можно описать состояния каждой сущности на каждом цикле (для модели ЖЦ на рис. 1 таких циклов 8).

Таким образом, становится понятно, что легко выявить причины того, почему состояние аномальное, а также выявить сущности, на которые следует воздействовать. Действительно, если значение предиката равно 0, то выбираем для соответствующей сущности те состояния, которые приняли значение 0. Для них вероятнее всего правильные состояния не достигнуты. Потенциально состояния 0 указывают на то, какие сущности и какие их состояния являются причиной аномального результата.

Среди всех сущностей и достигнутых значений кандидатов на неправильные состояния выявляется, связывают ли сущности-кандидаты отношения (уровня ядра). Если да, то неправильное состояние может быть не вследствие самого состояния, в котором обнаружена аномалия, а в силу того состояния, которое является участником отношения с рассматриваемым состоянием.

4. Граф взаимосвязей, построенный по модели отношений ядра, и алгоритм исследования прогресса и «здоровья» проекта

В ядре ESSENCE сущности связаны отношениями. Совокупность этих связей и вершин (сущностей) представляет собой ориентированный граф, который можно рассматривать как модель правильных отношений между сущностями. Эта модель может быть аннотирована отношениями. Назовем этот граф графом взаимосвязей.

Этот граф состоит их двух компонент связности (рис. 2), первая соответствует левой ветви Vee-модели (разработка ПО), а вторая — правой ветви Vee-модели (проверка). С учетом того, что обе компоненты являются ациклическими графами, сам граф может рассматриваться как эталон для проверки зависимости неправильных состояний, рассмотренных в предыдущем разделе.

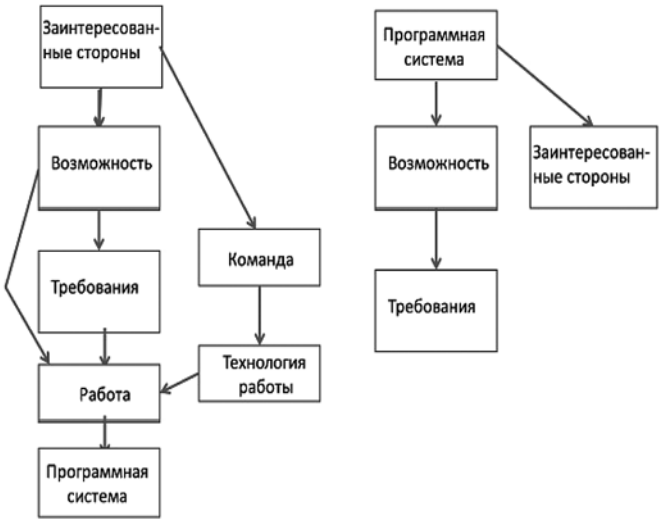


Рис. 2. Граф взаимосвязей

Таким образом, алгоритм исследования прогресса и «здоровья» проекта можно представить приведенным далее описанием.

Подготовительная фаза (выполняется однократно для модели ЖЦ проекта):

— выявление нормального состояния для каждой сущности и совокупности всех значимых на этом цикле сущностей;

— формирование эталонов для оценки текущего состояния (составление чек-листов правильных состояний).

Основная фаза (для каждой итерации):

— по заполненным чек-листам на основании предикатной модели определение сущностей, которые не достигли нормального состояния (определяется «здоровье» проекта);

— определение сущностей, которые участвовали в получении данного состояния по графу взаимосвязей, и определение возможных причин аномального состояния;

— по лепестковой диаграмме определение дисбаланса между нормальным и полученным состояниями (определяется прогресс проекта);

— формирование рекомендаций по улучшению состояния проекта.

Общая схема методики оценки состояния работ, основанная на описанном выше алгоритме, представлена в табл. 1.

Такой подход позволяет автоматизировать процесс исследования прогресса и «здоровья» проекта, а также дать человеко-машинную оценку качества состояния проекта на каждом этапе его развития, а следовательно, реального прогресса проекта. В случае, когда на графе взаимосвязей отношения между сущностями не определены, доступным является построение подграфов, свя-

Таблица 1

Общая схема методики оценки состояния работ

Шаги методики	Способ решения задачи	Действия для решения задачи
Оценка состояния работ с помощью математической модели для каждого цикла	Получение формальной оценки результата на текущем цикле	○ Получение оценки по каждой сущности, актуальной для анализируемого текущего цикла
Построение лепестковой диаграммы для каждого цикла с определением нормального состояния и текущего состояния с учетом оценки на математической модели	Визуализация оценки текущего состояния по отношению к нормальному для каждого цикла	○ Построение лепестковой диаграммы для текущего цикла с определением нормального состояния по модели ЖЦ и текущего состояния по результатам оценки цикла
Определение возможных причин отклонения текущего состояния от планового по графу взаимосвязей	Определение сущности, которая повлияла на аномальный результат на каждом цикле	○ Выявление по графу взаимосвязей сущностей, влияющих на те сущности, состояния которых признаны аномальными
Формирование оценки состояния и выработка рекомендаций по исправлению ситуации	Формирование объективной оценки состояния и рекомендаций по исправлению ситуации	○ Формирование заключения по текущему состоянию цикла, в котором описываются аномалии и их причины. ○ Формулировка рекомендаций по исправлению ситуации
Обсуждение с командой и руководителями проекта корректности заполнения чек-листов с возможностью демонстрации последствий неправильного заполнения	Обучение руководителя и команды проекта на материалах проекта, систематизация ошибок	○ Оперативная демонстрация причин и последствий неправильного заполнения чек-листов

зывающих сущность с аномальным результатом, и сущности, из которых она потенциально достижима. Этот подграф перечисляет возможные сущности, которые могут быть причиной аномалии. Если же отношения ядра перенесены на граф взаимосвязей, то сущности, являющиеся причиной аномалии, могут быть вычислены.

Важно отметить, что такой подход применим к любой итерационной модели ЖЦ проекта [12]. Нужно, чтобы для каждой итерации ЖЦ был установлен состав сущностей и их значимых состояний по результатам каждой итерации. Чек-листы в любом случае определены в стандарте ESSENCE.

5. Тестирование математической модели

Для проверки разработанной математической модели проведены эксперименты на нескольких проектах, которые проходили в рамках проектной деятельности студентов в МИЭМ НИУ ВШЭ. Проверка проводилась на нескольких программных проектах в 2020/2021 и в 2021/2022 учебных годах. В совокупности апробация методики проводилась на десяти проектах, результаты фиксировались поэтапно в продуктах Trello и Wekan. По данным из Wekan удалось провести подробный анализ с использованием математической модели и лепестковых диаграмм.

По итогам каждого цикла руководители проектных команд заполняли чек-листы. Полученные таким образом данные проанализированы, по заполненным чек-листам построены лепестковые диаграммы, в которых отражается нормальное состояние проекта, а также состояние, которое принимают сущности в исследуемом проекте. Причина отсутствия достижения нормального состояния на исследуемом цикле определялась с помощью графа взаимосвязей между сущностями.

6. Построение лепестковых диаграмм для реальных проектов

При исследовании состояния проекта в любой точке, в которой возможно получить оценки значений сущностей, удобно применять для анализа ситуаций изображение состояния некоторого количества показателей одновременно в нескольких осях, назовем такое изображение лепестковой диаграммой (иногда в литературе используют название «паутинка») [8]. В случае оценок проекта по состояниям сущностей (и с учетом ядра ESSENCE [14]) лепестковая диаграмма будет иметь семь осей (сущностей), на каждой из осей располагаются значения показателей — по числу нормальных состояний сущностей. Для любой модели ЖЦ (например, для ЖЦ на рис. 1) для каждого

цикла (итерации) проекта можно сформировать нормальные состояния каждой из сущностей, являющихся результатом этого цикла.

Из модели ЖЦ формируется нормальное состояние каждой сущности, являющейся результатом цикла. По чек-листам определяются значения признаков правильных состояний. Эта работа, как отмечено выше, проводится для каждой модели ЖЦ, по которой реализуется проект. Правильные значения признаков нормальных состояний сущностей для каждого цикла образуют на лепестковой диаграмме (рис. 3, см. вторую и третью стороны обложки) замкнутый контур, который отображает нормальное состояние проекта. При выполнении проекта такая лепестковая диаграмма является эталоном нормального состояния проекта на каждом цикле. Нормальные состояния циклов проекта на графике рис. 3 (см. вторую и третью стороны обложки) показаны красным цветом. Отклонение результатов каждого цикла от нормального состояния приводит к тому, что на лепестковой диаграмме появляются точки, показанные синим цветом. Соединение этих точек образует «синий контур», сопоставление которого с красным контуром показывает, в какой части состояние проекта отклоняется от нормального состояния, а также позволяет оценить причины отклонения проекта от нормы.

В качестве примера приведем серию лепестковых диаграмм для всех восьми циклов одного из проектов (проект 915). Диаграммы получены двумя способами:

1) (слева на рис. 3, см. вторую и третью стороны обложки) в результате ручного заполнения чек-листов и ручной обработки результатов применения математической модели для каждого цикла и использования Excel как средства построения лепестковых диаграмм;

2) (справа на рис. 3, см. вторую и третью стороны обложки) с использованием разработанного сервиса для проведения таких оценок (см. разд. 8); в этом случае чек-листы заполняются руководителем проекта с применением сервиса, а лепестковые диаграммы рассчитывает и отображает сервис.

Анализ данных рис. 3 (см. вторую и третью стороны обложки) показывает, что, начиная с цикла 1, проекты не полностью соответствуют нормальным значениям. По диаграммам можно увидеть, что на начальных циклах не полностью сформированы команды заинтересованных сторон и разработки, не освоена технология работ. Однако по мере совершенствования связей с заинтересованными сторонами (циклы 2, 3) формируются достаточно качественные требования (циклы 4, 5), к циклу 6 начинается систематическая работа по разработке программной системы, нормальные значения сущностей достигаются. Остаются только вопросы, связанные с тем, что программная система разработана как действующий прототип будущей системы, как ее макет. Это связано со студенческим статусом проектов, вследствие чего результат разработки далеко не всегда может быть передан в постоянную эксплуатацию. С этим связаны отклонения синего контура продукта проекта от нормальных значений (красного контура).

7. Экспериментальное исследование

Наиболее «слабым» элементом описываемого человеко-машинного процесса является работа руководителя проекта по заполнению чек-листов и степень достоверности оценок «здоровья» проекта, получаемых на основании автоматизированной обработки собранных данных. Применим для получения таких оценок принятые в математической статистике оценки ошибок первого и второго рода.

Под ошибкой первого рода понимается ложноположительное заключение, а именно оценка вероятности того, что любое состояние «здоровья» проекта будет идентифицировано как «нездоровье». Вероятность допустить такую ошибку называется уровнем значимости. Под ошибкой второго рода понимается ложно отрицательное заключение, т. е. оценка вероятности того, что состояние «нездоровья» проекта идентифицируется как «здоровье». Применительно к выборке по результатам оценок проектов описанным методом это означает следующее.

Основу исходных данных для получения оценок составляют нормальные и действительные значения каждой сущности по каждому из восьми циклов. Для рассмотренных в рамках экспериментальных работ четырех студенческих проектов получаем оценки, отображенные в табл. 2

Таблица 2

Результаты выполнения четырех проектов

Характеристика проекта	Ложно-положительная оценка	Ложно-отрицательная оценка
Наличие «нездоровья»	24	1
«Здоровый» проект	4	3
Всего	28	4
Уровень значимости	0,86	—
Вероятность ошибки второго рода	—	0,75

Из представленных данных следует, что алгоритм из 4 «здоровых» проектов признает «нездоровым» только 1, из 28 «нездоровых» циклов алгоритм признает «здоровыми» 4. Такие показатели указывают на то, что метод оценки обладает довольно высокими значениями уровня значимости и вероятности ошибки второго рода даже для сравнительно небольшой статистики приведенной выборки. Если анализируется состояние большего количества проектов, то статистическая точность результата должна увеличиваться. Это означает, что при длительном применении методики и оценок в разрабатывающей организации уровень значимости и вероятность ошибки второго рода могут использоваться для оценки прогресса и «здоровья» проекта как достаточно надежная оценка, а качество самой методики будет только возрастать по мере ее использования.

8. Принципиальная схема сервиса для автоматической выработки рекомендаций по улучшению прогресса и «здоровья» проекта

По результатам проведенных теоретических и экспериментальных работ разработан сервис для автоматической выработки рекомендаций по улучшению прогресса и «здоровья» проекта. Принципиальная схема сервиса приведена на рис. 4. Сервис работает в двух режимах: в режиме настройки на тип проекта и в режиме оценки текущего проекта.

В режиме настройки на тип проекта в сервис вводится модель ЖЦ программного проекта (например, модель на рис. 1), которая принята командой проекта. Чек-листы сущностей берут или из стандарта OMG ESSENCE или для более сложных моделей ЖЦ, например, с субальфами [9], дорабатывают чек-листы в соответствии с принятой в проекте моделью ЖЦ.

В режиме оценки сервис используют на каждом цикле выполнения проекта для заполнения чек-листов, после которого сервис выставляет соответствующие оценки, строит лепестковую диаграмму и предлагает рекомендации по развитию проекта.

Таким образом, сервис осуществляет автоматизированную оценку прогресса и «здоровья» проекта в соответствии с методикой, описанной в разд. 4.

При использовании компанией-разработчиком определенной модели ЖЦ разными (или всеми) командами разработчиков настройка на тип проекта осуществляется единожды. Последующие оценки хода выполнения проектов по этой модели ЖЦ позволяют не только оценивать прогресс и «здоровье» проектов, но и вырабатывать на основе собранной статистики правила и нормативы работы команд разработчиков.

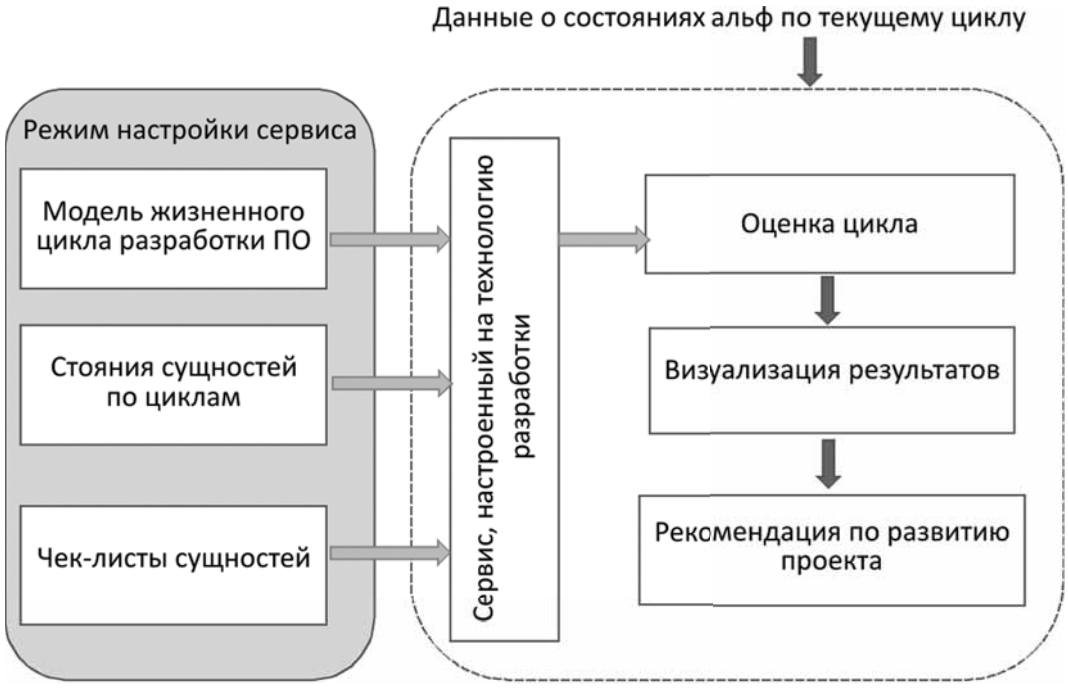


Рис. 4. Принципиальная схема сервиса

Заключение

Результаты, полученные в данной работе, позволяют сделать следующие выводы.

Разработан метод, основанный на предикатах первого порядка, позволяющий для различных итерационных моделей жизненного цикла разработки ПО по заполненным чек-листам автоматически получить оценку прогресса и «здоровья» проекта каждой итерации (этапа) проекта по состоянию сущностей.

В случае отклонения сущности от нормальных (правильных) значений предложено выявить ту (те) сущность(и), которые влияют на эти значения, выявление проводится по графу взаимосвязей сущностей, построенному по ядру ESSENCE.

Разработан алгоритм исследования и основанная на нем методика автоматизированной оценки состояния работ по проекту.

Показано, что дополнительным методическим материалом для оценки прогресса и «здоровья» проекта могут успешно использоваться лепестковые диаграммы, полученные в результате оценок и отражающие после каждой итерации ЖЦ проекта степень достижения сущностями нормальных (правильных) значений и демонстрирующими то, как отражают отклонения в значениях сущностей нарушения технологии ведения работ.

Проведено экспериментальное исследование поитерационных результатов оценки прогресса и «здоровья» проекта, получены предварительные оценки уровня значимости и вероятности ошибки второго рода в рамках предложенной методики автоматизированной оценки состояния работ по проекту.

Разработан сервис, который позволяет автоматически поитерационно оценивать прогресс и «здоровье» проекта и вырабатывать рекомендации по улучшению качества работ на последующих итерациях для команды проекта любой квалификации.

Список литературы

1. **Kruchten P.** The frog and the octopus: a conceptual model of software development, 2007. URL: https://arxiv.org/abs/1209.1327. (дата обращения 19.06.2023).
2. **Humphrey W. S.** Discipline for Software Engineering Addison-Wesley, 1995. 789 p.
3. **Boehm B., Jain A.** An initial theory of value-based software engineering // Value-based Software Engineering. Springer. 2005. P. 15—37.
4. **Capers Jones.** Software quality in 2010: A survey of the state of the art. Capers Jones & Associates, 2010. URL: https://sqgne.org/presentations/2010-11/Jones-Nov-2010.pdf (дата обращения 19.06.2023).
5. **Bjorner D.** The Triptych process model — process assessment and improvement. Tokio University, 2006. DOI: 10.1081/E-ESE-120044141.
6. **Capers Jones**. Software Engineering Best Practices Lessons from Successful Projects in the Top Companies, McGraw-Hill, 2010. 643 p.
7. **Змеев Д. О.** Прототип системы поддержки принятия решений для управления проектами на основе стандарта OMG ESSENCE и байесовских сетей: дисс. ... физ.-мат наук. Томск, 2022.
8. **Лепестковая** диаграмма — направления и числа. Информатика, Электронные таблицы. Фоксфорд Учебник. URL: https://foxford.ru (дата обращения: 09.05.2023).
9. **OMG** Essence: официальный сайт. URL: http://sewiki.ru/OMG_Essence (дата обращения 09.05.2023).
10. **Essence** — Kernel and Language for Software Engineering Methods: официальный сайт. URL: https://www.omg.org/spec/Essence/1.2/PDF (дата обращения 09.05.2023).
11. **ESSENCE** User Guide. SEMAT. URL: https://semat.org/view-1-essence-lite (дата обращения 09.05.2023).
12. **Semat:** официальный сайт. URL: https://www.semat.org/ (дата обращения 05.02.2022).
13. **Jacobson I., Ng P.-W., McMahon P. E., Spence I., Lidman S.** The Essence of Software Engineering Applying the SEMAT Kernel. Addison-Wesley, 2013. 352 p.
14. **Jacobson I., Lawson H., Ng P.-W., McMahon P. E., Goedicke M.** The Essentials of Modern Software Engineering, 2019. URL: https://morganclaypoolpublishers.com/essence/FULL_TOC.pdf

Automated Assessment of the Progress and "Health" of the Project Based on the OMG ESSENCE Standard

B. A. Pozin [1,2,3], Doctor of Technical Sciences, Professor, bpozin@ec-leasing.ru,
A. E. Garanina[1], Master, aegaranina@edu.hse.ru,
E. A. Ivanova[1], Master, eaivanova_23@edu.hse.ru
[1] Higher School of Economics Research University, Moscow, 109028, Russian Federation,
[2] Ivannikov Institute for System Programming of the Russian Academy of Sciences, Moscow, 109004, Russian Federation,
[3] JSC EC-leasing, Moscow, 117587, Russian Federation,

Corresponding author:

Boris A. Pozin Doctor of Technical Sciences, Professor, Higher School of Economics Research University, Moscow, 109028, Russian Federation, Ivannikov Institute for System Programming of the Russian Academy of Sciences, Moscow, 109004, Russian Federation, JSC EC-leasing, Moscow, 117587, Russian Federation
E-mail: bpozin@ec-leasing.ru

Received on 15.05.2023
Accepted on 20.06.2023

The automated technology developed by the authors for assessing the state of progress and "health" in the framework of a software project based on the OMG ESSENCE standard is presented. Such an assessment is based on the degree to which the normal values of certain seven entities are achieved, which are the basis for the implementation of any software project. The values of entities are defined in the ESSENCE standard. A method for assessing the state of progress and "health" of the project is proposed, which allows automating the receipt of such assessments at each iteration of its implementation based on the analysis of checklists filled out by the project manager. In case of deviation of the results from normal values, a method for identifying dependencies is proposed based on the analysis of the graph of relationships built on the ESSENCE core. A service to automate the assessment is developed, technology implementation statistics are provided, estimates of errors of the first and second kind are obtained.

Keywords: *ESSENCE, quality of the software project, project progress, "health" of the project, graph of relationships, predicate model of evaluation of iteration results, automation of evaluation of the quality of the software project, SEMAT*

For citation:

Pozin B. A., Garanina A. E., Ivanova E. A. Automated Assessment of the Progress and "Health" of the Project Based on the OMG ESSENCE Standard, *Programmnaya Ingeneria,* 2023, vol. 14, no. 8, pp. 367—376. DOI: 10.17587/prin.14.367-376.

References

1. **Kruchten P.** The frog and the octopus: a conceptual model of software development, 2007, available at: https://arxiv.org/abs/1209.1327 (date of access 19.06.2023).

2. **Humphrey W. S.** *Discipline for Software Engineering,* Addison-Wesley, 1995, 789 p.

3. **Boehm B., Jain A.** An initial theory of value-based software engineering, *Value-based Software Engineering*, Springer, 2005, pp. 15—32.

4. **Capers Jones.** Software quality in 2010: A survey of the state of the art. Capers Jones & Associates, 2010, available at: https://sqgne.org/presentations/2010-11/Jones-Nov-2010.pdf (date of access 19.06.2023).

5. **Bjorner D.** The Triptych process model — process assessment and improvement. Tokio University, 2006. DOI: 10.1081/E-ESE-120044141.

6. **Capers Jones**. *Software Engineering Best Practices Lessons from Successful Projects in the Top Companie*s, McGraw-Hill, 2010, 643 p.

7. **Zmeyev D. O.** Prototype of a decision support system for project management based on the OMG ESSENCE standard and Bayesian networks: dissertation. Tomsk, 2022 (in Russian).

8. **Petal** Diagram — Directions and Numbers, Computer Science, Spreadsheets Foxford Textbook, available at: foxford.ru (date of access 09.05.2023).

9. **OMG** Essence: official site, available at: http://sewiki.ru/OMG_Essence (date of access 09.05.2023).

10. **Essence —** Kernel and Language for Software Engineering Methods, available at: https://www.omg.org/spec/Essence/1.2/PDF (date of access 09.05.2023).

11. **ESSENCE** User Guide, SEMAT, available at: https://semat.org/view-1-essence-lite (date of access 09.05.2023).

12. **SEMAT:** official site, available at: https://www.semat.org/ (date of access 09.05.2023).

13. **Jacobson I., Ng P.-W., McMahon P. E., Spence I., Lidman S.** *The Essence of Software Engineering Applying the SEMAT Kernel,* Addison-Wesley, 2013, 352 p.

14. **Jacobson I., Lawson H., Ng P.-W., McMahon P. E., Goedicke M.** The Essentials of Modern Software Engineering, 2019, available at: https://morganclaypoolpublishers.com/essence/FULL_TOC.pdf (date of access 09.05.2023).

УДК 004.315
DOI: 10.17587/prin.14.377-387

П. Н. Бибило, д-р техн. наук, проф., зав. лаб., bibilo@newman.bas-net.by,
Объединенный институт проблем информатики Национальной академии наук Беларуси, г. Минск

Синтез схем модулярных умножителей

Поступила в редакцию 22.05.2023
Принята к публикации 08.06.2023

*Приведены результаты экспериментов по схемной реализации модулярных умножителей в библиотеке проектирования заказных СБИС (сверхбольших интегральных схем) и FPGA (Field-Programmable Gate Array — программируемая пользователем вентильная матрица). Исходные описания проектов модулярных умножителей задавались системами не полностью определенных (частичных) булевых функций и алгоритмическими VHDL-описаниями. Логическая оптимизация проводилась в классе дизъюнктивных нормальных форм (ДНФ) и представлений систем булевых функций бинарными диаграммами решений. Синтезированные схемы оценивались по площади и временной задержке. Было установлено, что использование моделей частичных булевых функций и предварительной логической BDD-оптимизации (Binary Decision Diagrams — бинарная диаграмма решений) позволяет улучшать параметры синтезируемых блоков заказных СБИС и FPGA для небольших значений модуля, однако лучшие решения для больших значений модуля можно получить, используя алгоритмические VHDL-описания модулярных умножителей. При синтезе схем модулярных умножителей в составе FPGA и применении систем проектирования ISE и Vivado (производства Xilinx) целесообразно использовать синтезируемые VHDL-операции (a*b) mod p.*

Ключевые слова: *модулярные вычисления, модулярный умножитель, операция деления целых чисел, булева функция, дизъюнктивная нормальная форма, бинарная диаграмма решений, синтез логической схемы, VHDL, заказная СБИС, FPGA*

Для цитирования:

Бибило П. Н. Синтез схем модулярных умножителей // Программная инженерия. 2023. Том 14, № 8. С. 377—387. DOI: 10.17587/prin.14.377-387.

Введение

Создание эффективных параллельных вычислительных систем часто опирается на модулярные вычисления [1, 2] и соответствующие им аппаратные реализации [3, 4]. Для небольших значений модуля (разрядность операндов не более 7, 8) в работах [5—7] предлагается использовать системы булевых функций для исходного функционального описания модулярных умножителей, а для уменьшения аппаратной сложности — минимизацию в классе ДНФ систем булевых полностью определенных функций. Экспериментальные исследования схемных реализаций модулярных умножителей для заказных СБИС (сверхбольших интегральных схем) были проведены в работе [7], где описаны различные способы алгоритмического описания умножителей и приведены результаты вычислительных экспериментов. Синтез схем не только модулярных умножителей, но и различных функциональных комбинационных блоков заказных цифровых СБИС в заданном базисе (библиотеке) логических элементов традиционно разбивается на два больших этапа: технологически независимую оптимизацию реализуемых систем булевых функций и технологическое отображение — собственно синтез. Под технологическим отображением понимается выделение в оптимизированных описаниях фрагментов, соответствующих функциям элементов технологической библиотеки (базиса синтеза), и покрытие описаний такими фрагментами. Решающее влияние на основные параметры (сложность, быстродействие, энергопотребление) логических схем оказывает первый этап.

В данной работе в отличие от работы [7] предлагается описывать функции модулярных умножителей системами *частичных* (не полностью определенных) булевых функций и осуществлять логическую оптимизацию, выполняя

совместную минимизацию в классах ДНФ [8, 9] и BDD-представлений (*Binary Decision Diagrams* — бинарная диаграмма решений) [10]. Экспериментально установлено, что использование моделей частичных функций и предварительной логической BDD-оптимизации позволяет значительно улучшать параметры синтезируемых логических схем по сравнению с использованием функциональных описаний умножителей, соответствующих моделям полностью определенных булевых функций. Полученные решения для заказных КМОП СБИС сравнивались со схемными реализациями, получаемыми в синтезаторе LeonardoSpectrum [11], при задании входных данных в виде алгоритмических VHDL-описаний (VHDL — *Very high speed integrated circuits Hardware Description Language* — язык описания аппаратуры сверхскоростных интегральных схем). Схемная реализация VHDL-описаний модулярных умножителей на микросхемах FPGA семейства Spartan-6 [12] осуществлялась в системе автоматизированного проектирования ISE (*Integrated System Environment*) Design Suite компании Xilinx. В системе Vivado выполнялись эксперименты по реализации модулярных умножителей для FPGA семейства Kintex-7 [13] (компания Xilinx). Заметим, что в ISE и Vivado реализованы программные средства схемной реализации VHDL-операции *mod* вычисления по модулю ($y <= a \bmod b$). В системе LeonardoSpectrum (версия 2010a.7) данная операция является несинтезируемой, поэтому в экспериментах в этой системе операция *mod* заменена синтезируемой VHDL-функцией операции деления целых неотрицательных чисел, представленных в двоичном коде.

1. Пример модулярного умножителя

Умножение по модулю p (основание модулярной вычислительной системы) для двух неотрицательных чисел (операндов) a, b, находящихся в диапазоне $\{0, 1, ..., p-1\}$, выполняется согласно формуле

$$|a \times b|_p = (a \times b) - \left\lfloor \frac{a \times b}{p} \right\rfloor \times p,$$

где через $\lfloor k \rfloor$ обозначена целая часть числа, т. е. ближайшее целое, меньшее либо равное k. В случае если $(a \times b) < p$, то $|a \times b|_p = a \times b$.

Число M двоичных разрядов, представляющих числа $\{0, 1,..., p-1\}$, определяется по формуле $M = \lceil \log_2 p \rceil$, где через $\lceil A \rceil$ обозначается ближайшее целое, большее либо равное A. В табл. 1 представлена система частичных булевых функций $y_1(a_1, a_2, a_3, b_1, b_2, b_3)$, $y_2(a_1, a_2, a_3, b_1, b_2, b_3)$, $y_3(a_1, a_2, a_3, b_1, b_2, b_3)$, задающая умножитель по модулю 5 ($p = 5$); числа a, b принимают значения из диапазона {0, 1, 2, 3, 4}. Неопределенные значения булевых функций y_1, y_2, y_3 обозначены символом «–».

Таблица 1

Умножитель по модулю 5 — система частичных булевых функций

Область определенных значений		Область неопределенных значений	
$a_1\ a_2\ a_3\ b_1\ b_2\ b_3$	$y_1\ y_2\ y_3$	$a_1\ a_2\ a_3\ b_1\ b_2\ b_3$	$y_1\ y_2\ y_3$
000000	000	000101	– – –
000001	000	000110	– – –
000010	000	000111	– – –
000011	000	001101	– – –
000100	000	001110	– – –
001000	000	001111	– – –
001001	001	010101	– – –
001010	010	010110	– – –
001011	011	010111	– – –
001100	100	011101	– – –
010000	000	011110	– – –
010001	010	011111	– – –
010010	100	100101	– – –
010011	001	100110	– – –
010100	011	100111	– – –
011000	000	101000	– – –
011001	011	101001	– – –
011010	001	101010	– – –
011011	100	101011	– – –
011100	010	101100	– – –
100000	000	101101	– – –
100001	100	101110	– – –
100010	011	101111	– – –
100011	010	110000	– – –
100100	001	110001	– – –
		110010	– – –
		110011	– – –
		110100	– – –
		110101	– – –
		110110	– – –
		110111	– – –
		111000	– – –
		111001	– – –
		111010	– – –
		111011	– – –
		111100	– – –
		111101	– – –
		111110	– – –
		111111	– – –

2. Алгоритмы и программы логической минимизации

Перед выполнением некоторых процедур технологически независимой логической оптимизации проводилось доопределение (замена) неопределенных значений «–» в таблицах истинности частичных булевых функций нулевыми значениями. Далее такое доопределение будет называться «*нулевым*».

2.1. Минимизация в классе ДНФ

Совместная минимизация систем булевых функций, реализуемых модулярными сумматорами, выполнялась программой Espresso IIC [8] и программой Minim [9], входящей в систему логической оптимизации FLC-2 [14]. В табл. 2 представлены одинаковые результаты минимизации программами Espresso IIC и Minim системы полностью определенных булевых функций умножителя по модулю 5 (см. табл. 1). Если перейти от данной минимизированной системы ДНФ к таблице истинности, то будет ясно, что при минимизации системы частичных функций использовано *ненулевое* доопределение исходных частичных функций. В примере умножителя по модулю 5 обе программы совместной минимизации выдали одинаковый результат — 15 элементарных конъюнкций. Заметим, что в результате совместной минимизации каждая компонентная функция представлена своей ДНФ, так нет общих элементарных конъюнкций, входящих в различные ДНФ.

2.2. Минимизация в классе BDD представлений частичных функций

Графовые BDD-представления систем функций строятся на основе разложения Шеннона. Разложением Шеннона полностью определенной либо частичной булевой функции $f = f(\boldsymbol{x})$, $\boldsymbol{x} = (x_1, x_2, ..., x_n)$, по переменной x_i называется представление

$$f = f(\boldsymbol{x})\,\bar{x}_i f(x_1, ..., x_{i-1}, 0, x_{i+1}, ..., x_n) \vee \vee\, x_i f(x_1, ..., x_{i-1}, 1, x_{i+1}, ..., x_n). \quad (1)$$

Таблица 2

Минимизированные системы ДНФ функций умножителя по модулю 5

$a_1\ a_2\ a_3\ b_1\ b_2\ b_3$	$y_1\ y_2\ y_3$
– 1 0 – 1 0	1 0 0
– 1 1 – 1 1	1 0 0
1 – – – 0 1	1 0 0
– 0 1 1 – –	1 0 0
– 1 – – 0 1	0 1 0
– 0 1 – 1 –	0 1 0
– 1 – 1 – –	0 1 0
1 – – – 1 –	0 1 0
– 0 1 – – 1	0 0 1
– 1 1 – 1 0	0 0 1
– 1 0 – 1 1	0 0 1
– 1 0 1 – –	0 0 1
1 – – – 1 0	0 0 1
– – 1 – 0 1	0 0 1
1 – – 1 – –	0 0 1

Функции $f_0 = f(x_1, ..., x_{i-1}, 0, x_{i+1}, ..., x_n)$, $f_1 = f(x_1, ..., x_{i-1}, 1, x_{i+1}, ..., x_n)$ в правой части (1) называются кофакторами (*cofactors*) разложения по переменной x_i. Они получаются из функции f подстановкой вместо переменной x_i константы 0 и 1 соответственно. Каждая из подфункций f_0 и f_1 может быть разложена по одной из переменных из множества $\{x_1, ..., x_{i-1}, x_{i+1}, ..., x_n\}$. Процесс разложения подфункций заканчивается, когда все n переменных будут использованы для разложения, либо когда все подфункции выродятся до констант 0, 1, «-». Затем выполняется доопределение — замена неопределенных значений листовых определенными 0, 1 с целью минимизации числа кофакторов, т. е. используется *ненулевое* доопределение [10, с. 102]. В результате оптимизации происходит доопределение системы частичных функций до полностью определенных.

В примере многоуровневое представление системы полностью определенных булевых функций (табл. 1), реализующих (в виде 15 формул разложения Шеннона) исходные частичные булевы функции, имеет следующий вид:

$$y_1 = \bar{a}_2 s_1 \vee a_2 s_2;\ y_2 = \bar{a}_2 s_3 \vee a_2 s_{10};\ y_3 = \bar{a}_2 s_4 \vee a_2 s_5;$$

$$s_1 = \bar{a}_3 s_6 \vee a_3 s_7;\ s_2 = \bar{a}_3 s_8 \vee a_3 s_9;\ s_3 = \bar{a}_3 s_{14} \vee a_3 b_2;$$

$$s_4 = \bar{a}_3 s_{11} \vee a_3 b_3;\ s_5 = \bar{a}_3 s_{12} \vee a_3 s_{13};\ s_6 = b_3 s_{15};$$

$$s_7 = \bar{b}_3 s_{15};\ s_8 = \bar{b}_3 b_2;\ s_9 = b_3 b_2;\ s_{10} = \bar{b}_3 s_{15} \vee b_3 \bar{b}_2;$$

$$s_{11} = \bar{b}_3 s_{16};\ s_{12} = \bar{b}_3 s_{15} \vee b_3 b_2;\ s_{13} = \bar{b}_3 b_2 \vee b_3 \bar{b}_2;$$

$$s_{14} = b_2 s_{17};\ s_{15} = \bar{b}_2 s_{17};\ s_{16} = \bar{b}_2 s_{18} \vee b_2 a_1;$$

$$s_{17} = \bar{a}_1 b_1 \vee a_1;\ s_{18} = a_1 b_1.$$

2.3. Минимизация в классе BDDI представлений полностью определенных функций

Под BDDI (*Binary Decision Diagram with Inverse cofactors*) понимается ориентированный бесконтурный граф, задающий последовательные разложения Шеннона булевой функции $f(x_1, ..., x_n)$ по всем ее переменным $x_1, ..., x_n$ при заданном порядке (перестановке) переменных, по которым проводятся разложения, при условии нахождения пар взаимно инверсных кофакторов [15]. Заметим, что в BDD функциональная вершина реализуют один кофактор, взаимно инверсные кофакторы представляются парой вершин. Построение BDDI осуществлялось программой BDD_Builder при *нулевом* доопределении исходной системы частичных функций. Программа BDD_Builder [15] получила

следующие 28 взаимосвязанных формул разложений Шеннона:

$$y_1 = \bar{a}_2 s_0 \vee a_2 s_1;\quad y_2 = \bar{a}_2 s_2 \vee a_2 s_3;\quad y_3 = \bar{a}_2 s_4 \vee a_2 s_5;$$

$$s_0 = \bar{a}_1 s_{11} \vee a_1 s_{12};\quad s_1 = \bar{a}_1 s_{13};\quad s_2 = \bar{a}_1 s_6 \vee a_1 s_7;\quad s_3 = \bar{a}_1 s_8;$$

$$s_4 = \bar{a}_1 s_9 \vee a_1 s_{10};\quad s_5 = \bar{a}_1 s_{14};\quad s_6 = a_3 s_{22};\quad s_7 = \bar{a}_3 s_{22};$$

$$s_8 = \bar{b}_3 s_{23} \vee b_3 s_{17};\quad s_9 = b_3 s_{15};\quad s_{10} = \bar{b}_3 s_{19};\quad s_{11} = \bar{b}_3 s_{16};$$

$$s_{12} = b_3 s_{18};\quad s_{13} = \bar{b}_3 s_7 \vee b_3 s_6;\quad s_{14} = \bar{b}_3 s_{20} \vee b_3 s_{21};\quad s_{15} = a_3 \bar{b}_1;$$

$$s_{16} = a_3 s_{23};\quad s_{17} = \bar{b}_1 \bar{b}_2;\quad s_{18} = \bar{a}_3 s_{17};\quad s_{19} = \bar{a}_3 s_{24};$$

$$s_{20} = \bar{a}_3 s_{23} \vee a_3 s_{22};\quad s_{21} = \bar{a}_3 s_{22} \vee a_3 s_{17};\quad s_{22} = \bar{b}_1 b_2;$$

$$s_{23} = b_1 \bar{b}_2;\quad s_{24} = \bar{b}_1 b_2 \vee b_1 \bar{b}_2.$$

Можно отметить тот факт, что BDD-оптимизация частичных функций позволила получить лучший результат — 15 формул разложений Шеннона.

3. Вычислительные эксперименты

Исходными описаниями логики являлись системы ДНФ булевых функций на языке SF в формате SDF (матричных описаний) [14]. Для генерации таблиц истинности систем булевых функций использовалось моделирование алгоритмических VHDL-моделей умножителей на всех возможных входных наборах значений сигналов (на наборах из левой части таблицы истинности). Для умножителя по модулю 5 пример такой модели представлен в листинге 1. Заметим, что эти же модели использовались и для схемной реализации умножителей в системе Vivado, так как в этой системе операция *rem* является синтезируемой. В параметризованной алгоритмической VHDL-модели модулярных умножителей число n задает число разрядов для представления чисел a, b, находящихся в диапазоне $\{0, 1, ..., p-1\}$. Для умножителя по модулю 5 число разрядов n для представления чисел 0, ..., 4 равно 3, число $p = 5$ (листинг 1). В листинге 1 параметры n, p выделены полужирным шрифтом. Для умножителя по модулю $p = 100$: $n = 7$ — нужно семь разрядов для представления чисел из множества {0, 1, 2, ..., 99, 100}.

```
Library ieee;
use ieee.std _ logic _ 1164.all;
use ieee.numeric _ std.all;
entity mult _ mod _ p _ n is
generic(n: integer := 3;
   p : integer := 5);
   port (
   a, b: in std _ logic _ vector(1 to n);
   y: out std _ logic _ vector (1 to n));
end mult _ mod _ p _ n;
architecture beh of mult _ mod _ p _ n is
function mult
(signal a, b : std _ logic _ vector(1 to n);
constant n : integer;
constant p : integer)
return std _ logic _ vector is
variable y : std _ logic _ vector(1 to n);
variable a _ int, b _ int, m : integer range
0 to (2**(n))-1 := 0;
variable r, e: integer range 0 to 49 := 0;
   begin
a _ int := to _ integer(unsigned(a));
b _ int := to _ integer(unsigned(b));
r := a _ int * b _ int;
m := r rem p;
-- в ISE и Vivado операция rem является
-- синтезируемой
    if ((a _ int >= p) or (b _ int >= p)) then
       y := ('-','-','-');
   end if;
   if ((r < p) and (a _ int < p) and (b _ int < p))
      then e := r;
      y := std _ logic _ vector(to _ unsigned(e, n));
   elsif ((r > = p) and (a _ int < p) and
      (b _ int < p)) then e := m;
      y := std _ logic _ vector(to _ unsigned(e, n));
   end if;
return y;
end mult;
     begin
   y <= mult (a, b, 3, 5);
end architecture beh;
```

Листинг 1. Алгоритмическое VHDL-описание умножителя по модулю пять

В системе FLC-2 [14] выполнялась логическая оптимизация и осуществлялся перевод оптимизированных SF-описаний в описания на языке VHDL. Схемная реализация полученных VHDL-описаний в библиотеке проектирования отечественных КМОП СБИС выполнялась с помощью синтезатора LeonardoSpectrum (эксперименты 1—6, 9). Функции логических КМОП-элементов используемой библиотеки синтеза приведены в работе [16]. Для каждого описания схемы модулярного умножителя синтез осуществлялся с одними и теми же опциями управления синтезом. Для каждой полученной схемы подсчитывались площадь схемы (сумма площадей элементов) в условных единицах, число логических элементов для оценки площади кристалла, требуемой для разводки межсоединений, и временная задержка.

3.1. Синтез схем модулярных умножителей в системе LeonardoSpectrum

Эксперимент 1. Синтез схем модулярных умножителей по алгоритмическим VHDL-описаниям.

Алгоритмические описания были получены заменой несинтезируемого оператора *m:=r rem p;* синтезируемой VHDL-функцией деления целых чисел, представленных в двоичном коде, при этом использовалось для вычислений только значение остатка, хотя функция вычисляла и частное от деления двух целых неотрицательных чисел.

Эксперимент 2. Предварительная логическая оптимизация в классе ДНФ программой Minim [9] и последующий синтез схемы. Использовалось *ненулевое* доопределение на этапе исходного задания и логической оптимизации.

Эксперимент 3. Предварительная логическая оптимизация в классе ДНФ программой Espresso IIC [8] и последующий синтез схемы. Использовалось *ненулевое* доопределение на этапе исходного задания и логической оптимизации.

Эксперимент 4. Предварительная логическая оптимизация в классе BDDI представлений с помощью программы BDD_Builder [15] и последующий синтез схемы. Использовалось *нулевое* доопределение на этапе исходного задания и оптимизации.

Эксперимент 5. Предварительная логическая оптимизация в классе BDD представлений частичных функций [10, с. 201] и последующий синтез схемы. Использовалось *ненулевое* доопределение на этапе исходного задания и логической оптимизации.

Эксперимент 6. Синтез схем по неоптимизированным VHDL-моделям систем частичных функций модулярных умножителей.

Пример такой модели для умножителя по модулю 5 (см. табл. 1) дан в листинге 2. Модель состоит из одного синтезируемого оператора условного назначения сигнала и перечисляет наборы из области определенных значений и значения функций на этих наборах. Вся область неопределенных значений описывается одной строкой (“---";). Входной порт (*x: in std_logic_vector (1 to 6);*) обозначает конкатенацию входных векторов *a*, *b* из листинга 1.

```
Library ieee;
use ieee.std _ logic _ 1164.all;
entity mult _ mod _ 5 is
   port (x: in std _ logic _ vector (1 to 6);
         y: out std _ logic _ vector (1 to 3));
   end;
architecture BEHAVIOR of mult _ mod _ 5 is
begin
y <= "000" when x = "000000" else
     "000" when x = "000001" else
     "000" when x = "000010" else
     "000" when x = "000011" else
     "000" when x = "000100" else
     "000" when x = "001000" else
     "001" when x = "001001" else
     "010" when x = "001010" else
     "011" when x = "001011" else
     "100" when x = "001100" else
     "000" when x = "010000" else
     "010" when x = "010001" else
     "100" when x = "010010" else
     "001" when x = "010011" else
     "011" when x = "010100" else
     "000" when x = "011000" else
     "011" when x = "011001" else
     "001" when x = "011010" else
     "100" when x = "011011" else
     "010" when x = "011100" else
     "000" when x = "100000" else
     "100" when x = "100001" else
     "011" when x = "100010" else
     "010" when x = "100011" else
     "001" when x = "100100" else
     "---";
end BEHAVIOR;
```

Листинг 2. Алгоритмическое VHDL-описание умножителя модулю 5, соответствующее системе частичных функций (см. табл. 1)

Результаты экспериментов 1—6 приведены в табл. 3, 4.

3.2. Синтез схем модулярных умножителей в системе ISE Xilinx

Эксперимент 7. Синтез в системе FPGA-структур, реализующих схемы модулярных умножителей.

В качестве базовой (целевой) микросхемы использовалась микросхема FPGA xc6slx4 семейства Spartan 6. Сложность схемной реализации оценивалась в числе программируемых элементов LUT-6, имеющих 6 входных переменных (LUT — Look-Up Table — таблица, реализующая логическую функцию). Для эксперимента 7 использовались те же VHDL-модели, что и в экспериментах 1—6. В системе ISE операторы *rem*, *mod* являются синтезируемыми. Для экспериментов в системе ISE использовались те же VHDL-модели, что и в экспериментах 1—6 и два дополнительных вида VHDL-описаний: первое из таких описаний содержало оператор *rem* (см. листинг 1), второе (листинг 3) — содержало единственный синтезируемый оператор *y <= (a*b) mod p.*

Результаты эксперимента 7 приведены в табл. 5, 6.

```
Library ieee;
use ieee.std _ logic _ 1164.all;
use ieee.numeric _ std.all;
entity mod _ 5 is
port (a, b: in unsigned (1 to 3);
y: out unsigned (1 to 6));
end;
architecture beh of mod _ 5 is
begin
   y <= (a * b) mod 5;
-- в ISE и Vivado операция mod является
-- синтезируемой
end beh;
```

Листинг 3. Алгоритмическое VHDL-описание умножителя модулю 5 с одним синтезируемым оператором `(a*b) mod 5` (для экспериментов в системах ISE и Vivado)

Таблица 3

Результаты экспериментов 1—3 в системе LeonardoSpectrum, библиотека проектирования заказных СБИС

Схема	*n, m, k*	Алгоритмическая модель (схемная реализация операции деления) **Эксперимент 1**			Совместная минимизация в классе ДНФ (программа Minim) **Эксперимент 2**				Совместная минимизация в классе ДНФ (программа Espresso) **Эксперимент 3**			
		S	*L*	τ, нс	k_{min}	*S*	*L*	τ, нс	k_{min}	*S*	*L*	τ, нс
Mult_mod_5	6,3,25	50 694	122	20,83	15	***6 769**	19	3,28	15	6 819	19	3,28
Mult_mod_7	6,3,49	45 103	109	16,57	23	11 824	30	2,72	18	11 964	35	2,61
Mult_mod_9	8,4,81	105 339	235	38,43	51	29 853	***77**	***3,57**	49	***29 211**	***77**	4,28
Mult_mod_13	8,4,169	102 030	232	37,91	100	62 596	163	***4,68**	96	65 799	174	5,58
Mult_mod_15	8,4,225	103 375	231	34,94	122	73 689	196	5,85	112	74 917	200	6,27
Mult_mod_17	10,5,289	189 865	415	65,25	192	118 737	304	7,19	180	115 484	293	7,54
Mult_mod_29	10,5,841	***166 189**	***375**	56,27	577	366 249	921	9,71	525	383 647	968	10,83
Mult_mod_33	12,6,1089	***295 589**	***633**	97,19	720	459 206	1 159	11,62	645	520 491	1 324	10,11
Mult_mod_41	12,6,1681	***267 907**	***582**	84,84	1271	814 758	2 130	12,54	1163	1 000 784	2 559	11,54
Mult_mod_63	12,6,3969	***254 911**	***558**	73,71	2757	2 385 684	6 067	17,49	2567	2 475 243	6 458	15,09

Таблица 4

Результаты экспериментов 4—6 в системе LeonardoSpectrum, библиотека проектирования заказных СБИС

Схема	*n, m, k*	Минимизация BDDI **Эксперимент 4**			VHDL-модель системы частичных функций **Эксперимент 5**			Минимизация BDD частичных функций **Эксперимент 6**		
		S	*L*	τ, нс	*S*	*L*	τ, нс	*S*	*L*	τ, нс
Mult_mod_5	6,3,25	10 602	30	3,20	9 012	24	3,91	6 863	***16**	***2,26**
Mult_mod_7	6,3,49	13 883	37	2,95	***10 608**	*26	2,10	12 159	30	2,58
Mult_mod_9	8,4,81	33 843	93	4,60	35 266	98	4,85	32 085	90	5,20
Mult_mod_13	8,4,169	68 885	187	5,44	***56 570**	***143**	5,39	64 042	172	5,73
Mult_mod_15	8,4,225	63 143	173	5,69	71 084	184	***5,50**	***46 727**	***114**	5,57
Mult_mod_17	10,5,289	110 099	300	***6,52**	***106 014**	***286**	7,59	111 539	292	6,74
Mult_mod_29	10,5,841	345 475	861	9,08	421 374	1 027	*7,04	265 708	699	8,54
Mult_mod_33	12,6,1089	348 471	887	***9,67**	527 165	1 425	11,78	311 107	820	9,76
Mult_mod_41	12,6,1681	733 630	1 892	***10,70**	804 084	2 063	11,26	667 999	1 702	11,15
Mult_mod_63	12,6,3969	864 124	2 294	11,80	2 066 787	4 829	*9,84	414 951	1 065	10,74

Таблица 5

Результаты эксперимента 7 в системе ISE Xilinx, микросхема xc6slx4 FPGA семейства Spartan 6

Схема	Алгоритмическая модель		Программа Minim		Программа Espresso	
	LUT-6	τ, нс	LUT-6	τ, нс	LUT-6	τ, нс
Mult_mod_5	***3**	***5,590**	***3**	***5,590**	***3**	***5,590**
Mult_mod_7	***3**	***5,590**	***3**	***5,590**	***3**	***5,590**
Mult_mod_9	57	11,372	16	6,902	15	***6,861**
Mult_mod_13	60	11,448	61	10,320	53	9,218
Mult_mod_15	52	21,515	57	9,067	27	8,110
Mult_mod_17	86	32,375	124	10,755	116	10,762
Mult_mod_29	75	28,967	396	13,290	378	13,332
Mult_mod_33	180	32,018	500	13,977	486	14,553
Mult_mod_41	172	30,852	864	15,050	866	14,998
Mult_mod_63	167	30,331	237	11,929	248	12,920

Таблица 6

Результаты эксперимента 7 в системе ISE Xilinx, микросхема xc6slx4 FPGA семейства Spartan 6

Схема	Минимизация BDDI		VHDL-модель частичных функций		Минимзация BDD частичных функций		Алгоритмическая модель с оператором *rem*		Один оператор (*a**b) mod *p*	
	LUT-6	τ, нс	LUT-6	τ, нс	LUT-6	τ, нс	LUT-6	τ, нс	LUT-6	τ, нс
Mult_mod_5	***3**	***5,590**	***3**	***5,590**	*3	***5,590**	19	11,191	15	10,302
Mult_mod_7	***3**	***5,590**	***3**	***5,590**	***3**	***5,590**	18	10,077	13	9,320
Mult_mod_9	***13**	6,994	17	6,927	17	6,972	32	13,229	27	13,193
Mult_mod_13	38	8,250	***16**	***6,236**	40	8,582	39	15,767	32	18,385
Mult_mod_15	46	9,081	***16**	***6,236**	24	8,093	38	17,945	32	17,744
Mult_mod_17	74	10,871	74	***9,424**	76	9,653	65	21,230	***58**	21,086
Mult_mod_29	286	11,844	90	***7,960**	224	11,249	65	25,281	***59**	25,136
Mult_mod_33	312	12,547	2 241	163,831	269	***12,531**	108	26,287	***101**	25,776
Mult_mod_41	641	***12,983**	4 328	232,936	620	13,384	96	32,119	***87**	31,328
Mult_mod_63	205	***11,712**	–	–	346	12,739	109	25,928	***100**	25,119

3.3. Синтез схем модулярных умножителей в системе Vivado

Эксперимент 8. Синтез в системе Vivado FPGA структур, реализующих схемы модулярных умножителей.

Схемная реализация VHDL-описаний модулярных умножителей для FPGA xc7k70tfbv676-1 семейства Kintex-7 осуществлялась в системе автоматизированного проектирования Vivado, опции синтеза — Vivado Synthesis Default. Сложность схемы оценивалась в числе программируемых элементов LUT-6. Результаты эксперимента 8 приведены в табл. 7, 8. Для эксперимента 8 использовались те же VHDL-модели, что и в эксперименте 7.

3.4. Сравнение аппаратных реализаций для значений модуля *p* = 65, 101, 127, 129, 255

Эксперимент 9. Сравнение аппаратных реализаций в библиотеке проектирования заказных КМОП СБИС модулярных умножителей по алгоритмическим VHDL-описаниям и VHDL-моделей, полученных BDDI-минимизацией систем полностью определенных функций.

Результаты эксперимента 9 приведены в табл. 9.

Эксперимент 10. Сравнение аппаратных реализаций в системе Vivado VHDL-операций *(a*b) mod p* и VHDL-моделей, полученных BDDI-минимизацией систем полностью определенных функций.

Результаты эксперимента 10 приведены в табл. 10.

3.5. Аппаратная реализация операции деления в системе LeonardoSpectrum

Эксперимент 11. Сравнение аппаратных реализаций VHDL-операций деления и вычисления остатка в библиотеке проектирования заказных КМОП СБИС. Результаты эксперимента 11 приведены в табл. 11.

Таблица 7

Результаты эксперимента 8 в системе VIVADO, микросхема xc7k70tfbv676-1 семейства Kintex7

Схема	Алгоритмическая модель		Программа Minim		Программа Espresso	
	LUT-6	τ, нс	LUT-6	τ, нс	LUT-6	τ, нс
Mult_mod_5	35	12,442	***3**	5,794	***3**	5,978
Mult_mod_7	29	11,242	***3**	5,981	***3**	***5,907**
Mult_mod_9	68	16,166	***11**	6,831	15	6,752
Mult_mod_13	59	15,240	***16**	6,475	***16**	6,655
Mult_mod_15	56	15,240	***16**	6,419	***16**	***6,347**
Mult_mod_17	106	22,707	72	7,809	46	7,770
Mult_mod_29	98	21,070	85	7,865	85	***7,734**
Mult_mod_33	157	25,538	305	10,694	132	10,103
Mult_mod_41	139	24,191	411	9,742	383	10,471
Mult_mod_63	137	23,748	164	9,753	408	9,799

Таблица 8

Результаты эксперимента 8 в системе VIVADO, микросхема xc7k70tfbv676-1 семейства Kintex7

Схема	Минимизация BDDI		VHDL-модель частичных функций		Минимизация BDD частичных функций		Алгоритмическая модель с оператором *rem*		Один оператор (*a***b*) mod *p*	
	LUT-6	τ, нс	LUT-6	τ, нс	LUT-6	τ, нс	LUT-6	τ, нс	LUT-6	τ, нс
Mult_mod_5	***3**	5,978	***3**	5,978	***3**	5,974	***3**	5,974	***3**	***5,971**
Mult_mod_7	***3**	5,974	***3**	5,915	***3**	5,978	***3**	5,974	***3**	5,971
Mult_mod_9	15	6,823	15	6,852	13	***6,475**	39	9,744	32	8,422
Mult_mod_13	***16**	6,475	20	6,739	***16**	***6,343**	39	9,927	32	8,545
Mult_mod_15	***16**	6,461	20	5,086	***16**	6,376	39	9,681	32	8,474
Mult_mod_17	46	7,583	36	7,388	***35**	***7,567**	57	13,153	47	12,060
Mult_mod_29	85	7,969	99	7,979	84	7,984	64	13,232	***55**	12,824
Mult_mod_33	132	8,960	330	10,198	128	***8,558**	93	16,409	***89**	14,952
Mult_mod_41	383	10,599	295	9,993	254	***9,722**	88	15,624	***79**	15,406
Mult_mod_63	408	10,101	214	9,554	171	***9,151**	70	16,184	***68**	15,023

Таблица 9

Результаты эксперимента 9 в системе LeonardoSpectrum, библиотека проектирования заказных СБИС

Схема	Алгоритмическая модель (схемная реализация операции деления)			Минимизация BDDI			Минимизация BDD частичных функций		
	S	*L*	τ, нс	*S*	*L*	τ, нс	*S*	*L*	τ, нс
Mult_mod_65	***421 837**	***883**	133,10	1 560 302	3 910	13,19	1 125 380	3 043	***12,11**
Mult_mod_101	***360 016**	***780**	109,74	3 172 855	8 267	15,82	3 234 905	8 410	***14,28**
Mult_mod_127	***362 064**	***778**	98,26	2 141 091	5 830	***13,88**	1 241 087	3 283	14,10
Mult_mod_129	***566 085**	***1 173**	179,97	4 291 238	10 580	***16,11**	3 186 805	8 160	16,18
Mult_mod_255	***481 197**	***1 057**	135,17	5 354 250	13 098	***15,77**	–	–	–

Таблица 10

Результаты эксперимента 10 в системе Vivado, микросхема xc7k70tfbv676-1 семейства Kintex7

Схема	Минимизация BDDI		Минимизация BDD частичных функций		Один оператор (*a***b*) mod *p*	
	LUT-6	τ, нс	LUT-6	τ, нс	LUT-6	τ, нс
Mult_mod_65	493	***12,134**	487	12,228	***101**	14,862
Mult_mod_101	1 608	15,530	1 436	16,477	***97**	***15,443**
Mult_mod_127	1 934	16,695	694	***12,347**	***90**	15,214
Mult_mod_129	2 169	18,404	1 796	17,837	***150**	***16,260**
Mult_mod_255	3 081	19,034	–	–	***117**	***16,121**

Таблица 11

Результаты эксперимента 11 в системе LeonardoSpectrum, библиотека проектирования заказных СБИС

Разрядность входных операндов операции деления	VHDL-операции деления — вычисление частного и остатка (*/*, *rem*)			VHDL-операция вычисления остатка (*rem*)		
	S	*L*	τ, нс	*S*	*L*	τ, нс
3	37 180	80	11,74	35 109	69	13,48
4	72 562	160	16,56	62 931	125	24,93
5	114 942	257	22,48	95 876	187	36,45
6	158 556	346	33,99	134 171	262	49,27
8	250 336	519	71,17	227 898	446	78,33
10	374 000	779	109,90	342 612	671	114,08
12	503 489	1 016	163,60	489 165	972	159,61

4. Анализ результатов экспериментов

Результаты экспериментов 1—11 приведены в табл. 3—11. Имя примера Mult_mod_*p* соответствует умножителю по модулю *p*. Далее в таблицах, задающих результаты экспериментов 1—6, используются следующие обозначения параметров систем функций: *n* — число переменных; *m* — число функций; *k* — число строк в таблице истинности, для которых функции умножителя определены (не являются строками, состоящими только из "–"); $k_{\min}$ — число *общих* элементарных конъюнкций, входящих в совместно минимизированные ДНФ всех *m* функций системы;

Для рассмотренного примера модулярного умножителя (см. табл. 1): $n = 6$, $m = 3$, $k = 25$. Результаты совместной *м*инимизации в классе ДНФ системы функций умножителя по модулю 5 представлены в табл. 2. В табл. 2 система ДНФ характеризуется следующими параметрами: $n = 4$, $m = 2$, $k_{\min} = 15$. Логические схемы из библиотечных КМОП-элементов оценивались следующими параметрами: *S* — площадь (в условных единицах); *L* — число логических элементов; τ — задержка в наносекундах.

Для экспериментов 1—10 символом * отмечены лучшие решения по площади, числу логических элементов и задержке. Как уже говорилось, сложность структур FPGA оценивалась в числе программируемых элементов LUT-6.

Результаты экспериментов 1—6 свидетельствуют о том, что описания модулярных умножителей системами частичных булевых функций и последующая логическая минимизация позволяют получать логические схемы из библиотечных КМОП-элементов, характеризуемые небольшой задержкой по сравнению со схемами, получаемыми на основе алгоритмической модели: задержки схем, синтезируемых по алгоритмическим моделям (см. табл. 3, эксперимент 1) в 7—8 раз больше соответствующих задержек схем, синтезируемых по оптимизированным логическим моделям. Однако площади схем, получаемых по алгоритмическим моделям, в несколько раз меньше. Проектировщики могут выбирать соответствующие схемные решения, рассматривая пару соответствующих параметров «площадь-задержка».

Результаты эксперимента 7. Для значений модуля $p_i = 29$ (и больше) сложность структур FPGA, синтезируемых в ISE, значительно меньше сложности структур, синтезируемых по оптимизированным логическим моделям. Однако для меньших значений модуля имеет смысл использовать оптимизированные формы систем булевых функций, либо подавать на вход системы ISE неоптимизированные VHDL-описания систем частичных функций (пример в листинге 2), тогда предварительную логическую оптимизацию выполняет синтезатор XST, входящий в систему ISE.

Результаты эксперимента 8. Для значений модуля $p \leqslant 17$ сложность структур FPGA, синтезируемых в Vivado по оптимизированным логическим представлениям систем булевых функций, реализующих модулярные умножители, оказывается меньше, чем сложность структур, реализующих операцию (*a*b*) mod *p*. Однако для значений $p > 17$ целесообразно использовать VHDL-операцию (*a*b*) mod *p*.

Результаты эксперимента 9. Получение быстродействующих аппаратных реализаций для значений модуля $p = 65, 101, 127, 129, 255$ возможно на основе многоуровневых реализаций систем частичных функций в классе BDD. Оптимизация в классе BDDI систем полностью определенных функций (при нулевом доопределении) проигрывает данному виду логической оптимизации по аппаратной сложности. Получение схем меньшей сложности для данных значений модуля возможно на основе алгоритмической модели, однако получаемые задержки схем во много раз превосходят задержки схем, получаемых на основе BDD-минимизации. Значительный выигрыш в быстродействии имеет обратную сторону — проигрыш (примерно в столько же раз) в числе логических элементов (в площади логической схемы).

Результаты эксперимента 10. Алгоритмы (и программы) реализации VHDL-операций (*a*b*) mod *p* в системе Vivado являются более эффективными соответствующих алгоритмов, реализованных в системе ISE Xilinx. Однако для небольших значений модуля ($p \leqslant 63$) получение быстродействующих и менее сложных реализаций возможно на основе оптимизации в классе BDD систем частичных функций, описывающих функционирование модулярных умножителей. Заметим, что подобный эффект наблюдался и при схемной реализации цепочек арифметических операций, результаты экспериментов приведены в работе [17].

Результаты эксперимента 11. Основную долю аппаратных затрат при схемной реализации VHDL-операции (*/*, *rem*) составляет реализация операции *rem* вычисления остатка от деления двух целых неотрицательных чисел, представленных в двоичном коде. Заметим, что схема деления вычисляет и значения выходных сигналов при делении на нуль, однако нулевое значение делителя не должно подаваться на вход схемы.

Заключение

При схемной реализации модулярных умножителей как в библиотеке проектирования заказных СБИС, так и в составе FPGA для небольших значений модуля ($p_i \leqslant 29$) целесообразно составлять таблицы истинности соответствующих систем неполностью определенных булевых функций, проводить логическую оптимизацию в различных классах их представлений (ДНФ, BDD), получать соответствующие VHDL-описания и выполнять их аппаратные реализации. Однако для больших значений модуля системы булевых функций являются громоздкими, поэтому алгоритмические модели будут давать схемы меньшей сложности, при этом задержки (число логических каскадов) таких схем в библиотеке проектирования заказных КМОП СБИС будет достаточно большим, так как основную «лепту» в число уровней вносит аппаратная реализация вычисления остатка от деления. Модулярные умножители для небольших значений модуля используются в системах параллельной обработки, например, в работе [18] используется система модулей (3, 4, 5), в работе [19] — система модулей (31, 32, 63). При большом числе модулярных операций, требуемых при параллельной обработке информации, «капельный» эффект, достигаемый при сокращении сложности одного модулярного умножителя, может быть значительным. Например, при реализации схемы Mult_mod_17 найденное лучшее решение (см. табл. 8) имеет сложность 35 LUT, а реализация VHDL операции *mod* в Vivado — 47 LUT. При параллельной обработке 1000 подсхем выигрыш может достигать 12 000 LUT. Однако при реализации модулярных умножителей в составе FPGA для $p > 17$ целесообразно использовать синтезируемую VHDL-операцию $(a*b) \bmod p$.

Список литературы

1. **Акушский И. Я., Юдицкий Д. И.** Машинная арифметика в остаточных классах. М.: Советское радио, 1968. 440 с.

2. **Червяков Н. И., Сахнюк П. А., Шапошников А. В., Рядов С. А.** Модулярные параллельные вычислительные структуры нейропроцессорных систем. М.: Физматлит, 2003. 288 с.

3. **Стемпковский А. Л., Корнилов А. И., Семенов М. Ю.** Особенности реализации устройств с цифровой обработкой сигналов в интегральном исполнении с применением модулярной арифметики // Информационные технологии. 2004. № 2. С. 2—9.

4. **Червяков Н. И., Дьяченко И. В.** Принципы построения модулярных сумматоров и умножителей // Модулярная арифметика: мат-лы междунар. науч. конф. URL: http://www.computer-museum.ru/histussr/sokconf0.htm (дата обращения 29.03.2022).

5. **Амербаев В. М., Соловьев Р. А., Тельпухов Д. В.** Реализация библиотеки модульных арифметических операций на основе алгоритмов минимизации логических функций // Известия ЮФУ. Технические науки, Таганрог. 2013. № 7. С. 221—225.

6. **Амербаев В. М., Соловьев Р. А., Тельпухов Д. В., Щелоков А. Н.** Исследование эффективности модулярных вычислительных структур при проектировании аппаратных однотактных умножителей // Известия ЮФУ. Технические науки. Таганрог. 2014. № 7 (156). С. 248—254.

7. **Соловьев Р. А., Тельпухов Д. В., Балака Е. С.** и др. Особенности проектирования модулярных умножителей с помощью современных САПР // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС). 2016. № 1. С. 249—254.

8. **Brayton K. R., Hactel G. D., McMullen C. T., Sangiovanni-Vincentelli A. L.** Logic minimization algorithm for VLSI synthesis. Boston, e.a.: Kluwer Academic Publishers, 1984. 193 p.

9. **Торопов Н. Р.** Минимизация систем булевых функций в классе ДНФ // Логическое проектирование. Минск: Ин-т техн. кибернетики НАН Беларуси, 1999. Вып. 4. С. 4—19.

10. **Бибило П. Н.** Применение диаграмм двоичного выбора при синтезе логических схем. Минск: Беларус. навука, 2014. 231 с.

11. **Бибило П. Н.** Системы проектирования интегральных схем на основе языка VHDL. StateCAD, ModelSim, LeonardoSpectrum. М.: СОЛОН-Пресс, 2005. 384 с.

12. **Соловьев В. В.** Архитектуры ПЛИС фирмы Xilinx: FPGA и CPLD 7-й серии. М.: Горячая линия — Телеком, 2016. 392 с.

13. **Тарасов И. Е.** ПЛИС Xilinx. Языки описания аппаратуры VHDL и Verilog, САПР, приемы проектирования. М.: Горячая линия — Телеком, 2020. 538 с.

14. **Бибило П. Н., Романов В. И.** Система логической оптимизации функционально-структурных описаний цифровых устройств на основе продукционно-фреймовой модели представления знаний // Проблемы разработки перспективных микро- и наноэлектронных систем. 2020. Сб. трудов / под общ. ред. акад. РАН А. Л. Стемпковского. М.: ИППМ РАН, 2020. № 4. С. 9—16. DOI: 10.31114/2078-7707-2020-4-9-16.

15. **Бибило П. Н., Ланкевич Ю. Ю.** Использование полиномов Жегалкина при минимизации многоуровневых представлений систем булевых функций на основе разложения Шеннона // Программная инженерия. 2017. Том 8, № 8. С. 369—84. DOI: 10.17587/prin.8.369-384.

16. **Bibilo P. N., Kirienko N. A.** Estimating Energy Consumption in Logical CMOS Circuits Based on Their Switching Activity // Russian Microelectronics. 2012. Vol. 41, No. 1. P. 59—70. DOI: 10.1134/S1063739711060035.

17. **Бибило П. Н., Романов В. И.** Интеграция САПР для синтеза логических схем с использованием глобальной оптимизации // Программные продукты и системы. 2019. Том 32, № 1. С. 26—33. DOI: 10.15827/0236-235X.125.026-033.

18. **Salamat S., Shubhi S., Khaleghi B., Rosing T.** Residue-Net: Multiplication-free Neural Network by In-situ No-loss Migration to Residue Number Systems // 26th Asia and South Pacific Design Automation Conference (ASP-DAC), 2021. P. 222—228.

19. **Samimi N., Kamal M., Afzali-Kusha A., Pedram M.** Res-DNN: A residue number system-based DNN accelerator unit. // IEEE Trans. Circuits Syst. 2020. Vol. 67, No. 2. P. 658—671. DOI: 10.1109/TCSI.2019.2951083.

Synthesis of Modular Multipliers

P. N. Bibilo, bibilo@newman.bas-net.by
The United Institute of Informatics Problems of the National Academy of Sciences of Belarus, Minsk, 220012, Belarus

Corresponding author:

Bibilo Petr N., Head of Laboratory,
The United Institute of Informatics Problems of the National Academy of Sciences of Belarus, Minsk, 220012, Belarus,
E-mail: petr.olibib@yandex.ru; bibilo@newman.bas-net.by

Received on May 22, 2023
Accepted on June 08, 2023

*The results of experiments on the circuit implementation of modular multipliers in the design library of ASIC (Application-Specific Integrated Circuits) and FPGA (Field-Programmable Gate Array) are presented. The initial descriptions of modular multiplier projects were given by systems of not fully defined (partial) Boolean functions and algorithmic VHDL descriptions. Logical optimization was carried out in the class of disjunctive normal forms (DNF) and representations of Boolean function systems by BDD (Binary Decision Diagrams). The synthesized circuits were evaluated by area and time delay. It is established that the use of partial Boolean function models and preliminary logical BDD optimization allows one to improve the parameters of synthesized ASIC and FPGA blocks for small module values, however, the best solutions for large module values can be obtained using algorithmic VHDL descriptions of modular multipliers. In the synthesis of modular multiplier circuits as part of an FPGA and the use of ISE and Vivado design systems it is advisable to use synthesized VHDL operations (a*b) mod p.*

Keywords: *modular calculations, modular multiplier, integer division operation, system of Boolean functions, Disjunctive Normal Form, Binary Decision Diagram, Shannon expansion, digital logic synthesis, VHDL, ASIC, FPGA*

For citation:

Bibilo P. N. Synthesis of Modular Multipliers, *Programmnaya Ingeneria,* 2023, vol. 14, no. 8, pp. 377—387. DOI: 10.17587/prin.14.377-387.

References

1. **Akushskij I.YA., Yudickij D. I.** *Machine arithmetic in residual classes*, Moscow, Sovetskoe radio, 1968, 440 p. (in Russian).
2. **Chervyakov N. I., Sakhnyuk P. A., Shaposhnikov A. V., Ryadnov S. A.** *Modular parallel computing structures of neuroprocessor systems*, Moscow, Fizmatlit, 2003, 288 p. (in Russian).
3. **Stempkovskiy A. L., Kornilov A. I., Semenov M. Yu.** Features of the implementation of devices with digital signal processing in the integrated design using modular arithmetic, *Informatsionnye tekhnologii*, 2004, no. 2, pp. 2—9 (in Russian).
4. **Chervyakov N. I., D'yachenko I. V.** Principles of construction of modular adders and multipliers, *Modulyarnaya arifmetika: mat-ly mezhdunar. nauch. konf.*, available at: http://www.computer-museum.ru/histussr/sokconf0.htm (in Russian).
5. **Amerbaev V. M., Solov'ev R. A., Tel'puhov D. V.** Implementation of a library of modular arithmetic operations based on algorithms for minimizing logical functions, *Izvestiya YuFu, Tekhnicheskie nauki,* 2013, no. 7, pp. 221—225 (in Russian).
6. **Amerbaev V. M., Solov'ev R. A., Tel'puhov D. V., Shchelokov A. N.** Investigation of the effectiveness of modular computing structures in the design of hardware single-stroke multipliers, *Izvestiya YuFu, Tekhnicheskie nauki*, 2014, no. 7 (156), pp. 248—254 (in Russian).
7. **Telpukhov D. V., Sovovyev R. A., Balaka E. S.** et al. Design features of the RNS-based multipliers using advanced CAD, *Problemy razrabotki perspektivnyh mikro- i nanoelektronnyh sistem (MES)*, 2016, no. 1, pp. 249—254.
8. **Brayton K. R., Hachtel G. D., McMullen C., Sangiovanni-Vincentelli A. L.** *Logic Minimization Algorithm for VLSI Synthesis,* Boston, Kluwer Academic Publishers, 1984, 193 p.
9. **Toropov N. R.** Minimization of systems of Boolean functions in the class DNF, *Logicheskoe proektirovanie*, Minsk, Institut tehnicheskoj kibernetiki Nacional'noj akademii nauk Belarusi, 1999, no. 4, pp. 4—19 (in Russian).
10. **Bibilo P. N.** *Application of Binary Decision Diagrams in the Synthesis of Logic Circuits,* Minsk, Belaruskaja navuka, 2014, 231 p. (in Russian).
11. **Bibilo P. N.** *Integrated Circuit Design Systems Based on the VHDL Language. StateCAD, ModelSim, LeonardoSpectrum,* Moscow, SOLON-Press Publ., 2005. 384 p. (in Russian).
12. **Solovyov V. V.** *XILINX FPGA Architectures: FPGA and CPLD 7-Series.* Moscow, Goryachaya liniya — Telekom, 2016. 392 p. (in Russian).
13. **Tarasov I. E.** *XILINX FPGA. Hardware Description Languages VHDL and Verilog, CAD, Design Techniques,* Moscow, Goryachaya liniya — Telekom, 2020, 538 p. (in Russian).
14. **Bibilo P. N., Romanov V. I.** The system of logical optimization of functional structural descriptions of digital circuits based on production-frame knowledge representation model, *Problemy razrabotki perspektivnyh mikro- i nanoelektronnyh sistem*, 2020, Sb. trudov / Eds akad. RAN A. L. Stempkovskiy. Moscow, IPPM RAN, 2020, no. 4, pp. 9—16. DOI: 10.31114/2078-7707-2020-4-9-16.
15. **Bibilo P. N., Lankevich Yu. Yu.** The Use of Zhegalkin Polynomials for Minimization of Multilevel Representations of Boolean Functions Based on Shannon Expansion, *Programmnaya Ingeneria*, 2017, vol. 8, no. 8, pp. 369—384. DOI: 10.17587/prin.8.369-384 (in Russian).
16. **Bibilo P. N., Kirienko N. A.** Estimating Energy Consumption in Logical CMOS Circuits Based on Their Switching Activity, *Russian Microelectronics*, 2012, vol. 41, no. 1, pp. 59—70. DOI: 10.1134/S1063739711060035.
17. **Bibilo P. N., Romanov V. I.** CAD integration for logic synthesis using global optimization, *Software & Systems*, 2019, vol. 32, no. 1, pp. 26—33. DOI:10.15827/0236-235X.125.026-033 (in Russian).
18. **Salamat S., Shubhi S., Khaleghi B., Rosing T.** Residue-Net: Multiplication-free Neural Network by In-situ No-loss Migration to Residue Number Systems. *26th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2021, pp. 222—228.
19. **Samimi N., Kamal M., Afzali-Kusha A., Pedram M**. Res-DNN: A residue number system-based DNN accelerator unit, *IEEE Trans. Circuits Syst.*, 2020, vol. 67, no. 2, pp. 658—671. DOI: 10.1109/TCSI.2019.2951083.

УДК 004.58
DOI: 10.17587/prin.14.388-400

Д. Е. Пальчунов, д-р физ.-мат. наук, академик РИА, вед. науч. сотр., palch@math.nsc.ru,
Институт математики СО РАН им. С. Л. Соболева, Новосибирск,
С. И. Чернявцева, студент, s.chernyavtseva@g.nsu.ru,
Новосибирский государственный университет

Разработка интеллектуального помощника для автоматизированного порождения документов кафедры*

Поступила в редакцию 13.06.2023
Принята к публикации 27.06.2023

Статья посвящена разработке формальных математических методов моделирования бизнес-процессов и применению этих методов для создания интеллектуальных помощников. Для формализации документооборота на кафедре университета применяется теория частичных моделей. Рассмотрена задача создания интеллектуального помощника для автоматизированного порождения документов кафедры. Для решения этой задачи была разработана онтологическая модель предметной области и построены шаблоны документов. В центре внимания находится проблема автоматизированного порождения заведомо корректных документов. Благодаря автоматизированному заполнению документов повышается эффективность и качество изготовленных документов, а также уменьшается время на работу с ними. Возможность гибкого изменения разработанной онтологической модели позволяет оптимизировать работу с документами согласно меняющимсярегламентам.

Ключевые слова: *интеллектуальный помощник, автоматизация документооборота, частичная модель, атомарная диаграмма, гомоморфизмы частичных моделей, онтология предметной области, онтологическая модель, четырехуровневая модель представления знаний*

Для цитирования:

Пальчунов Д. Е., Чернявцева С. И. Разработка интеллектуального помощника для автоматизированного порождения документов кафедры // Программная инженерия. 2023. Том 14, № 8. С. 388—400. DOI: 10.17587/prin.14.388-400.

Введение

В статье представлены результаты исследований, направленных на разработку подхода к автоматизации бизнес-процессов и созданию интеллектуальных помощников, основанного на теоретико-модельных методах извлечения, формального представления и обработки знаний. Рассмотрена задача создания интеллектуального помощника для автоматизации документооборота на кафедре университета.

В настоящее время актуальность автоматизации бизнес-процессов трудно переоценить [1]. Так, автоматизация создания и обработки документов на кафедре в значительной степени упрощает работу секретаря, способствует уменьшению числа ошибок, позволяет ему более эффективно выполнять задачи, в отличие от чисто ручного труда, который не всегда гарантирует положительный результат.

Ручное создание документов может повлечь за собой неправильное их заполнение, поскольку разные люди принимают участие в изготовлении документов, вследствие чего возникают ошибки, опечатки, неточности. Например, могут быть неверно указаны сведения о должностях, научных степенях руководителей, местах прохождения практики, неправильно написано название выпускной квалификационной работы (ВКР). Некор-

* Работа выполнена в рамках государственного задания ИМ СО РАН, проект FWNF-2022-0011.

ректно заполненные документы будут отклонены секретарем государственной экзаменационной комиссии, вследствие чего придется вносить изменения. В связи с этим обстоятельством секретарю кафедры приходится вручную проверять большой объем документов и отслеживать верность чисто рутинных сведений, чтобы избежать ошибок.

Таким образом, возникает необходимость решать задачу порождения документов, являющихся заведомо правильными (с формальной точки зрения). Это, в частности, означает:

- соответствие регламентам, положениям и приказам факультета, вуза, министерства и др.;
- правильность написания имен сотрудников, названий организаций и подразделений;
- правильность указанных должностей и ролей;
- правильность ссылок на приказы, распоряжения и нормативные акты;
- правильность соответствий между конкретными сотрудниками и их ролями (например, научный руководитель, соруководитель и консультант ВКР у данного студента).

В работе рассматриваются вопросы моделирования бизнес-процессов, регламентов и ролей. Данная статья посвящена, в первую очередь, разработке методов моделирования ролей бизнес-процессов и автоматизации документооборота.

При моделировании роли описываются ее функциональные возможности, горизонтальные и вертикальные связи в рамках бизнес-процессов, права доступа к информации и т. д. Информация, которую использует роль, хранится к корпоративной базе знаний. При моделировании и разработке цифровых двойников ролей параллельно разрабатывается постоянно пополняемая и обновляемая база знаний, которую используют цифровые двойники.

В работе, результаты которой приведены в статье, моделируется роль секретаря кафедры университета. Целью этого моделирования является разработка цифрового помощника секретаря кафедры. Рассмотрены функциональные возможности цифрового помощника, связанные с организацией документооборота на кафедре.

Для решения целевой задачи используется теоретико-модельный подход: знания формализуются при помощи формул логики предикатов первого порядка, а документы формально представляются в виде частичных моделей.

1. Обзор инструментальных средств автоматизации бизнес-процессов кафедры

В этом разделе рассмотрим программные средства, позволяющие автоматизировать работу секретаря кафедры, в частности, предоставляющие возможность автоматически порождать документы. Существующие программные средства можно условно разделить на три вида: АРМ (автоматизированные рабочие места), СЭД (системы электронного документооборота) и интеллектуальные помощники.

АРМ (автоматизированное рабочее место) — комплекс, состоящий из технических и программных средств и предназначенный для автоматизации работы специалиста.

АРМ заведующего кафедрой [2] — это программа, которая позволяет на основе данных кафедры автоматизировать распределение нагрузки среди преподавателей и получать различные отчеты по кафедре. В базе данных системы содержится информация о преподавателях и их учебной нагрузке, о дисциплинах кафедры, группе и количестве занятий.

АРМ преподавателя [3] предназначен для выполнения следующих задач: расчета рейтинга студенческой группы; формирования отчетов по пропущенным занятиям студентов; формирования отчета о выполненной нагрузке; автоматического сведения выполненной нагрузки в электронный индивидуальный план преподавателя. Структура программы состоит из нескольких модулей: журнал (автоматический контроль задолженностей студентов, расчет средних оценок и рейтинга, возможность устанавливать уровень порогового балла за контрольную неделю); индивидуальный план (автоматическое сведение выполненной нагрузки и формирование листа сводных данных); ассистент (надстройка для Microsoft Word и автоматизация процесса создания тестов); контроль знаний (отслеживание прогресса выполнения тестов студентами).

Системы электронного документооборота. Для того чтобы упростить процессы работы с документами, автоматизировать делопроизводство, ускорить процессы и сэкономить время на выполнение рутинных задач, используются СЭД, такие как DocVision, 1С: Документооборот, Directum и др. [4]. Данные системы позволяют осуществлять рассылку уведомлений по почте, проводить атрибутивный поиск документов, задач и поручений, повышать качество документов и создавать их шаблоны.

Интеллектуальные помощники — это своего рода приложения или сервисы, которые в ответ на запрос пользователя способны выполнять различные действия в соответствии с заданными сценариями. Они значительно повышают эффективность и качество выполнения задач в различных

предметных областях, а также снижают время на их выполнение. Одна из целей создания интеллектуальных помощников — автоматизация выполнения рутинных задач.

Интеллектуальная обработка документов для глобальных финансов и казначейства [5]. Данный интеллектуальный помощник самостоятельно извлекает данные из финансовых документов. При этом документы могут быть в различных форматах, языках и содержать сложные табличные данные. Помимо этого, помощник упрощает работу с управлением рисками, денежными средствами и банковской сверкой.

Expert.ai [6] — платформа для работы с языковыми данными и текстами, которая использует машинное обучение и искусственный интеллект для классификации, извлечения и анализа информации. На основе проанализированных документов Expert.ai выделяет основные понятия, факты, которые связаны с темой документа. После этого система может провести анализ, который найдет несоответствия между найденными фактами благодаря графу знаний, который используется для устранения двойного толкования смысла используемых понятий.

При наличии существенных достоинств, перечисленные выше системы имеют ряд перечисленные далее недостатков в контексте задач, решения которых представлены в данной работе.

1. Отсутствует возможность гарантированного порождения заведомо правильных документов, в составлении которых принимают участие люди — участники (роли) бизнес-процессов (например, в нашем случае это могут быть рецензия, отзыв руководителя ВКР и т. д.).

2. Отсутствуют средства нахождения и устранения следующих противоречий:

а) противоречий в имеющихся данных и знаниях: в информации, на основе которой автоматически или автоматизированно заполняются документы;

б) противоречий между документами, когда данные, содержащиеся в одних документах, могут противоречить другим, например, порожденные документы могут противоречить существующим регламентам, изложенным в других документах [7].

3. Отсутствует возможность автоматической подстройки бизнес-процессов и, как следствие, порождаемых документов под изменения в регламентах; в идеале, изменение в регламентах должно автоматически приводить к изменению в порождаемых документах.

Разрабатываемый теоретико-модельный подход направлен на устранение этих недостатков. В данной статье в основном рассмотрены подходы к устранению первого из них путем автоматизированного порождения заведомо правильных документов.

2. Интеллектуальный помощник секретаря кафедры

В рамках разрабатываемого авторами подхода решаются отмеченные выше задачи путем создания интеллектуального помощника секретаря кафедры, при этом с частью функциональных возможностей интеллектуального заместителя секретаря кафедры [8]. Интеллектуальный заместитель, в отличие от интеллектульного помощника, не только помогает секретарю кафедры выполнять некоторые функции, но и может делать вместо него часть работы, а именно:

1) порождать заготовки документов (предзаполненные шаблоны), которые должны быть в дальнейшем заполнены;

2) рассылать студентам информацию о сроках сдачи документов, рассылать заготовки документов и осуществлять сбор заполненных документов;

3) проводить проверку собранных документов на наличие опечаток, выявлять как противоречия с актуальной информацией о должностях, названиях тем, местах работы и пр., так и содержательные противоречия документов с существующими регламентами (например, рецензент работает на той же кафедре, где проходит специализацию студент, что является недопустимым согласно положениям из существующих регламентов).

Достаточно объемная часть работы секретаря кафедры связана с проверкой правильности заполнения документов, относящихся к прохождению студентами практики и защитой ВКР. Эти документы содержат информацию об организациях, в которых проходит практика, о научных руководителях, руководителях практики от университета и организации, в которой проходит практика, о приказах о направлении на практику и т. д. Любая, даже мелкая ошибка или опечатка, сделанная при заполнении этих документов, приводит к необходимости замены документа. Это может повлечь необходимость повторного сбора всех подписей и печатей, что вызывает большую (и бессмысленную) трату времени и усилий. Например, неправильная буква или даже отсутствие запятой в формулировке темы ВКР делает соответствующий документ (индивидуальное задание, отчет, рецензию или отзыв) непригодным и требующим замены.

В целях устранения возможностей появления перечисленных выше неточностей и ошибок се-

кретарю кафедры приходится тратить много времени на предварительную проверку указанных документов, а их суммарное количество за семестр исчисляется сотнями. На кафедре, бизнес-процессы которой нужно автоматизировать, проходят специализацию более 150 бакалавров и магистрантов и работает более 90 сотрудников. При этом каждый документ является достаточно объемным, его требуемая форма содержит много полей и изложена на нескольких страницах [9, 10]. Решением задачи обеспечения корректности документов кафедры является замена проверки готовых документов на изготовление предзаполненного пакета документов для каждого студента. Понятно, что для секретаря кафедры эта работа будет еще более объемной, если ее проделывать вручную. Однако данная работа может быть автоматизирована и передана для исполнения цифровому помощнику секретаря кафедры.

Таким образом, в конечном итоге, цифровой помощник секретаря кафедры будет порождать для каждого студента пакет предзаполненных документов, которые далее окончательно заполняют участники (роли) данного бизнес-процесса: сами студенты, научные руководители, руководители практики, рецензенты. Процесс строится так, чтобы его участники при изготовлении окончательных документов не имели возможность вносить в них сведения, которые могут сделать эти документы недействительными. Для этого, в частности, происходит автоматическая проверка окончательных вариантов документов на отсутствие внесения изменений в те разделы, которые уже были автоматически заполнены интеллектуальным помощником.

Таким образом, решается проблема порождения заведомо формально правильных документов. А за правильность содежательной части — соответствие документов реальности — несут ответственность участники (роли) бизнес-процесса: научные руководители, руководитель практики, рецензенты.

Такой подход отчасти идейно подобен методам порождения заведомо правильных программ из спецификаций (из конструктивных доказательств).

Для реализации этого подхода необходимо решать следующие две задачи:

1) создание и изменение шаблонов документов;

2) обеспечение верной (актуальной) информации в базе знаний; в частности, решение задачи выявления и разрешения противоречий.

Один из проблемных вопросов состоит в том, что вид требуемых документов достаточно часто меняется (изменяются образовательные стандарты, могут меняться компетенции и т. д.). Изменение шаблонов документов вручную является очень трудоемким, кроме того, по невнимательности при этом могут появляться ошибки. Выход из этой ситуации — формальное представление шаблонов документов таким образом, чтобы их можно было автоматически изменять при изменении формализованных регламентов, представленных в базе знаний системы.

Для этого используется теоретико-модельная формализация регламентов и шаблонов документов. Шаблоны формально представляются в виде частичных моделей. В базе знаний системы хранятся атомарные диаграммы этих частичных моделей. Регламенты бизнес-процессов, относящиеся к шаблонам документов, формально представляются в виде бесквантворных предложений или $\forall$-предложений логики предикатов первого порядка. Вследствие изменения регламентов и изменения соответствующих наборов предложений логики предикатов в базе знаний в автоматизированном режиме меняются и шаблоны документов. Это означает, что интеллектуальный помощник секретаря кафедры порождает новый, измененный шаблон, а секретарь кафедры подтверждает это изменение или вносит в него свои корректировки.

Изменение шаблонов автоматически влечет изменение вида предзаполненных документов, порождаемых цифровым помощником. Например, форма оценок в полях документа может поменяться с числовой («5») на словесную («отлично»); могут быть добавлены, удалены или изменены поля с компетенциями и т. д.

3. Математические основы разрабатываемого подхода

В работе применяются теоретико-модельные методы для формального описания бизнес-процессов, регламентов, ролей и документов. Регламенты бизнес-процессов формально представляются в виде бесквантворных предложений или $\forall$-предложений логики предикатов первого порядка.

Заметим, что имеет место двойственность между бесквантворными предложениями, содержащими параметры (константы), и $\forall$-предложениями: можно заменять предложение $\varphi(p)$ на $\forall x\varphi(x)$ и наоборот. Параметры, входящие в формулы (например, параметр p, входящий в $\varphi(p)$), представляются сигнатурными константами. Например, в предложении логики предикатов, описывающем регламент «студент должен проверить ВКР системой Антиплагиат до 31 мая», можно взять новую константу s, чтобы обозначить студента, и использовать формулу $\psi(s)$, а можно студента обозначить переменной x и писать $\forall x\psi(x)$.

Как было указано выше, как шаблоны документов, так и сами документы представляются в виде частичных моделей.

Определение [11]. Кортеж $\mathfrak{A}^p = \langle A;\ P_1, ...,\ P_m,\ c_1, ...,\ c_l \rangle$ назовем частичной моделью в сигнатуре $\sigma = \langle P_1, ...,\ P_m,\ c_1, ..., c_l \rangle$, если значение n-местного предиката P_i на модели $\mathfrak{A}^p$ определено как пара $P_i^{\mathfrak{A}^p} = \left(P_i^+,\ P_i^-\right)$, причем $P_i^+,\ P_i^- \subseteq |\mathfrak{A}^p|^n$ и $P_i^+ \cap P_i^- = \varnothing$.

Тогда для элементов $a_1, ..., a_n \in |\mathfrak{A}^p|$, если $(a_1, ..., a_n) \in P_i^+$, то выполнено $\mathfrak{A}^p \models P_i(a_1, ..., a_n)$, если $(a_1, ..., a_n) \in P_i^-$, то выполнено $\mathfrak{A}^p \models \neg P_i(a_1, ..., a_n)$, а если $(a_1, ..., a_n) \notin \left(P_i^+ \cup P_i^-\right)$, то значение предиката $P_i(a_1, ..., a_n)$ на модели $\mathfrak{A}^p$ не известно (не определено).

Обозначим $K^p(\sigma)$ класс частичных моделей сигнатуры σ.

В нашем случае рассматриваются алгебраические системы сигнатур, содержащих только символы предикатов и констант.

Алгебраические системы чисто предикатной сигнатуры (модели) удобно использовать по следующим причинам:

— если сигнатура содержит только символы предикатов, то:

а) объединение любых двух моделей данной сигнатуры — это модель данной сигнатуры;

б) любое непустое подмножество модели данной сигнатуры определяет модель этой сигнатуры (является ее основным множеством);

— можно взять объединение двух произвольных моделей разных сигнатур, и в результате получить модель объединенной сигнатуры.

Очевидно, эти свойства не выполняются для алгебраических систем, содержащих хотя бы один функциональный сигнатурный символ. Однако эти свойства исключительно полезны:

— для интеграции знаний, содержащихся в разных документах;

— для объединения формальных описаний различных бизнес-процессов, регламентов и ролей.

При этом произвольная алгебраическая система, сигнатура которой содержит символы предикатов, функций и констант, легко сводится к модели чисто предикатной сигнатуры заменой каждой n-местной функции $(n + 1)$-местным предикатом — ее графиком (здесь константы можно рассматривать как 0-местные функции).

Добавление констант к чисто предикатной сигнатуре не создает особых трудностей.

Шаблон документа, как и предзаполненный шаблон, не содержит всей информации; эта информация будет в дальнейшем пополняться. Поэтому шаблоны и частично заполненные документы естественно формализовать не обычными моделями, а частичными.

Информация, которая представлена в частичной модели, описывается атомарной диаграммой этой модели. Определение атомарной диаграммы частичной модели $\mathfrak{A}^p$ аналогично определению атомарной диаграммы обычной модели.

Определение. Атомарной диаграммой частичной модели $\mathfrak{A}^p$ называется множество $AD(\mathfrak{A}^p) = \{\varphi \in S(\sigma_A) | \mathfrak{A}_A^p \models \varphi$ и φ атомарное, либо $\varphi = \neg\psi$ и ψ атомарное$\}$, где $S(\sigma_A)$ — это множество предложений (формул без свободных переменных) сигнатуры σ_A.

Заметим, что если на $\mathfrak{A}^p$ не верно ни $P(a_1, ..., a_n)$, ни $\neg P(a_1, ..., a_n)$, то $P(a_1, ..., a_n)$, $\neg P(a_1, ..., a_n) \notin AD(\mathfrak{A}^p)$.

Для формального описания взаимосвязей между шаблонами и документами используется понятие подмодели частичной модели.

Определение [12]. Пусть $\mathfrak{A}^p \in K^p(\sigma_1)$, $\mathfrak{B}^p \in K^p(\sigma_2)$ и $\sigma_1 \subseteq \sigma_2$. Будем говорить, что частичная модель $\mathfrak{A}^p$ является ***подмоделью*** частичной модели $\mathfrak{B}^p$ и обозначать $\mathfrak{A}^p \leqslant \mathfrak{B}^p$, если $|\mathfrak{A}^p| \subseteq |\mathfrak{B}^p|$ и для любых P^n, $c \in \sigma_1$ и $a_1, ..., a_n \in |\mathfrak{A}^p|$ выполнено:

а) если $\mathfrak{A}^p \models P(a_1, ..., a_n)$, то $\mathfrak{B}^p \models P(a_1, ..., a_n)$;

б) если $\mathfrak{A}^p \models \neg P(a_1, ..., a_n)$, то $\mathfrak{B}^p \models \neg P(a_1, ..., a_n)$;

в) $c^{\mathfrak{A}^p} = c^{\mathfrak{B}^p}$.

Замечание. Рассмотрим частичные модели $\mathfrak{A}^p \in K^p(\sigma_1)$ и $\mathfrak{B}^p \in K^p(\sigma_2)$, где $\sigma_1 \subseteq \sigma_2$. Частичная модель $\mathfrak{A}^p$ является подмоделью частичной модели $\mathfrak{B}^p$ $\left(\mathfrak{A}^p \leqslant \mathfrak{B}^p\right)$ тогда и только тогда, когда $AD(\mathfrak{A}^p) \subseteq AD(\mathfrak{B}^p)$.

Преобразования шаблонов и документов формализуются гомоморфизмами частичных моделей.

Определение [13]. Рассмотрим частичные модели $\mathfrak{A}^p \in K^p(\sigma_1)$ и $\mathfrak{B}^p \in K^p(\sigma_2)$, пусть $\sigma_1 \subseteq \sigma_2$. Отображение $h : |\mathfrak{A}^p| \to |\mathfrak{B}^p|$ называется ***гомоморфизмом*** частичной модели $\mathfrak{A}^p$ в частичную модель $\mathfrak{B}^p$ $\left(h : |\mathfrak{A}^p| \to |\mathfrak{B}^p|\right)$, если для любых P^n, $c \in \sigma_1$ и $a_1, ..., a_n \in |\mathfrak{A}^p|$ выполнено:

а) если $\mathfrak{A}^p \models P(a_1, ..., a_n)$, то

$$\mathfrak{B}^p \models P\left(h(a_1), ..., h(a_n)\right);$$

б) $h\left(c^{\mathfrak{A}^p}\right) = c^{\mathfrak{B}^p}$.

Напомним, что не только шаблоны, но и предзаполненные документы, а также окончательные полностью заполненные документы формально представляются в виде частичных моделей. Это позволяет формализовать отношения между шаблонами, отношения между шаблонами и предзаполненными документами, а также отношения между шаблонами, предзаполненными документами и полностью заполненными документами.

Теоретико-модельная формализация этих отношений проводится при помощи отношений между частичными моделями: подмодель-надмодель и гомоморфизмы частичных моделей. Например, поэтапное заполнение шаблона формализуется как переход от частичной подмодели $\mathfrak{A}^p$ к частичной надмодели $\mathfrak{B}^p$: $\mathfrak{A}^p \leqslant \mathfrak{B}^p$. При этом атомарная диаграмма подмодели $\mathfrak{A}^p$ содержится в атомарной диаграмме надмодели $\mathfrak{B}^p$: $AD(\mathfrak{A}^p) \subseteq AD(\mathfrak{B}^p)$.

Изменение шаблона (и соответствующих предзаполненных и окончательно заполненных документов) формализуется как переход от частичной модели $\mathfrak{B}^p$ к частичной модели $\mathfrak{B}_1^p$ так, что имеется подмодель $\mathfrak{A}^p$ модели $\mathfrak{B}^p$: $\mathfrak{A}^p \leqslant \mathfrak{B}^p$, которая также является и подмоделью модели $\mathfrak{B}_1^p$: $\mathfrak{A}^p \leqslant \mathfrak{B}_1^p$. Подмодель $\mathfrak{A}^p$ формализует ту часть шаблона (предзаполненного или заполненного документа), которая остается неизменной.

В терминах формул — атомарных предложений, такой переход от старого шаблона к новому будет формально описываться следующим образом. Рассмотрим фрагмент атомарной диаграммы $\Delta = AD(\mathfrak{B}^p) \setminus AD(\mathfrak{A}^p)$. Δ — это та часть шаблона, формально — та часть атомарной диаграммы частичной модели $\mathfrak{B}^p$, которая должна измениться. Тогда $\Delta_1 = AD(\mathfrak{B}_1^p) \setminus AD(\mathfrak{A}^p)$ — это та часть атомарной диаграммы частичной модели $\mathfrak{B}_1^p$, которая получилась в результате изменения шаблона. А $AD(\mathfrak{B}^p) \setminus \Delta$ — это та часть атомарной диаграммы частичной модели $\mathfrak{B}^p$, которая осталась неизменной: $AD(\mathfrak{B}^p) \setminus \Delta = AD(\mathfrak{B}_1^p) \setminus \Delta_1 = AD(\mathfrak{A}^p)$.

Таким образом, переход от старого шаблона к новому формально представляется как переход от множества атомарных предложений (фрагмента атомарной диаграммы) Δ к фрагменту атомарной диаграммы Δ_1. Например, в результате изменения регламентов, в шаблоне необходимо удалить некоторые поля с компетенциями и добавить поля с новыми компетенциями. Заметим, что тогда $AD(\mathfrak{B}_1^p) = (AD(\mathfrak{B}^p) \setminus \Delta) \cup \Delta_1$.

Данное формальное представление позволяет автоматизировать переход от устаревших форм документов к новым, соответствующим действующим регламентам.

Использование множеств предложений — фрагментов атомарных диаграмм для формального описания шаблонов и изменений в них дает возможность применять технологии и инструментарий семантической паутины (SemanticWeb). Это, в частности, языки OWL, SWRL, а также ризонеры для OWL [14, 15]. Эти программные средства используются для поиска несоответствий и противоречий между порождаемыми документами (шаблонами, предзаполненными и полностью заполненными документами) и регламентами, а также для поиска противоречий в данных, содержащихся в нашей базе знаний.

4. Разработка интеллектуального помощника

Как отмечалось выше, проверка правильности документов требует огромного объема рутинной работы. Предлагаемое решение этой задачи — изготовление предзаполненных документов (заявление на практику, индивидуальное задание, отчет о практике, отзыв о практике, рецензия, отзыв руководителя ВКР), в которых полностью и правильно заполнена формальная часть, содержащая название ВКР, номера приказов, места работы, должности, степени, звания и другие данные о студентах, руководителях и рецензентах.

Для решения проблемы порождения заведомо правильных документов нужно разделить данные, которые должны быть внесены в шаблон, на два вида:

- то, что однозначно определяется информацией, содержащейся в базе знаний (ФИО студента, его группа, тема ВКР, ФИО, должности, научные степени научного руководителя и рецензента и пр.);
- содержательная часть документа, которая включает мнение научного руководителя или рецензента, оценки, поставленные ими студенту, и т. д.

Работа происходит с тремя видами объектов: шаблоны документов, предзаполненные документы и заполненные документы.

Целью интеллектуального помощника секретаря кафедры является изготовление предзаполненных документов на основе имеющейся базы знаний с использованием шаблонов документов, которые являются универсальными для всех студентов данного потока. При изготовлении предзаполненного документа в шаблон вносится вся та информация, которая является детерминированной (т. е. заранее определенной).

Принципиальная разница между информацией, которая уже внесена в предзаполненный документ, и той информацией, которая будет добавлена при создании окончательного документа, — это то, что первая однозначно определена и уже представлена в базе знаний.

Шаблоны и предзаполненные документы описываются частичными моделями, потому что в них содержится не вся необходимая информация, а лишь часть ее. Например, в предзаполнен-

ном документе отсутствуют оценки компетенций студента. В то же время окончательный документ содержит полную информацию.

Вместе с тем обычная модель может рассматриваться как частный случай частичной модели. Поэтому можно использовать частичные модели для формального представления и итоговых полностью заполненных документов.

Отношение между шаблонами, предзаполненными документами и заполненными документами формализуется отношением подмодель — модель на частичных моделях, как это было описано выше.

После того как интеллектуальный помощник изготовил предзаполненными документ, научному руководителю или рецензенту остается внести в документ только содержательные оценки и обоснование этих оценок. Следует отметить, что именно в этой части не возникает вопросов с правильностью или несоответствием регламентам, поскольку оценки человек пишет исходя из своего мнения, что по определению является корректным. Таким образом, полностью решается проблема контроля за правильностью оформления документов.

Задачу по изготовлению документов кафедры можно разбить на две подзадачи:

1) заполнение шаблонов документов на основе имеющихся источников, представленных в нашей базе знаний (о должностях, студентах, группах, научных руководителях, практиках, ВКР и т. д.);

2) наполнение базы данных, актуализация, проверка данных на соответствие реальности, например, человек был избран на новую должность, а в базе данных содержится старая информация, следовательно, данные необходимо изменить.

Для разработки модели предметной области в работе используются онтологии [16—18]. Применение онтологий дает большие возможности и преимущества при моделировании различных предметных областей [19—22].

В широком смысле онтологией называют спецификацию смысла ключевых понятий предметной области и отношений между ними. Для формального описания предметной области используется онтологическая модель, которая содержит информацию об объектах, событиях, процессах данной предметной области. Вся информация определяется в терминах, заданных онтологией [23].

При разработке онтологической модели предметной области кафедры в работе используется четырехуровневая модель представления знаний [24]:

1) онтология предметной области;

2) общие знания (закономерности), регламенты;

3) прецеденты;

4) вероятностные знания.

Определение [25]. Формальной онтологией предметной области *SD* называется пара $O =< A, \sigma >$, где σ — множество ключевых понятий предметной области; A — множество аналитических предложений, описывающих смысл данных ключевых понятий.

Онтология рассматриваемой предметной области включает в себя такие понятия, как: преподаватель, студент, профессор, практика, руководитель практики, ассистент, старший преподаватель, ВКР, бакалавриат, магистратура и многие другие. Помимо этого, учитываются отношения между понятиями, например, иерархия должностей: <ассистент $\Rightarrow$ старший преподаватель $\Rightarrow$ доцент $\Rightarrow$ $\Rightarrow$ профессор> и пр.

Общие знания — это утверждения, которые должны выполняться для всех прецедентов:

1) верхнеуровневые — регламенты министерства;

2) внутренние — регламенты факультета, университета;

3) знания о должностях, местах работы, ролях конкретных людей.

При помощи первых двух уровней порождается третий — уровень *прецедентов*, в нашем случае это шаблоны, автоматически предзаполненные шаблоны и заполненные документы.

Таким образом, в контексте автоматизации документооборота, знания о том, что преподаватель является научным руководителем студента, — это общие знания, а прецеденты — это те шаблоны и документы, которые изготавливаем.

Для описания модели предметной области в работе используется язык OWL и редактор онтологий Protégé. Рассматриваем классы «Рецензент», «ВКР», «Практика», «Документы», «Учащиеся», «Руководители» и др. и их подклассы (рис. 1 и 2). Использование языка представления онтологий OWL дает возможность автоматической поддержки рассуждений и выявления противоречий при помощи ризонеров для OWL [15, 26].

Глубину представления знаний о бизнес-процессах кафедры в разработанной онтологии можно продемонстрировать с помощью графа отношений между классами, который приведен на рис. 3.

В базе знаний содержатся экземпляры этих классов онтологической модели. Например, для конкретного представителя класса «Учащиеся», приведен перечень свойств, необходимый для формирования предзаполненных документов (рис. 4).

При наполнении базы знаний необходимо проверять данные на возможные несоответствия

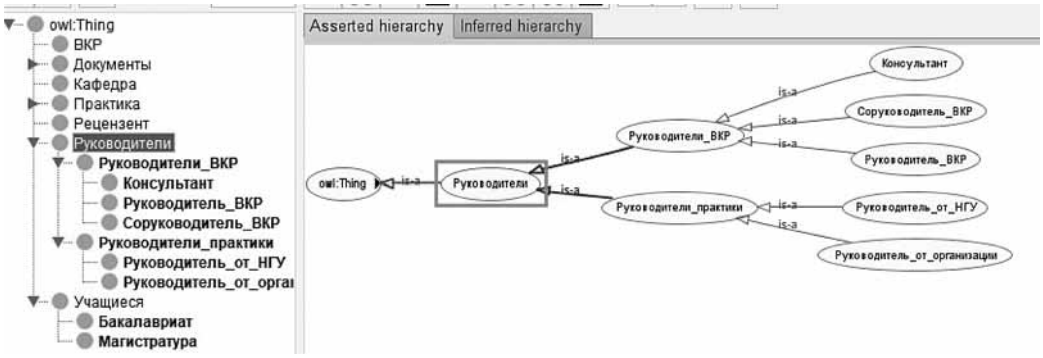


Рис. 1. Представление класса «Руководители»

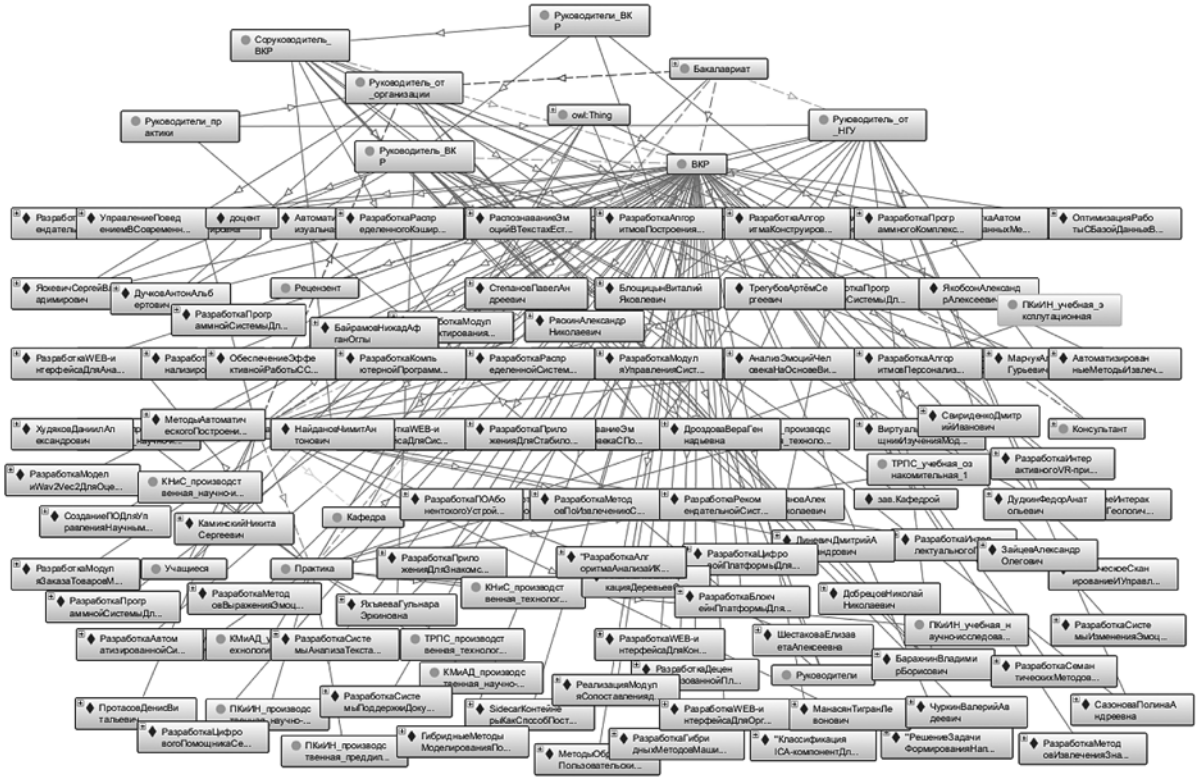


Рис. 2. Фрагмент онтологии предметной области

регламентам. Для этого составлены правила, благодаря которым можно выявлять противоречия. Например, если в онтологической модели обнаружилось, что рецензент студента кафедры работает на этой же кафедре, то это определяется как противоречие (рис. 5).

На рис. 6 приведен пример выявления ризонером несоответствия, которое необходимо разрешить.

5. Описание программной системы SecretaryDoc

В этом разделе приведено описание программной системы SecretaryDoc, реализующей функции интеллектуального помощника секретаря кафедры.

На основании формализованного описания бизнес-процессов интеллектуальному помощнику секретаря кафедры необходимо выполнять следующие задачи:

- создавать шаблоны документов;
- заполнять шаблоны документов;
- наполнять базу знаний и проверять (корректировать) источники информации;
- организовывать рассылку писем (студентам, научным руководителям, рецензентам и др.).

Программная система SecretaryDoc направлена на эффективное решение перечисленных задач. В отличие от конкурентных программных ре-

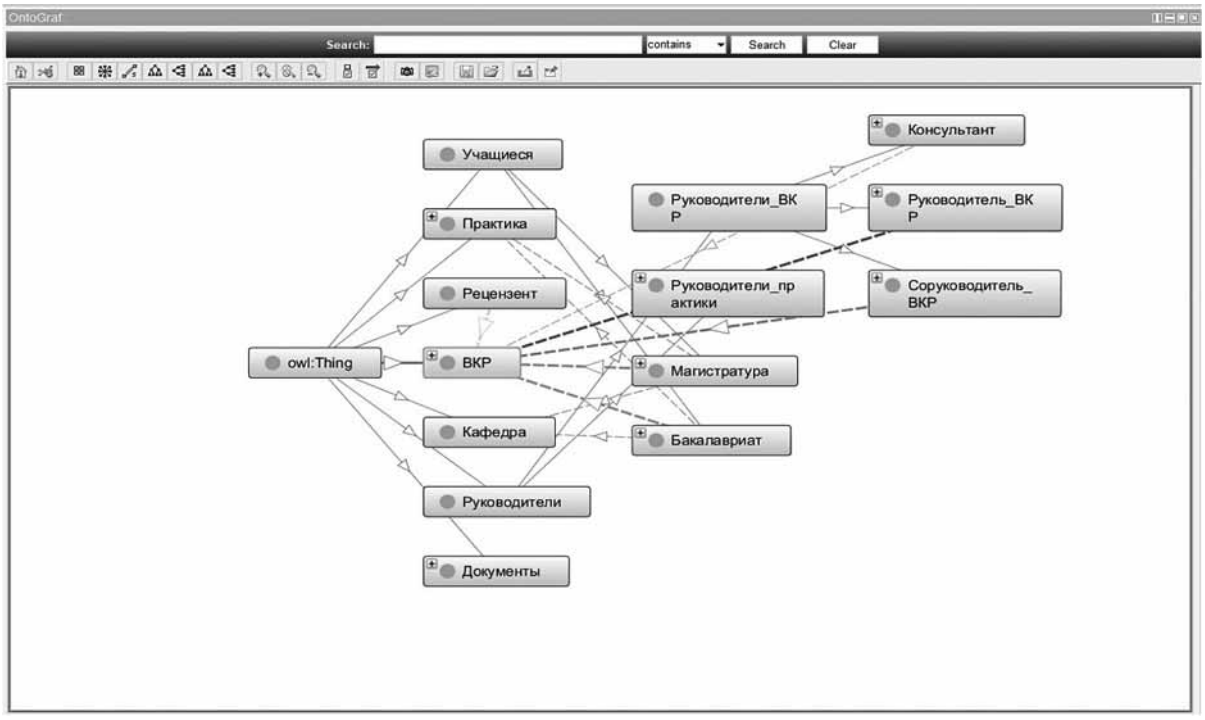


Рис. 3. Фрагмент онтологии предметной области

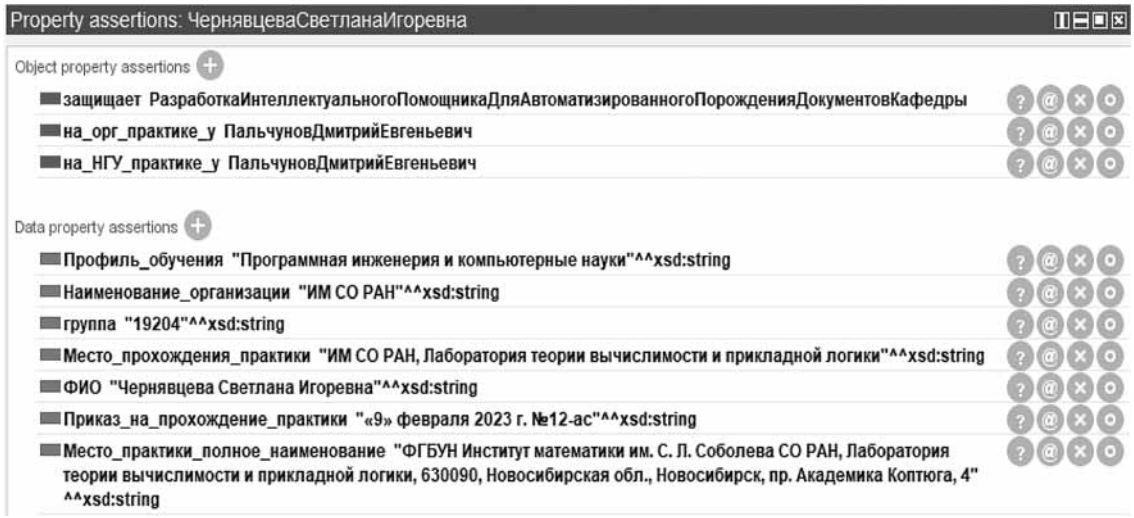


Рис. 4. Информация о представителе класса «Учащиеся»

шений она нацелена не только на создание и заполнение шаблонов, но и на возможность оперативной подстройки системы под изменяющиеся регламенты.

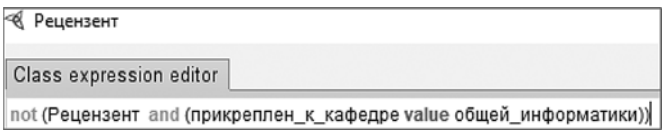


Рис. 5. Пример правила для выявления противоречий

Программный комплекс SecretaryDoc имеет модульную структуру, позволяющую в дальнейшем подключать к нему новые модули. Точкой входа является модуль для работы с интерфейсом программы. Далее происходит взаимодействие с онтологической моделью, а после — заполнение документов.

Модуль взаимодействия с онтологической моделью. Для взаимодействия с базой знаний было выбрано расширение библиотеки Jena — Jena SPARQL API [27]. Данная библиотека позволяет

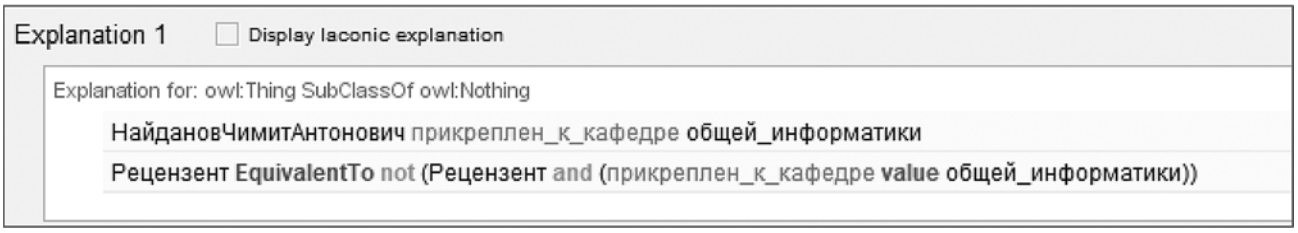


Рис. 6. Нахождение и объяснение имеющегося противоречия

```
PREFIX my: <http://www.semanticweb.org/oleyn/ontologies/2022/4/кафедра#>
SELECT * WHERE {
?студент a my:$degree .
?студент my:защищает ?вкр .
?студент my:ФИО ?фио_студента .
?студент my:группа ?группа_студента .
?студент my:Место_прохождения_практики ?место_практики .
?студент my:Приказ_на_прохождение_практики ?приказ_практика .
?студент my:Место_практики_полное_наименование ?место_практики_полное_наименование .
?студент my:Наименование_организации ?наименование_организации .
?студент my:Профиль_обучения ?профиль .
FILTER regex(?профиль, "$profile")
```

Рис. 7. Пример извлечения данных из модели при помощи SPARQL-запроса

добавлять, изменять, извлекать данные с помощью запросов. Чтобы это сделать, используется SPARQL [28], который является языком запросов к данным в формате RDF (рис. 7). Для того чтобы между ключевыми словами и данными в модели была связь, указывается наименование свойства в OWL для сопоставления данных.

После выполнения запросов модуль преобразует полученную информацию в структуру данных HashMap, далее они передаются в модуль для заполнения шаблонов документов.

Модуль заполнения шаблонов документов. Наиболее важными в настоящий момент являются шесть шаблонов: заявление на практику, индивидуальное задание, отчет о практике, отзыв руководителя практики, отзыв руководителя ВКР и рецензия.

Для работы с документами MS Word используется библиотека Apache POI [29], позволяющая читать и записывать файлы форматов MS Excel и MS Word.

Правильное заполнение документов включает в себя не только извлечение актуальной и верной информации из онтологической модели, но и подстановку правильной формы слова (изменение слова по падежам). Для склонения фамилии, имени и отчества была выбрана библиотека petrovich4j [30]. Данная библиотека на основании отчества позволяет определить пол, и в зависимости от полученного результата для индивида женского пола подставить на месте ключевого слова «обучающаяся», если нужен именительный падеж и «обучающейся» — для родительного (для мужской формы слова «обучающийся» и «обучающегося» соответственно). Выходными данными после работы модуля являются заполненные документы.

Модуль рассылки писем. На основе регламентов определяются сроки, согласно которым нужно разослать студентам письма на электронную почту. Отметим, что функциональная часть интеллектуального помощника секретаря кафедры, отвечающая за рассылку писем, не только включает в себя отправку предзаполненных документов, но и дает возможность уведомить секретаря кафедры и студентов о том, что необходимо предоставить документы на прохождение практики или защиты ВКР.

Используется три вида сообщений с разной степенью срочности исполнения:

- уведомление (о сроках);
- напоминание;
- предупреждение.

Например, уведомление — это сообщение о том, что до сдачи документов остается 14 дней; напоминание— это письмо, в котором говорится, что осталось 3 дня до сдачи документов, а предупреждение — это сообщение о том, остается 1 день до завершения приема документов.

Модуль взаимодействия с программным интерфейсом. Интерфейс построен с использованием языка Java и библиотеки Swing, которая позволяет работать с графическими компонентами. Пользовательский интерфейс предназначен для секретаря кафедры.

Заключение

Рассмотрены подходы к автоматизации бизнес-процессов на кафедре, в частности, автоматизации порождения документов. Проведен анализ существующих программных систем, предназначенных для выполнения подобных задач. Одним из недостатков рассмотренных систем является то, что предложенные решения не позволяют пользователю гибко подстраивать систему под изменяющиеся регламенты, а также порождать гарантированно правильные документы (в составлении которых участвуют различные люди — роли бизнес-процессов). Предложен подход к созданию интеллектуальных помощников, направленных на автоматизацию документооборота на кафедре университета. Данный подход основан на использовании онтологий и теории частичных моделей. Он позволяет гибко модифицировать документооборот при изменении требований и регламентов.

Изложены математические основы предлагаемого подхода к созданию интеллектуальных помощников. Для создания и заполнения иерархических шаблонов используются частичные модели, где одна часть информации уже известна, другая — еще нет, и гомоморфизмы частичных моделей.

Построена онтологическая модель кафедры. На основе онтологической модели реализуется автоматическое заполнение шаблонов документов. Разработана часть функциональных возможностей цифрового заместителя секретаря кафедры, позволяющая создавать шаблоны, заполнять документы и взаимодействовать с администратором (секретарем кафедры).

Список литературы

1. **Свистунов В. М., Лобачев В. В.** Актуальные тренды автоматизации бизнес-процессов в отечественных компаниях // Управление персоналом и интеллектуальными ресурсами в России. 2022. Том 11, № 2. С. 72—76. DOI: 10.12737/2305-7807-2022-11-2-72-76.

2. **Допира Р. И., Кельдибекова А. Б.** Программа «АРМ заведующего кафедрой» // Молодой ученый. 2015. № 11 (91). С. 183-186. URL: https://moluch.ru/archive/91/19434/ (дата обращения 12.06.2023).

3. **АРМ** Преподаватель. URL: https://zifra-plus.ru/product/arm-prep/

4. **Инструменты** для автоматизации документооборота. Лучшие СЭД-системы 2023 года. URL: https://top10-sed.ru (дата обращения 12.06.2023).

5. **Интеллектуальный** помощник для обработки финансовых документов. URL: https://www.emagia.com/products/gia-docs-intelligent-document-processing/ (дата обращения 12.06.2023).

6. **Платформа** для работы с языковыми данными и текстами. URL: https://www.expert.ai/products/expert-ai-platform/?%20 https://www.expert.ai/products/expert-ai-platform/? (дата обращения 12.06.2023).

7. **Lingam V., Bhuria S., Nair M.** et al. Deep learning for conflicting statements detection in text (No. e26589v1). PeerJ Preprints, 2018. DOI: 10.7287/peerj.preprints.26589.

8. **Palchunov D., Vaganova A.** Methods for Developing Digital Twins of Roles Based on Semantic Domain-Specific Languages // 2021 IEEE 22nd International Conference of Young Professionals in Electron Devices and Materials (EDM). 2021. P. 515—519. DOI: 10.1109/EDM52169.2021.9507716.

9. **Шаблоны** для практик. URL: https://www.nsu.ru/n/information-technologies-department/education_fit/praktika/praktika.php (дата обращения 12.06.2023).

10. **Формы** отзыва руководителя ВКР, Формы рецензии. URL: https://www.nsu.ru/n/information-technologies-department/education_fit/gia/docs/ (дата обращения 12.06.2023).

11. **Palchunov D.** Application of FCA for Domain Model Theory Investigation Artificial Intelligence / Eds S. M. Kovalev, S. O. Kuznetsov, A. I. Panov // RCAI 2021. Lecture Notes in Computer Science. Vol. 12948. Springer, Cham., 2021. P. 119—134. DOI: 10.1007/978-3-030-86855-0_9.

12. **Palchunov D. E.** Methodological Aspects of the Application of Model Theory in Knowledge Engineering and Artificial Intelligence // 2022 Ural-Siberian Conference on Computational Technologies in Cognitive Science, Genomics and Biomedicine (CSGB). 2022. P. 210—215. DOI: 10.1109/CSGB56354.2022.9865602.

13. **Palchunov D. E.** Modeling Reasoning and Argumentation for the Development of Intelligent Assistants // 2022 IEEE International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON). 2022. P. 820—825. DOI: 10.1109/SIBIRCON56155.2022.10017050.

14. **OWL** mapping. URL: https://www.w3.org/TR/owl-mapping-to-rdf/

15. **Ризонеры** OWL. URL: http://owl.cs.manchester.ac.uk/tools/list-of-reasoners/ (дата обращения 12.06.2023).

16. **Palchunov D., Yakhyaeva G., Yasinskaya O.** Software system for the diagnosis of the spine diseases using case-based reasoning // Proceedings of the International Conference on Biomedical Engineering and Computational Technologies (SIBIRCON / SibMedInfo). 2015. P. 205—210.

17. **Капустина А. И., Пальчунов Д. Е.** Разработка онтологической модели тарифов и услуг сотовой связи, основанной на логически полных определениях понятий // Вестник НГУ. Серия: Информационные технологии. 2017. Том 15, № 2. С. 34—46.

18. **Олейник А. О.** Разработка интеллектуального помощника секретаря кафедры // Материалы 60-й Международной научной студенческой конференции. Информационные технологии. Программная инженерия и инженерия знаний. 2022. С. 159.

19. **Flanagan K., Stevens R., Pocock M., Lee P., Wipat A.** Ontology for genome comparison and genomic rearrangements // Conference Papers, Comparative and Functional Genomics. 2004. Vol. 5, No. 6—7. P. 537—544, DOI: 10.1002/cfg.436.

20. **Клещев А. С., Москаленко Ф. М., Черняховская М. Ю.** Модель онтологии предметной области «Медицинская

диагностика». Ч.1: Неформальное описание и определение базовых терминов // НТИ. Серия 2. 2005. № 12. С. 1—7.

21. **Клещев А. С., Москаленко Ф. М., Черняховская М. Ю.** Модель онтологии предметной области «Медицинская диагностика». Ч. 2: Формальное описание причинно-следственных связей, причин значений признаков и причин заболеваний // НТИ. Серия 2. 2006. № 2. С. 19—30.

22. **Найданов Ч. А.** Разработка ядра онтологической модели, настраиваемой под предметную область // Вестник НГУ. Серия: Информационные технологии. 2019. Том 17, № 1. С. 72—81. DOI: 10.25205/1818-7900-2019-17-1-72-81.

23. **Пальчунов Д. Е.** Моделирование мышления и формализация рефлексии. I: Теоретико-модельная формализация онтологии и рефлексии // Философия науки. 2006. № 4 (31). С. 62—99.

24. **Naydanov Ch., Palchunov D., Sazonova P.** Development of automated methods for the prevention of risks of critical conditions, based on the analysis of the knowledge extracted from the medical histories // Сибирский научный медицинский журнал. 2016. Том 36, № 1. С. 105—113.

25. **Пальчунов Д. Е.** Решение задачи поиска информации на основе онтологий // Бизнес-информатика. 2008. № 1. С. 3—13.

26. **Трофимов И. В.** Эволюция выразительных способностей языка OWL // Программные системы: теория и приложения: электрон. научн. журнал. 2011. № 4 (8). С. 85—94.

27. **Jena** SPARQL API. URL: https://github.com/SmartDataAnalytics/jena-sparql-api (дата обращения: 12.06.2023).

28. **SPARQL.** URL: https://www.w3.org/TR/rdf-sparql-query/ (дата обращения 12.06.2023).

29. **Apache** POI. URL: https://poi.apache.org (дата обращения: 12.06.2023).

30. **Библиотека** petrovich. URL: https://github.com/damirazo/Petrovich?ysclid=lgubb4xenj46782821 (дата обращения 12.06.2023).

Development of Intelligent Assistant for Automated Generation of Department Documents

D. E. Palchunov, Academician of RAE, Leading Researcher, palch@math.nsc.ru,
S. L. Sobolev Institute of Mathematics SB RAS, Novosibirsk, 630090, Russian Federation,
S. I. Chernyavtseva, Student, s.chernyavtseva@g.nsu.ru,
Novosibirsk State University, Novosibirsk, 630090, Russian Federation

Corresponding author:

Dmitry E. Palchunov, Academician of RAE, Leading Researcher,
S. L. Sobolev Institute of Mathematics SB RAS, Novosibirsk, 630090, Russian Federation
E-mail: palch@math.nsc.ru

Received on June 13, 2023
Accepted on June 27, 2023

The article is devoted to the development of formal mathematical methods for modeling business processes and the application of these methods to create intelligent assistants. The theory of partial models is used to formalize the workflow at the university department. The problem of creating an intelligent assistant for automated generation of university department documents is considered. To solve this problem, an ontological model of the subject area is developed and document templates are built. The focus is on the problem of automated generation of assuredly correct documents. The generation of documents occurs in three stages: a) creation of templates; b) automatic partial filling of documents with fixed information contained in the knowledge base; c) the final meaningful filling of documents by participants in the business process. Relationships between templates, prepopulated documents, and fully populated documents are formally described in terms of homomorphisms of partial models and submodel-supermodel relationships between partial models. Semantic Web technologies are used to search for and eliminate contradictions in the information presented in the knowledge base. Thanks to the automated filling of documents, the efficiency and quality of the produced documents are increased, as well as the time for working with them is reduced. The ability to flexibly change the developed ontological model allows you to optimize the work with documents in accordance with changing regulations

Keywords: *intellectual assistant, workflow automation, partial model, atomic diagram, partial model homomorphisms, domain ontology, ontological model, four-level knowledge representation model*

Acknowledgements: *This work was carried out within the framework of the state contract of the Sobolev Institute of Mathematics (project no. FWNF-2022-0011).*

For citation:

Palchunov D. E., Chernyavtseva S. I. Development of Intelligent Assistant for Automated Generation of Department Documents, *Programmnaya Ingeneria,* 2023, vol. 14, no. 8, pp. 388—400. DOI: 10.17587/prin.14.388-400.

References

1 **Svistunov V. M., Lobachev V. V.** Actual trends of automation of business processes in domestic companies, *Personnel and intellectual resources management in Russia*, 2022, vol. 11, no. 2, pp. 72—76. DOI: 10.12737/2305-7807-2022-11-2-72-76 (in Russian).

2. **Dopira R. I., Keldibekova A. B.** The program "Automated workplace of the head of the department", *Young scientist*, 2015, no. 11, pp. 183—186, available at: https://moluch.ru/archive/91/19434/ (date of access 12.06.2023) (in Russian).

3. **Automated** workplace teacher, available at: https://zifra-plus.ru/product/arm-prep/ (date of access 12.06.2023) (in Russian).

4. **Tools** for document management automation, available at: https://top10-sed.ru (date of access 12.06.2023) (in Russian).

5. **Intelligent** assistant for processing financial documents, available at: https://www.emagia.com/products/gia-docs-intelligent-document-processing/ (date of access 12.06.2023) (in Russian).

6. **Platform** for working with language data and texts, available at: https://www.expert.ai/products/expert-ai-platform/? (date of access 12.06.2023) (in Russian).

7. **Lingam V., Bhuria S., Nair M.** et al. Deep learning for conflicting statements detection in text, no. e26589v1, PeerJ Preprints, 2018, DOI: 10.7287/peerj.preprints.26589.

8. **Palchunov D., Vaganova A.** Methods for Developing Digital Twins of Roles Based on Semantic Domain-Specific Languages, *2021 IEEE 22nd International Conference of Young Professionals in Electron Devices and Materials (EDM)*, 2021, pp. 515—519. DOI: 10.1109/EDM52169.2021.9507716.

9. **Templates** for practices, available at: https://www.nsu.ru/n/information-technologies-department/education_fit/praktika/praktika.php (date of access 12.06.2023) (in Russian).

10. **Forms** of review of the head of the FQW, Review forms, available at: https://www.nsu.ru/n/information-technologies-department/education_fit/gia/docs/ (date of access 12.06.2023) (in Russian).

11. **Palchunov D.,** Application of FCA For Domain Model Theory Investigation», *Artificial Intelligence, Rcai 2021*, *Lecture Notes in Artificial Intelligence*, 12948 / Eds. S. Kovalev, S. Kuznetsov, A. Panov, Springer International Publishing Ag, 2021, pp. 119—134. DOI: 10.1007/978-3-030-86855-0_9.

12. **Palchunov D. E.** Methodological Aspects of the Application of Model Theory in Knowledge Engineering and Artificial Intelligence, *2022 Ural-Siberian Conference on Computational Technologies in Cognitive Science, Genomics and Biomedicine (CSGB),* 2022, pp. 210—215. DOI: 10.1109/CSGB56354.2022.9865602.

13. **Palchunov D. E.** Modeling Reasoning and Argumentation for the Development of Intelligent Assistants, *2022 IEEE International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON)*, 2022, pp. 820—825. DOI: 10.1109/SIBIRCON56155.2022.10017050.

14. **OWL** mapping, available at: https://www.w3.org/TR/owl-mapping-to-rdf/ (date of access 12.06.2023).

15. **Reasoners** OWL, available at: http://owl.cs.manchester.ac.uk/tools/list-of-reasoners/ (date of access 12.06.2023).

16. **Palchunov D., Yakhyaeva G., Yasinskaya O.** Software system for the diagnosis of the spine diseases using case-based reasoning, *Proceedings of the International Conference on Biomedical Engineering and Computational Technologies (SIBIRCON)*, 2015, pp. 205—210.

17. **Kapustina A. I., Palchunov D. E.** Development of an ontological model of tariffs and cellular communication services based on logically complete definitions of concepts, *Bulletin of the NSU. Series: Information Technology*, 2017, vol. 15, no. 2, pp. 34—46 (in Russian).

18. **Oleynik A. O.** Development of an intellectual assistant secretary of the department, *Materials of the 60th International Scientific Student Conference. Information technology. Software Engineering and knowledge engineering*, Novosibirsk, 2022, pp. 159 (in Russian).

19. **Flanagan K., Stevens R., Pocock M., Lee P., Wipat A.** Ontology for genome comparison and genomic rearrangements, *Conference Papers, Comparative and Functional Genomics*, 2004, vol. 5, no. 6-7, pp. 537—544, DOI: 10.1002/cfg.436.

20. **Kleshchev A. S., Moskalenko F. M., Chernyakhovskaya M. Yu**. Model of ontology of the subject area "Medical diagnostics". Part 1: Informal description and definition of basic terms, *NTI, Series 2*, 2005, no. 12, pp. 1—7 (in Russian).

21. **Kleshchev A. C., Moskalenko F. M., Chernyakhovskaya M. Yu.** Model of ontology of the subject area "Medical diagnostics". Part 2: Formal description of cause-and-effect relationships, causes of signs and causes of diseases, *NTI. Series 2*, 2006, no. 2, pp. 19—30 (in Russian).

22. **Naidanov Ch. A.** Development of the core of the ontological model, customizable for the subject area, *Bulletin of the NSU. Series: Information Technology*, 2019, vol. 17, no. 1, pp. 72—81. DOI: 10.25205/1818-7900-2019-17-1-72-81 (in Russian).

23. **Palchunov D. E.** Modeling of thinking and formalization of reflection. I: Model-theoretic formalization of ontology and reflection, *Philosophy of Science*, 2006, no. 4, pp. 62—99 (in Russian).

24. **Naydanov Ch., Palchunov D., Sazonova P.** Development of automated methods for the prevention of risks of critical conditions, based on the analysis of the knowledge extracted from the medical histories, *The Siberian Scientific Medical Journal*, 2016, vol. 36, no. 1, pp. 105—113.

25. **Palchunov D. E.** Solving the problem of information retrieval based on ontologies, *Business Informatics*, 2008, no. 1, pp. 3—13 (in Russian).

26. **Trofimov I. V.** Evolution of the expressive abilities of the OWL language, *Software systems: theory and applications*, 2011, no. 4, pp. 85—94 (in Russian).

27. **Jena** SPARQL API, available at: https://github.com/SmartDataAnalytics/jena-sparql-api (date of access 12.06.2023).

28. **SPARQL**, available at: https://www.w3.org/TR/rdf-sparql-query/ (accessed: 12.06.2023).

29. **Apache** POI, available at: https://poi.apache.org (date of access 12.06.2023).

30. **Library** petrovich, available at: https://github.com/damirazo/Petrovich?ysclid=lgubb4xenj46782821 (date of access 12.06.2023).

УДК 004.021+004.42
DOI: 10.17587/prin.14.401-406

В. И. Макаров, аспирант, ассистент, vadimpsuty@gmail.com,
Поволжский государственный университет телекоммуникаций и информатики, Самара

Оптимизация программной реализации генетического алгоритма с применением параллельных вычислений

Поступила в редакцию 16.06.2022
Принята к публикации 04.07.2023

Дана характеристика генетического алгоритма на примере решения задачи минимизации функции Швефеля и определена возможность его ускорения с применением методов параллелизации. Для возможности оптимизации программы, реализующей алгоритм, дан анализ самых «медлительных» ее участков и расчет приблизительного показателя увеличения производительности. В рамках проводимой оптимизации удалось сократить время реализации алгоритма в 1,63 раза, что соответствует предварительным расчетам.

Ключевые слова: *генетический алгоритм, оптимизация генетического алгоритма, параллелизация вычислений, расчет потенциальной производительности*

Для цитирования:

Макаров В. И. Оптимизация программной реализации генетического алгоритма с применением параллельных вычислений //Программная инженерия. 2023. Том 14, № 8. С. 401—406. DOI: 10.17587/prin.14.401-406.

Введение

Генетический алгоритм[1] (ГА) является эвристическим алгоритмом поиска, используемым для решения задач моделирования путем случайного подбора исходных комбинаций или вариаций определенных моделей по аналогии с принципом естественного отбора. Являясь разновидностью эволюционных вычислений, основанных на принципе рекомбинации исходных данных, ГА обладает существенным недостатком в плане вычислительной производительности. При большом количестве исходных данных и в зависимости от сложности вычисляемой задачи, достижение конечных вычислений в ГА может занимать от нескольких часов до нескольких дней. Если рассматривать вопрос об эффективности использования ГА, то его производительность можно повысить за счет методов оптимизаций. Из существующих подходов оптимизации применительно к производительности ГА можно выделить следующие.

- Использование параллельных вычислений [1—3].
- Комбинирование генетических алгоритмов с другими методами оптимизации, например, выполнение глобального поиска с помощью ГА, а затем использование специализированных методов оптимизации [1, 4].
- Модификация основных операторов ГА [1].
- Модификация всего ГА для повышения эффективности решения конкретной задачи [2, 5].
- Совместное использование ГА и искусственных нейронных сетей [4].
- Выбор оптимальных параметров ГА, таких как размер популяции, число эволюций (эпох), вероятность мутации и скрещивания на основе анализа особенностей решения конкретной задачи [5].

На основании анализа особенностей перечисленных подходов к оптимизации можно выдвинуть предположение о том, что комбинация нескольких из них может существенно повысить эффективность вычислений программной реализации ГА. Таким образом, целью данной работы является оценка повышения производительности ГА, используемого для решения задач оптимизации за счет параллелизации вычислений и предварительного выбора наиболее оптимального размера популяции с обоснованием необходимого количества

[1] Генетический алгоритм. URL: https://ru.wikipedia.org/wiki/Генетический_алгоритм

эволюций. С одной стороны, подобный подход должен ускорить вычисления за счет оптимизации использования всех доступных вычислительных мощностей, а с другой стороны, рационализация выборки из области поиска может позволить сократить время на поиск требуемого решения с заданным уровнем надежности.

Для достижения поставленной цели необходимо решить следующие задачи: определить оптимальный способ параллелизации вычислений; разработать метод определения размера популяции.

Анализ структуры существующего генетического алгоритма

Представленный в исследовании генетический алгоритм проводит решение задачи минимизации многомерной функции Швефеля. Данная функция представляется следующим выражением:

$$F(x) = 418,9829 - \sum_{i=1}^{N} x_i \sin\left(\sqrt{|x_i|}\right). \qquad (1)$$

Выбор данной функции обоснован тем обстоятельством, что она имеет много локальных минимумов и максимумов, однако глобальный минимум у нее только один. Минимум функции Швефеля на интервале $-500 \leqslant x_i \leqslant 500$ расположен в точке x', где все $x_i = 420,9687$ для $i = 1, 2, ..., N$; N — число особей в популяции, а значение функции в этой точке $f(x') = 0$.

Процесс исполнения единичной итерации ГА по набору основных методов состоит в последовательном прохождении этапов вычислений и манипуляций с данными. Поэтапный процесс вычисления в рассматриваемом ГА можно представить следующим набором действий (далее используется термин «методов») в порядке их выполнения:

1) формирование популяции — «популяция»;

2) фиксация события о результате работы алгоритма (в процентном соотношении) — «текущий прогресс»;

3) вычисление средней функции популяции — «средняя ФП»;

4) выборка родителей — «родители»;

5) скрещивание родительских хромосом — «скрещивание»;

6) мутация родительских хромосом — «мутация»;

7) вычисления функции приспособленности потомков — «функция»;

8) формирование новой популяции — «репродукция».

Выборка или селекция родителей определяется из попарных характеристик ограниченного набора родителей. В качестве конечных пар используются следующие значения: лучший со всеми; самый близкий к случайному; самый близкий к лучшему; случайный; самый дальний от случайного; лучший с худшим. Подобная выборка или разбиение на группы позволяет разнообразить набор данных для формирования родительских пар при скрещивании и получения уникальной репродукции.

На каждом из этапов исполнения ГА происходит определение временных характеристик, которые принимаются за время выполнения каждого из методов ГА. Для этого вначале запуска очередного метода запускается таймер. Таймер определяет время (в миллисекундах), затраченное ГА на исполнение каждого из этапов исполнения алгоритма. Завершающим этапом расчета временных затрат является вычисления «эпохи» — параметра T_{epoh}. Под «эпохой» подразумевается временной промежуток, необходимый для исполнения одной отдельно взятой итерации алгоритма.

Для контроля правильности временных замеров было определено условие

$$T_\Sigma = \sum_{i=1}^{8} T_i,\ T_\Sigma \leqslant T_{epoh}, \qquad (2)$$

где T_Σ — суммарное значение времени выполнения всех восьми методов ГА; T_i — время исполнения текущего, завершаемого в текущий момент исполнения ГА, метода; i — индексное обозначение исполняемого в текущий момент времени метода ГА.

Согласно условию (2), временная характеристика исполнения отдельно взятой итерации алгоритма считается верной в случае, если общее время T_{epoh}, затраченное на исполнение одной итерации ГА, соответствует сумме всех временных замеров на каждом отдельно взятом этапе выполнения ГА.

Анализ производительности генетического алгоритма

В рамках проводимого эксперимента по определению времени выполнения алгоритма были определены следующие характеристики запуска ГА: число эволюций — 1200; размер популяции — 190; вероятность скрещивания — 0,92; вероятность мутации — 0,08; вероятность смены популяции при стабилизации функции приспособленности — 0,97; стабилизация проверяется с 300-й эволюции; частота проверки на стабилизацию — 50 эволюций.

По результатам проводимого запуска на каждый этап исполнения ГА согласно условию (2)

получается по 1200 вычислений временных затрат. Усредненные статистические показатели временных затрат на исполнение каждого метода ГА приведены в табл. 1. Дополнительно, на основе полученных данных в табл. 1, были рассчитаны: процентное соотношение доли исполнения эпохи относительно общего времени исполнения итерации; размах R; дисперсия S; среднее квадратическое отклонение σ.

Основываясь на данных, полученных из табл. 1 (выделено полужирным шрифтом), можно сделать вывод о том, что самыми затратными по времени являются методы формирования популяции, выборки родителей и значения функции определения приспособленности потомков. Именно эти методы лучшим образом могут подойти для применения параллелизации вычислений, что впоследствии может ускорить исполнение ГА. Как видно из данных табл. 1, на обозначенные три метода приходится около 97,803 % от общего времени исполнения отдельно взятой итерации ГА.

Исходя из полученных значений и опираясь на формулировку закона Амдала [6], можно получить приблизительное значение увеличения производительности программной ГА с применением распределенной параллелизации вычислений.

Можно получить значение ускорения вычислений за счет организации параллельных вычислений по формуле

$$Speedup \leqslant \frac{1}{(1-pctPar)+\frac{pctPar}{p}}, \qquad (3)$$

где $Speedup$ — ускорение за счет применения параллелизации; $pctPar$ — доля временного интервала, на котором будут выполняться параллельные вычисления (в нашем случае 0,97803); p — число ядер процессора (в нашем случае 2).

Вычисления по формуле (3) дают следующий результат:

$$Speedup \leqslant \frac{1}{(1-0,978)+\frac{0,978}{2}} \Rightarrow Speedup \leqslant 1,957.$$

На основе данных табл. 1, самым затратным по времени выполнения является этап, который выполняет селекцию родителей. Так как внутри метода на этом этапе выполняется набор вложенных функций, обеспечивающих разнообразие скрещиваемых родителей.

Организация параллельных вычислений

Рассматриваемый в рамках исследования ГА реализован на платформе .NET с применением методов классов Task и Parallel [7] для реализации параллельных вычислений в нескольких потоках. Схема организации вычислений выполнена по способу Master—Slave [3]. Здесь в качестве «мастера» (Master) выступает основной цикл ГА, а «подчиненным» (Slave) является метод вычисления функции приспособленности для каждого потомка. Поскольку потомков два, необходимо выделить два потока для вычислений каждого отдельного потомка [8]. В свою очередь, метод формирования популяции разбивается по потокам в зависимости от числа ядер процессора.

Для обеспечения максимальной производительности выбранной реализации учитывается

Таблица 1

Усредненные статистические показатели временных затрат исполнения ГА в разрезе методов

Метод	Имя	Время, мс	Доля, %	max, мс	min, мс	R	S	σ
Популяция	*T1*	**36,336**	19,074	72,640	0,031	272,640	74,333	8,622
Текущий прогресс	*T2*	0,624	0,328	1,247	0,001	0,021	$1 \cdot 10^{-6}$	0,001
Средняя ФП	*T3*	0,742	0,389	1,472	0,012	0,045	$5 \cdot 10^{-6}$	0,002
Родители	*T4*	**147,532**	77,447	294,993	0,072	100,992	1756,000	41,906
Скрещивание	*T5*	1,327	0,697	2,639	0,015	1,639	0,017	0,131
Мутация	*T6*	0,926	0,486	1,849	0,002	0,055	$5 \cdot 10^{-6}$	0,002
Функция	*T7*	**2,442**	1,282	4,873	0,011	4,873	0,035	0,186
Репродукция	*T8*	0,567	0,298	1,060	0,074	0,287	0,000	0,015
Эпоха	T_{epoh}	190,447	97,803	72,640	2,526	270,251	1812,000	42,576

также теорема о зависимости по данным [6] для каждого потока. В итоге общий подход к формированию параллелизации выполняемых потоков выглядит следующим образом:

1) выполняется локализация данных для каждого потока;

2) в каждом потоке происходит формирование действий;

3) запускаются потоки Slave;

4) по завершении всех потоков собираются локализованные данные и передаются в Master (основному ГА) [9].

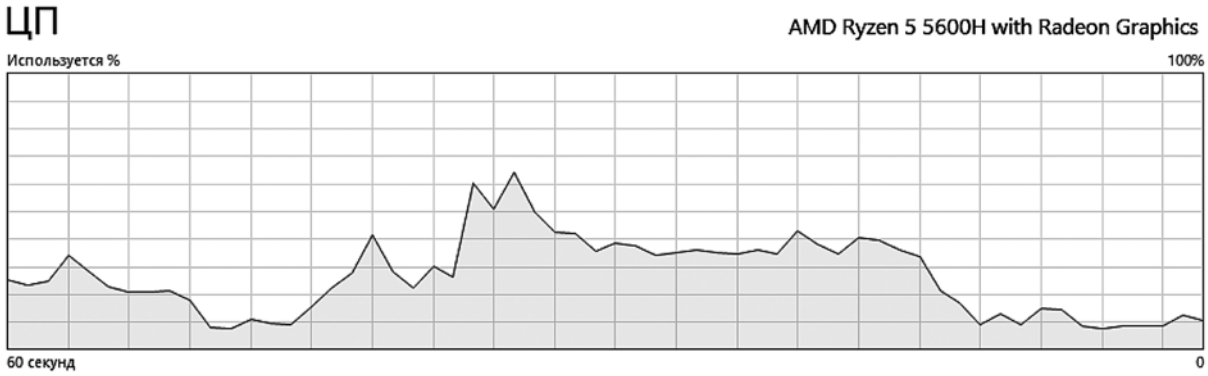
Усредненные результаты временных замеров запуска программной реализации ГА с применением параллельных вычислений приведены в табл. 2.

Из результатов табл. 2 можно заметить существенное сокращение времени исполнения ГА. При условии, что начальные параметры вычислений, производимые программой ГА с применением параллелизации, были аналогичны запуску последовательной

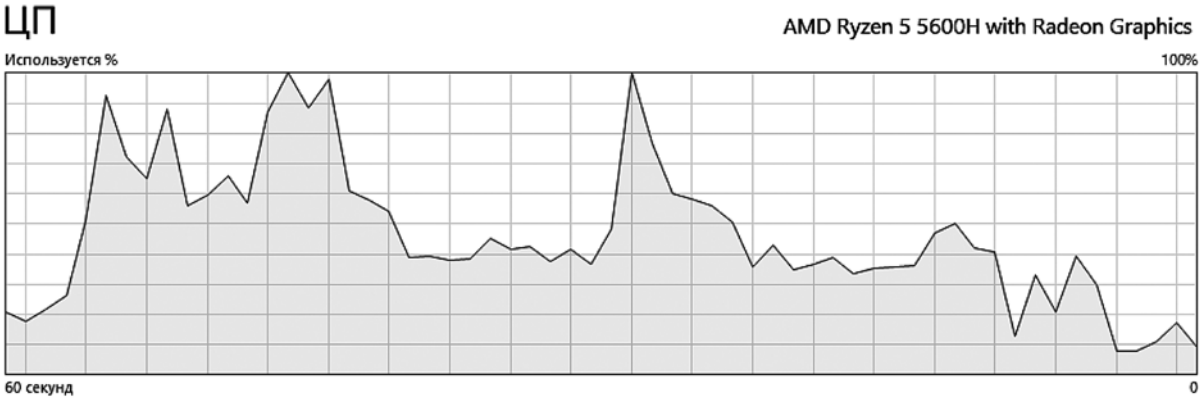
Таблица 2

Усредненные статистические показатели временных затрат на работу программы ГА с применением механизмов параллелизации

Метод	Имя	Время, мс	max, мс	min, мс
Популяция	*T1*	22,292	44,564	0,019
Текущий прогресс	*T2*	0,383	0,765	0,001
Средняя ФП	*T3*	0,455	0,903	0,007
Родители	*T4*	90,510	180,977	0,044
Скрещивание	*T5*	0,814	1,619	0,009
Мутация	*T6*	0,568	1,134	0,001
Функция	*T7*	1,498	2,990	0,007
Репродукция	*T8*	0,348	0,650	0,045
Итого	T_{epoh}	116,839	72,640	2,526



а)



б)

Нагрузка ядер процессора:

а — при последовательном вычислении ГА; *б* — при параллельном вычислении ГА

программной реализации ГА, можно сделать вывод о существенном росте ее производительности.

Как итог оптимизации вычислений производимых ГА, время T_{epoh} сократилось и теперь составляет 116,839 мс. Из отношения полученных данных при последовательном и параллельных вычислениях можно найти коэффициент прироста производительности алгоритма: 190,447/116,839 = 1,63.

Таким образом, решение обратной задачи для формулы Амдала поможет определить долю последовательно исполняемого кода, который удалось оптимизировать, и она равняется 0,77. Однако, если оценить нагрузку на ядра центрального процессора (ЦП), отображенную на рисунке, можно сделать вывод о том, что ускорение выполнения алгоритма реализуется за счет большей нагрузки на аппаратные ресурсы вычислительной машины: около 50 % при последовательном выполнении ГА (см. рисунок, *а*) против использования около 99 % ресурсов ЦП (см. рисунок, *б*) в параллельном режиме. Этот факт является подтверждением эффективности применения параллелизации в качестве решения задачи повышения быстродействия программной реализации ГА.

Заключение

Эффективность применения параллелизации отдельных методов ГА определяется особенностями реализации как алгоритма, так и его функций. Она определяется посредством временных замеров на всех промежуточных этапах выполнения алгоритма. Повышение эффективности при помощи параллелизации вычислений относительно предложенной реализации ГА позволит сократить время расчета и рационально использовать аппаратные средства. Согласно проведенным расчетам, применение оптимизации позволило сократить в 1,63 раза временные затраты на проводимые вычисления. Кроме того, оптимизация программы позволила более полно задействовать доступные аппаратные возможности вычислительной машины.

Следует однако отметить, что в ходе проводимых исследований рассматривалась возможность ускорения программы ГА за счет применения параллельных вычислений без оптимизации параметров запуска самого ГА. Таким образом, при выборе оптимальных «стартовых» значений, можно добиться лучшей производительности программы алгоритма.

Представленные в рамках исследований данные могут быть полезны при разработке и оптимизации генетического алгоритма. При условии, что оптимизированная версия алгоритма с применением параллелизации дает результат, аналогичный последовательно исполняемому ГА. Исходя из усредненных статистических показателей временных затрат исполнения ГА до и после оптимизации, можно исключить вероятность ошибок.

Потенциально, при решении задачи подбора определенного размера требуемой популяции для выполнения ГА существует возможность ускорения работы алгоритма за счет более раннего поиска оптимального решения в части выбора популяции. Данное решение во многом определяется качественным составом самих особей в выбираемой популяции. Таким образом, чем более разнообразным будет материал для скрещивания, тем выше будет вероятность быстрого нахождения оптимума.

Список литературы

1. **Емельянов В. В., Курейчик В. В., Курейчик В. М.** Теория и практика эволюционного моделирования. М.: ФИЗМАТЛИТ, 2003. 432 с.
2. **Кононюк А. Е.** Дискретно-непрерывная математика. (Алгоритмы): в 12 кн. Кн. 10 Алгоритмы. Ч. 3: Генетические алгоритмы, 2017. 444 с.
3. **Серажиев Р. Р., Тынченко В. С.** Исследование параллельных генетических алгоритмов // Актуальные проблемы авиации и космонавтики. 2011. № 1 (7). С. 406—407.
4. **Рутковская Д., Пилиньский М., Рутковский Л.** Нейронные сети, генетические алгоритмы и нечеткие системы. М.: Горячая линия Телеком, 2006. 452 с.
5. **Семенычев Е. В., Куркин Е. И., Данилова А. А.** Выбор параметров генетических алгоритмов в задачах параметрической идентификации нелинейных моделей динамики // Вестник Самарского муниципального института управления. 2013. № 1 (24). С. 130—140.
6. **Карпов В. Е.** Введение в распараллеливание алгоритмов и программ // Компьютерные исследования и моделирование. 2010. Том 2, № 3. С. 231—272.
7. **Генетические** алгоритмы. От теории к практике. URL: https://habr.com/ru/post/138091/ (дата обращения 12.11.2022).
8. **Гергель В. П.** Теория и практика параллельных вычислений. URL: http://www.intuit.ru/department/calculate/paralltp/ (дата обращения 18.10.2022).
9. **Параллельный** генетический алгоритм на основе модели «рабочий-хозяин». URL: https://intuit.ru/studies/courses/14227/1284/lecture/24174?page=2 (дата обращения 12.11.2022).

Optimization of the Genetic Algorithm using Parallel Computing

V. I. Makarov, Postgraduate Student, Teaching Assistant,
Volga Region State University of Telecommunications and Informatics, Samara,
443011, Russian Federation

Corresponding author:

Makarov Vadim I. Postgraduate Student, Teaching assistant,
Volga Region State University of Telecommunications and Informatics, Samara, 443011, Russian Federation
E-mail: vadimpsuty@gmail.com

Received on June 16, 2022
Accepted on July 04, 2023

In the presented work, a characteristic of the genetic algorithm is given on the example of solving the problem of minimizing the Schwefel function and the possibility of its acceleration using parallelization methods is determined. For the possibility of optimizing the program that implements the algorithm, an analysis of its «slowest» sections and the calculation of an approximate indicator of performance increase are given. As part of the ongoing optimization, it was possible to reduce the execution time of the algorithm by 1.63 times, which corresponds to preliminary calculations.

Keywords: *genetic algorithm, genetic algorithm optimization, parallelization of computations, calculation of expected performance*

For citation:

Makarov V. I. Optimization of the Genetic Algorithm using Parallel Computing, *Programmnaya Ingeneria,* 2023, vol. 14, no. 8, pp. 401—406. DOI: 10.17587/prin.14.401-406.

References

1. **Emelyanov V. V., Kureichik V. V., Kureichik V. M.** *Theory and practice of evolutionary modeling.* Moscow: FIZMATLIT, 2003, 432 p. (in Russian).

2. **Kononyuk A. E.** Discrete-continuous mathematics. (*Algorithms*): in 12 books. Book. 10 Algorithms. Part 3: Genetic Algorithms, 2017. 444 p. (in Russian).

3. **Serazhiev R. R., Tynchenko V. S.** Research of parallel genetic algorithms, *Actual problems of aviation and astronautics*, 2011, no. 1 (7), pp. 406—407 (in Russian).

4. **Rutkovskaya D., Pilinsky M., Rutkovsky L.** *Neural networks, genetic algorithms and fuzzy systems*, Moscow: Hotline Telecom, 2006, 452 p. (in Russian).

5. **Semenychev E. V., Kurkin E. I., Danilova A. A.** Choice of parameters of genetic algorithms in problems of parametric identification of nonlinear models of dynamics, *Bulletin of the Samara Municipal Institute of Management,* 2013, no. 1 (24), pp. 130—140.

6. **Karpov V. E.** Introduction to the parallelization of algorithms and programs, *Computer Research and Modeling*, 2010, vol. 2, no. 3, pp. 231—272.

7. **Genetic** algorithms. From theory to practice, available at: https://habr.com/ru/post/138091/ (date of access 11.12.2022).

8. **Gergel V. P.** Theory and practice of parallel computing. available at: http://www.intuit.ru/department/calculate/paralltp/ (date of access 18.10.2022).

9. **Parallel** genetic algorithm based on the "worker-master" model, available at: https://intuit.ru/studies/courses/14227/1284/lecture/24174?page=2 (date of access 11.12.2022).

УДК 004.428.4
DOI: 10.17587/prin.14.407-415

А. И. Хвостов, вед. программист, khvostov@rambler.ru,
ООО «РОК ФЛОУ ДИНАМИКС», Москва,
С. И. Жуков, канд. физ.-мат. наук, руководитель проектов, serge.zhukov@auriga.com,
ООО «Аурига», НИВЦ МГУ, Москва,
А. Ю. Чаускин, канд. техн. наук, ген. директор, a.chauskin@remsystems.ru,
ООО «РЕМ Системс», Москва

Моделирование стохастических свойств конструкционных материалов с помощью пользовательских подпрограмм в SIMULIA Abaqus

Поступила в редакцию 13.02.2023
Принята к публикации 21.06.2023

Описан компонент программного модуля, интегрированный с программным комплексом инженерного анализа SIMULIA Abaqus и предназначенный для моделирования случайных значений параметров материала в конечно-элементной модели на основе заданных статистических характеристик с возможностью учета физической нелинейности поведения материала при различных воздействиях и сочетаниях нагрузок. Целевая группа исследуемых материалов — материалы несущих элементов строительных конструкций, такие как бетон, камень, сталь. Данный программный модуль может быть рекомендован для применения специалистам, инженерам и ученым, занимающимся вероятностным анализом надежности конструкций зданий и сооружений, аппаратов, машин, приборов, при совместном использовании комплексов компьютерного моделирования и инженерного анализа. Программный модуль имеет свидетельство государственной регистрации программы для ЭВМ «АС моделирования стохастических свойств материалов» № 2019667439 от 24.12.2019.

Ключевые слова: *метод конечных элементов, теория надежности, случайные числа, компьютерное моделирование, программирование*

Для цитирования:

Хвостов А. И., Жуков С. И., Чаускин А. Ю. Моделирование стохастических свойств конструкционных материалов с помощью пользовательских подпрограмм в SIMULIA Abaqus // Программная инженерия. 2023. Том 14, № 8. С. 407—415. DOI: 10.17587/prin.14.407-415.

Введение

В настоящее время обеспечение механической безопасности зданий, сооружений и их конструкций регламентируется нормативно-правовыми актами [1—3], основанными на полувероятностном методе предельных состояний с применением коэффициентов надежности. Заложенные в нормы методы расчета не позволяют проектировщику получить количественные показатели надежности разрабатываемого объекта. Например, часто в расчетную модель на наиболее нагруженных участках конструкций добавляются виртуальные дефекты, размер которых назначается, исходя из максимальной разрешающей способности оборудования, применяемого при дефектоскопии или неразрушающем контроле, что позволяет обеспечить требуемую надежность, но количественно завышает закладываемые риски, возникающие при эксплуатации зданий и сооружений. Традиционно используемая качественная характеристика надежности субъективна, существенно зависит от квалификации экспертов и нередко ведет к нецелесообразным экономическим решениям, в том

числе основанным на личной перестраховке специалиста в процессе проектирования.

Вместе с тем имеют место случаи, когда надежность объектов повышенной ответственности оказывается ниже объектов нормального и пониженного уровней. Также следует отметить, что в настоящее время повреждения и дефекты в конструкциях нередко являются следствием ошибок при проектировании. Одним из существенных современных факторов, влияющих на вероятность ошибки, является проблема дефицита времени при проектировании — общепринятые принципы поиска оптимальных решений, основанные на итерационном проектировании и опытной проверке, сейчас становятся все менее актуальными. Одним из эффективных решений отмеченной проблемы является применение универсальных программных комплексов численного моделирования, имеющих гибкие возможности пользовательской настройки, такие как разработка новых типов материала, конечных элементов, настройка решателя и многое другое.

Преобразование Бокса—Мюллера

Обычно алгоритмы генераторов (псевдо)случайных чисел, используемые в программном обеспечении для персональных компьютеров, порождают последовательность чисел, которая подчиняется равномерному распределению. Однако многие физические величины имеют распределение, близкое к нормальному. Поэтому на практике часто возникает задача о преобразовании равномерно распределенной случайной величины в нормально распределенную. Этот вопрос давно и подробно изучен, а наиболее широкое распространение получил метод полярных координат, предложенный Джорджем Боксом, Мервином Мюллером и Джорджем Марсальей в 1958 г. [4]. В настоящей работе авторы воспользовались следующим вариантом преобразования Бокса—Мюллера.

Пусть x и y — независимые случайные величины, равномерно распределенные на отрезке [−1; 1]. Вычислим $s = x^2 + y^2$. Если окажется, что $s > 1$ или $s = 0$, то значения x и y следует перегенерировать заново. Если же выполнится условие $0 < s \leqslant 1$, то далее по формулам следует рассчитать:

$$z_0 = x\sqrt{\frac{-2\ln s}{s}},$$

$$z_1 = y\sqrt{\frac{-2\ln s}{s}}.$$

Полученные z_0 и z_1 являются независимыми величинами, удовлетворяющими стандартному нормальному распределению.

Пример реализации алгоритма метода Бокса—Мюллера на языке FORTRAN:

```
FUNCTION GRAND()
DOUBLE PRECISION R, V1, V2,FAC R = 2.0D0
DO WHILE (R.GE. 1. 0 D0) CALL RANDOM_NUMBER(V1) CALL
RANDOM_NUMBER(V2) V1 = 2.0D0*V1–1.0D0 V2 = 2.0D0*V2–1.0D0
R = V1*V1 + V2*V2
END DO
FAC = SQRT(–2.0D0*LOG(R) /R) GRAND = V2*FAC
RETURN
END FUNCTION
```

Рассмотренный выше метод позволяет получить пару независимых нормально распределенных случайных величин с математическим ожиданием 0 и дисперсией 1. Для того чтобы получить распределение с другими характеристиками достаточно умножить результат функции на среднее квадратическое отклонение и прибавить математическое ожидание.

Таким образом, значение модуля Юнга для изотропного материала с нормально распределенными свойствами вычисляется по формуле:

$$E = M[E] + \sigma[E]z,$$

где z — случайная величина со стандартным нормальным распределением; $M[E]$ — математическое ожидание; $\sigma[E]$ — среднее квадратическое отклонение модуля Юнга материала.

Моделирование материала с накопленными повреждениями

В качестве примера использования модели материала со стохастическими свойствами рассмотрим его применение при расчете железобетонных конструкций, обладающих переменными значениями модуля упругости бетона. Значение модуля упругости в строительных конструкциях является интегральным показателям как для прочностных, так и деформационных характеристик несущих конструкций, и непосредственно связано с обеспечением требований к первой и второй группам предельных состояний.

Так в программном комплексе SIMULIA Abaqus реализована модель разрушения хрупких материалов CDP (*Concrete Damage Plasticity*) [5, 6], в которых снижение модуля упругости выражено через скалярную величину повреждения материала d и начальный модуль Юнга E_0:

$$E = (1 - d)\,E_0,$$

а зависимости между напряжениями и деформациями при одноосном растяжении и сжатии принимают вид

$$\sigma_t = (1 - d_t) E_0 \left(\varepsilon_t - \tilde{\varepsilon}_t^{pl}\right),$$

$$\sigma_c = (1 - d_c) E_0 \left(\varepsilon_c - \tilde{\varepsilon}_c^{pl}\right),$$

где ε_t и ε_c — величины полной относительной деформации при растяжении и сжатии; $\tilde{\varepsilon}_t^{pl}$ и $\tilde{\varepsilon}_c^{pl}$ — величины эквивалентной (эффективной) пластической деформации. Для сложного напряженного состояния эти зависимости Abaqus пересчитывает автоматически.

Диаграмма деформирования бетона, описываемая этой моделью при растяжении и сжатии, представлена на рис. 1.

Близкую к данной реализации формулировку регламентируют и китайские нормы GB 50010-2010 [7]. Например, при определении напряжений

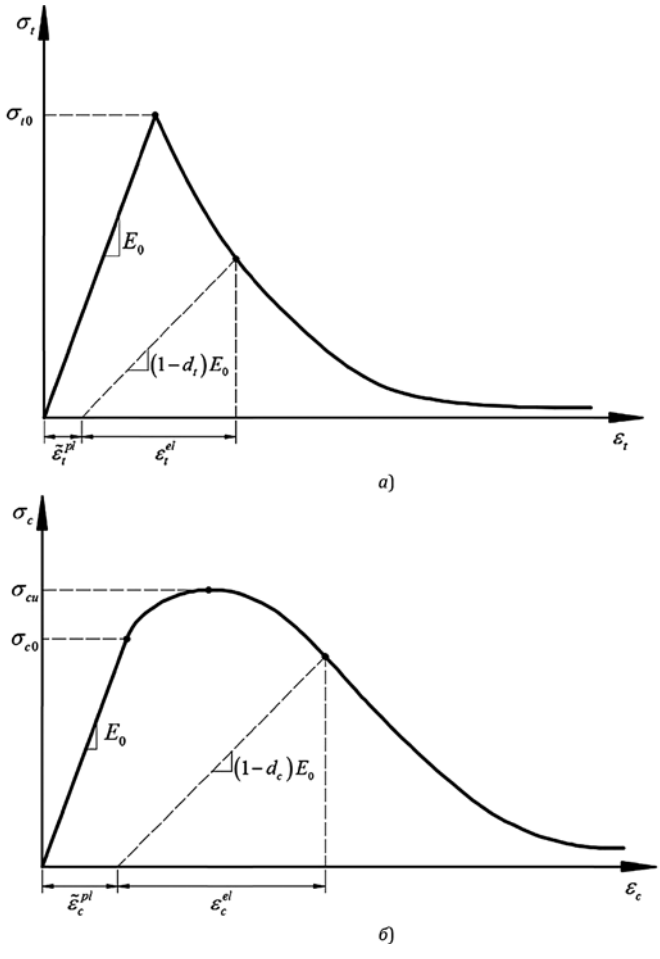


Рис. 1. Диаграммы деформирования бетона, описываемые моделью CDP:

а — при растяжении; *б* — при сжатии (σ_{t0}, σ_{c0} — предел пропорциональности при растяжении/сжатии; σ_{cu} — временный предел прочности при сжатии)

σ при растяжении для бетонов класса С20—С80 плотностью 2200...2400 кг/м^3:

$$\sigma = (1 - d_t) E_t \varepsilon^{el},$$

где ε^{el} — величина соответствующей компоненты упругой деформации при растяжении.

Если рассматривать величину повреждения материала d в качестве случайной величины, распределенной по объему материала, то $M[E]$ и $\sigma[E]$ в заданной точке можно принять равными следующим значениям:

$$M[E] = E_0(1 - M[d]),$$

$$\sigma[E] = E_0 \sigma[d].$$

Также через модуль Юнга выражаются и собственные динамические характеристики объекта, что позволяет говорить об их остаточном ресурсе после длительного времени эксплуатации или после сейсмического воздействия высокой интенсивности, где имеют место остаточные повреждения конструкций. Такие зависимости описаны, например, в работе [8].

Использование пользовательской подпрограммы UMAT

Довольно сложно создать универсальный графический интерфейс, который удовлетворял бы ускоспецифичным требованиям постоянно растущего числа крайне разнообразных задач, решаемых с помощью метода конечных элементов. Для дополнительной настройки работы программного комплекса Abaqus существуют пользовательские подпрограммы (*user subroutines*). Принцип работы с ними заключается в том, что ядро программного комплекса Abaqus во время процесса построения матрицы жесткости вызывает набор написанных пользователем подпрограмм.

Пользовательские подпрограммы — это чрезвычайно мощный инструментарий, который позволяет расширить некоторые из функциональных возможностей Abaqus, на которые обычный интерфейс для ввода исходных данных накладывает слишком строгие ограничения. Пользовательские подпрограммы предоставляют возможность гибко задавать свойства и поведение материалов, начальные и граничные условия и даже напрямую вычислять локальные матрицы жесткости для пользовательских типов конечных элементов.

Пользовательские подпрограммы обычно пишут на языке FORTRAN. Подпрограммы не могут

вызывать друг друга, но из них доступен вызов ряда утилит из программного интерфейса Abaqus.

В Abaqus пользователь может создать свой тип материала. Для задания его поведения создается пользовательская подпрограмма UMAT, которая для запуска из Abaqus должна иметь следующий программный интерфейс:

```
SUBROUTINE UMAT(STRESS,STATEV,DDSDDE, SSE, SPD, SCD,
1 RPL,DDSDDT,DRPLDE,DRPLDT,
2 STRAN,DSTRAN,
TIME,DTIME,TEMP,DTEMP,PREDEF,DPRED,CMNAME,
3 NDI,NSHR,NTENS,NSTATV,PROPS,NPROPS,COORDS,DROT,
PNEWDT,
4 CELENT,DFGRD0,DFGRD1,NOEL,NPT,LAYER, KSPT, JSTEP, KINC)
C
INCLUDE 'ABA_PARAM.INC '
C
CHARACTER*80 CMNAME
DIMENSION STRESS(NTENS),STATEV(NSTATV),
1 DDSDDE(NTENS,NTENS),DDSDDT(NTENS),DRPLDE(NTENS),
2 STRAN(NTENS),DSTRAN(NTENS),TIME(2),PREDEF(1),DPRED(1),
3 PROPS(NPROPS),COORDS(3),DROT(3, 3),DFGRD0(3, 3),
DFGRD1(3, 3),
4 JSTEP(4)

user coding to define DDSDDE, STRESS, STATEV, SSE, SPD, SCD
and, i f necessary, RPL, DDSDDT, DRPLDE, DRPLDT, PNEWDT
RETURN
END
```

Подпрограмма UMAT предназначена для моделирования нелинейного поведения материалов, однако в данной статье ограничимся описанием генерации модели с локально заданными линейно-упругими случайными свойствами. Расчетное ядро Abaqus вызывает ее в каждой Гауссовой точке интегрирования конечного элемента при построении его локальной матрицы жесткости на каждом шаге нагружения.

В подпрограмму передаются перечисленные далее параметры.

- *DDSDDE*(*NTENS, NTENS*) — матрица Якоби, задающая свойства модели материала. *DDSDDE*(*I, J*) задает изменение *i*-й компоненты тензора напряжений в конце шага нагружения, вызванного бесконечно малым приращением *j*-й компоненты тензора деформации. Если не будет активирована опция решения несимметричного уравнения для определяемого пользователем материала, Abaqus будет использовать только симметричную часть *DDSDDE*. Симметричная часть вычисляется как половина суммы матрицы и ее транспонированной матрицы.
- *STRESS*(*NTENS*), этот массив передает в подпрограмму компоненты тензора напряжений в начале итерации численного решения, и в результате выполнения этой подпрограммы его значения должны быть изменены на значения тензора напряжений в конце итерации. В случае, если в модели задано начальное преднагружение, то этот массив будет содержать значения напряжений в начальный момент времени. Размер массива зависит от значения *NTENS*, что будет отмечено далее. При решении задач с учетом конечных деформаций в компонентах тензора напряжений приращение углов поворота за счет движения жесткого целого уже учтено до вызова UMAT. По этой причине в UMAT необходимо вычислять только коротационную часть приращения тензора напряжений. В качестве значений величин напряжений используются значения величины истинных напряжений (тензор напряжений Коши).
- *STATEV*(*NSTATV*) — массив, содержащий значения переменных состояния. В подпрограмму UMAT передаются значения в начале итерации численного решения, если они не были ранее изменены в подпрограммах USDFLD или UEXPAN. В этом случае передаются уже измененные значения. В обоих случаях *STATEV* должен возвращать значения переменных в конце шага нагружения. При решении задач с учетом конечных деформаций любые векторные и тензорные величины из этого массива должны быть преобразованы с учетом поворота модели материала как жесткого целого в добавлении к любым другим изменениям, связанным с поведением материала. Для этой цели в подпрограмму передается матрица *DROT*.
- *SSE, SPD, SCD* — скалярные величины, задающие значения удельной энергии упругой деформации, удельной работы пластической деформации и удельной работы ползучей деформации соответственно. В подпрограмму передаются эти значения в начале шага нагружения и они должны быть изменены на соответствующие значения, вычисленные в конце шага нагружения. Величины удельных энергий не влияют на ход численного решения и используются только для вывода этих значений при обработке результатов.

Подпрограммы USDFLD или UEXPAN используются для вычисления в точке интегрирования конечного элемента значений полей физических величин, заданных пользователем. Переменные *RPL*, *DDSDDT*, *DRPLDE* и *DRPLDT* используются только для решения связанных задач термомеханики. В подпрограмму UMAT передаются также перечисленные далее параметры.

- *STRAN*(*NTENS*) — массив, содержащий компоненты тензора полной деформации в начале итерации численного решения. Если в расчете

учитывается тепловое расширение, то в UMAT передаются только механические деформации (т. е. тепловые деформации, вычисленные с помощью коэффициента теплового расширения, были вычтены из полной деформации). Эти деформации доступны как упругие деформации при выводе результатов. При решении задач с учетом конечных деформаций в компонентах тензора деформаций приращение углов поворота за счет движения жесткого целого уже учтено до вызова UMAT и их значения являются аппроксимацией логарифмических деформаций.

- *DSTRAN*(*NTENS*) — массив приращений деформаций. Если в расчете учитывается тепловое расширение, то это приращения механических деформаций (т. е. полная деформация минус тепловая деформация).
- *TIME*(1) — величина шага по времени в начале текущей итерации.
- *TIME*(2) — абсолютная величина времени от начала расчета до текущей итерации.
- *DTIME* — шаг по времени.
- *TEMP* — температура в начале итерации.
- *DTEMP* — шаг по температуре.
- *PREDEF* — массив — интерполяция значений предзаданного поля величин в каждой точке в начале каждой итерации численного решения по значениям, заданным в узлах сетки.
- *DPRED* — массив приращений значений предзаданного поля величин.
- *CMNAME* — имя заданного пользователем материала. Название некоторых моделей материалов в Abaqus начинается с префикса «ABQ_», поэтому, чтобы не возникал конфликт имен, не стоит использовать «ABQ_» в начале строки *CMNAME*.
- *NDI* — число компонент нормальных напряжений в данной точке.
- *NSHR* — число компонент «инженерных» сдвиговых напряжений в данной точке.
- *NTENS* — число компонент напряжений или деформаций в массиве (*NDI* + *NSHR*).
- *NSTATV* — число переменных состояния, определяющих поведение этого типа материала.
- *PROPS*(*NPROPS*) — массив физико-механических констант, определяющих поведение заданного пользователем материала.
- *NPROPS* — число физико-механических констант, определяющих поведение заданного пользователем материала.
- *COORDS* — массив, содержащий координаты точки интегрирования конечного элемента. Если расчет учитывает геометрическую нелинейность, то массив содержит текущие координаты точки. В противном случае — исходные координаты.
- *DROT*(3,3) — приращение матрицы поворота. Матрица представляет собой приращение матрицы углов поворота системы координат, в которой вычислены компоненты тензоров напряжений (*STRESS*) и деформаций (*STRAIN*). Она используется, чтобы значения векторных и тензорных величин переменных состояния можно было преобразовать непосредственно в этой подпрограмме. Эта матрица передается как единичная матрица как для случая малых, так и для случая больших перемещений, если система координат материала в точке интегрирования перемещается вместе с самим материалом (например, в оболочечных элементах или в случае, когда задана локальная система координат).
- *CELENT* — характерный размер элемента. Для элементов первого порядка это длина некоторой линии, пересекающей элемент, для элемента второго порядка — половина длины этой линии. Для балок и ферм вычисляется как характерная длина вдоль оси элемента. Для мембран и оболочек вычисляется как характерный размер базовой поверхности.
- *DFGRD0*(3,3) — массив, содержащий градиент деформаций в начале итерации численного решения. Опция доступна не для всех типов элементов.
- *DFGRD1*(3,3) — массив, содержащий градиент деформаций в конце итерации численного решения. Массив содержит нули, если в решении не учитываются нелинейные геометрические эффекты.
- *NOEL* — номер элемента.
- *NPT* — номер Гауссовой точки интегрирования.
- *LAYER* — номер слоя (для композитных оболочечных и объемных элементов).
- *KSPT* — номер точки интегрирования в текущем слое.
- *KSTEP* — номер шага нагружения.
- *KINC* — номер итерации численного решения.

Подробное описание и полный список аргументов подпрограммы UMAT можно найти в документации к программному комплексу Abaqus [9].

Из передаваемых ядром Abaqus данных в подпрограмму UMAT в рассматриваемом случае понадобятся тензор напряжений $\sigma = STRESS(NTENS)$ и приращение тензора деформаций $\Delta\epsilon = DSTRAN(NTENS)$ на n шаге нагружения. При этом необходимо вернуть обратно матрицу якобиана

$$J = DDSDDE(i,\ j) = \frac{\Delta\sigma_i}{\Delta\epsilon_i} = E(i,\ j) = E_{ijkl},$$

которая в рассматриваемом случае равна тензору упругих констант материала и величины напряжений на следующем (n + 1)-м шаге нагружения:

$$\sigma_{m+1} = \sigma_n J \Delta\epsilon_n.$$

Моделирование железобетонных конструкций со случайным распределением модуля Юнга тяжелого бетона

На основании действующих нормативно-правовых актов, в частности ГОСТ 28570-2019 «Бетоны. Методы определения прочности по образцам, отобранным из конструкций», распределение значений прочностных характеристик бетона при сжатии принимается стремящимся к нормальному закону распределения и описываются такими показателями, как среднее квадратическое отклонение, коэффициент вариации и математическое ожидание. При этом необходимо отметить, что для образцов бетонов с низкими показателями прочности и большим разбросом характеристик применение нормального закона распределения может быть некорректным, таким, например, является образец газобетона, где в значительном диапазоне прочностные характеристики будут принимать отрицательные значения, что противоречит физическим принципам. Представленный далее пример является демонстратором описанных ранее методов и является предпосылкой для развития практических, верифицированных методов моделирования подобного класса задач.

Ниже представлен порядок действий пользователя при использовании стохастических значений параметров материала на примере моделирования элементарной конструкции — расчете пилона или участка стены на сжатие (рис. 2, *а*). В конечно-элементной модели использовались элементы C3D8 (8-узловой шестигранный конечный элемент сплошной среды с линейной функцией формы). Для того чтобы воспользоваться описываемым программным модулем пользователь в графическом интерфейсе Abaqus должен:

1) создать пользовательский материал (User Material) и задать размер передаваемого в подпрограмму вектора переменных состояния материала, в данном случае три (рис. 2, *б*);

2) указать в параметрах пользовательского материала требуемые значения математического ожидания и среднего квадратического отклонения модуля Юнга $M[E]$ и $\sigma[E]$, а также коэффициент Пуассона μ (рис. 2, *в*);

3) в меню запуска задачи на расчет указать путь к программному модулю (рис. 2, *г*).

Для верификационных тестов были построены конечно-элементные модели со случайными значениями модуля Юнга без учета корреляции значений по пространству, что в действительности может являться случайным разбросом низкого качества материала, полученного при производстве строительных работ. Результаты моделирования для одного из тестовых случаев показаны на рис. 3, см. четвертую сторону обложки.

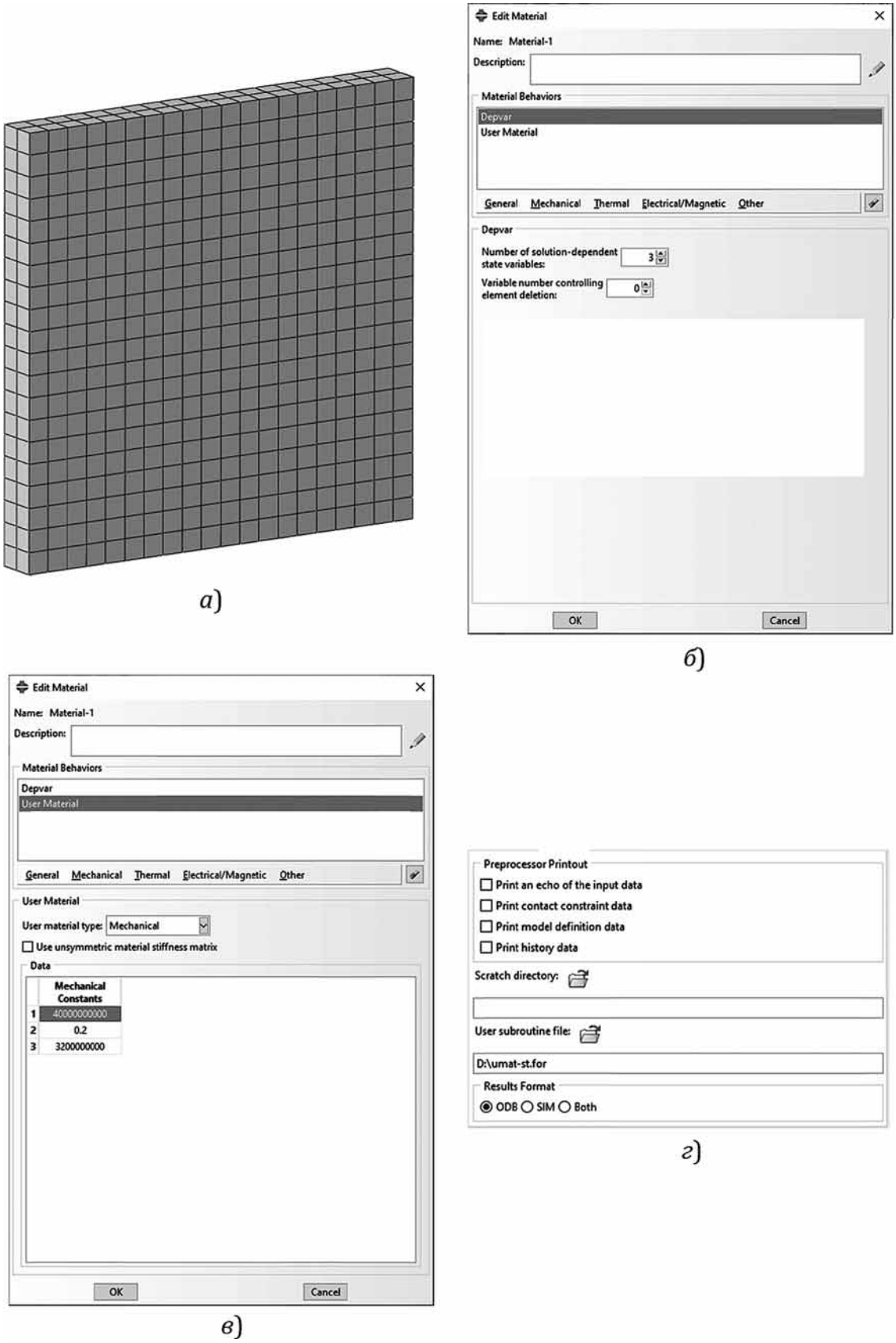
В дальнейшей реализации были выполнены виртуальные испытания в двух видах:

1) для показанной выше элементарной конструкции вида участка стены или пилона, с вычислением расчетных значений вертикальных напряжений для бетона класса B40;

2) для сжатия армированной колонны (класс бетона B25, стали A400), со случайным значением полученного для всего объема модуля Юнга и последующим аналитическим анализом надежности по всей высоте.

В случае 1 после проведения 500 виртуальных испытаний локальные вертикальные внутренние напряжения в бетоне в 498 случаях не превышали допустимых расчетных значений. Результаты расчета показаны на рис. 4, см. четвертую сторону обложки. В двух случаях данные напряжения были превышены, но эти случаи можно идентифицировать как локальные отказы, а не отказ конструкции в целом. Значение надежности составило 99,96 %. При этом индекс надежности составил $\beta \approx 3{,}35$, что является относительно высоким показателем надежности для несущих конструкций, исходя, например, из рекомендательных данных [2] для строительных объектов нормального уровня ответственности с расчетным сроком эксплуатации 50 лет индекс надежности β составляет 3,8.

В случае 2 для одного испытания было получено распределение значения надежности по высоте колонны с учетом возникающих внутренних усилий (рис. 5, см. четвертую сторону обложки). Для данного примера с помощью пользовательских выходных данных для стержневого аналога была получена эпюра надежности по всему конструктивному элементу, что является удобным интегральным показателем оценки механической безопасности.



а) *б)* *в)* *г)*

Рис. 2. Задание стохастических значений параметров пользовательского материала в графическом интерфейсе Abaqus

Заключение

На основании проведенных аналитических и численных исследований с использованием программного модуля генерации случайных значений параметров материала для численных, конечно-элементных моделей, можно сделать следующие выводы.

Разработан программный модуль генерации случайных значений параметров материала, позволяющий проводить численный анализ в стохастической постановке как для строительной отрасли, так и для других технических направлений, что также актуально при разработке новых изделий с применением современных материалов, требующих изучения и соответствующую сертификацию.

Показано практическое применение метода в универсальном комплексе компьютерного моделирования Abaqus: проведена серия виртуальных испытаний с вычислением значения надежности и индекса надежности, применен численно-аналитический метод построения эпюры надежности, позволяющий интегрально в удобном виде оценивать участки конструкций, не удовлетворяющие требованиям прочности, устойчивости и жесткости.

Список литературы

1. **ГОСТ 27751—2014.** Надежность строительных конструкций и оснований. Основные положения. М.: Стандартинформ, 2015. 16 с.
2. **EN 1990:2002+A1:2005** Eurocode — Basis of structural design. Brussels, 2005. 116 p.
3. **ISO 2394:2015.** General principles on reliability for structures. Switzerland, 2015. 111 p.
4. **Кнут Д. Э.** Искусство программирования. Том 2. Получисленные алгоритмы (3-е издание, исправленное и дополненное). М.: Диалектика, 2019. 832 с.
5. **Lee J., Fenves G. L.** Plastic-Damage Model for Cyclic Loading of Concrete Structures // J. Engng. Mech. 1998. Vol. 124, No. 8. P. 892—900. DOI: 10.1061/(ASCE)0733-9399(1998)124:8(892).
6. **Lubliner J., Oliver J., Oller S., Onate E.** A Plastic-Damage Model for Concrete // Int. J. Solids Struct. 1989. Vol. 25, No. 3. P. 229—326.
7. **GB 50010-2010.** Code for design of concrete structures. National standard of People's Republic of China. China architecture & building press. China. 2010. 441 p.
8. **Чаускин А. Ю.** Оценка надежности монолитного железобетонного здания при воздействии максимального расчетного землетрясения: дис. ... канд. техн. наук. М., 2017. 157 с.
9. **Abaqus** User Subroutines Reference Manual. Dassault Systemes. France. 2014. 647 p.

Modeling of Stochastic Material Properties with User Subroutines in SIMULIA Abaqus

A. I. Khvostov, Lead Programmer, khvostoff@rambler.ru,
LLC RFD, Moscow, 117418, Russian Federation,
S. I. Zhukov, Project Manager, serge.zhukov@auriga.com,
Research Computing Center of Moscow State University, Moscow, 119991, Russian Federation
A. Y. Chauskin, CEO, a.chauskin@remsystems.ru,
REM Systems LLC, Moscow, 127083, Russian Federation

Corresponding author:

Alexander I. Khvostov, Lead Programmer,
LLC RFD, Moscow, 117418, Russian Federation
E-mail: khvostoff@rambler.ru

Received on February 13, 2023
Accepted on June 21, 2023

This article describes a software module component integrated with the SIMULIA Abaqus engineering analysis software package and designed to simulate random values of material parameters in a finite element model based on specified statistical characteristics, with the possibility of taking into account the physical nonlinearity of material behavior under various combinations of loads and influences. The target group of materials under study is materials of load-bearing elements of building structures, such as concrete, stone, steel. This software module can be recommended for use by specialists, engineers and scientists engaged in probabilistic analysis of the reliability of structures

of buildings and structures, apparatus, machines, devices, with the combined use of complexes of computer modeling and engineering analysis. Has a certificate of state registration of the computer program «AS for modeling stochastic properties of materials» No. 2019667439 dated 12.24.2019.

Keywords: *Finite element method, theory of reliability, random numbers, computer modeling, programming*

For citation:

Khvostov A. I., Zhukov S. I., Chauskin A. Y. Modeling of Stochastic Material Properties with User Subroutines in SIMULIA Abaqus, *Programmnaya Ingeneria,* 2023, vol. 14, no. 8, pp. 407—415. DOI: 10.17587/prin.14.407-415.

References

1. **GOST 27751—2014.** *Reliability for constructions and foundations. General principles,* Moscow, Standartinform, 2015, 16 p. (in Russian).

2. **EN 1990:2002+A1:2005** Eurocode — Basis of structural design. Brussels, 2005, 116 p.

3. **ISO 2394:2015.** General principles on reliability for structures — Switzerland, 2015. 111 p.

4. **Knuth D. E.** The Art of Computer Programming, vol. 2. Seminumerical Algorithms, 3-ed, Addison-Wesley Professional, 1997.

5. **Lee J., Fenves G. L**. Plastic-Damage Model for Cyclic Loading of Concrete Structures, *J. Engng. Mech*., 1998, vol. 124, no. 8, pp. 892—900. DOI: 10.1061/(ASCE)0733-9399(1998)124:8(892).

6. **Lubliner J., Oliver J., Oller S., Onate E.** A Plastic-Damage Model forConcrete*, Int. J. Solids Struct*., 1989, vol. 25, no. 3, pp. 229—326.

7. **GB 50010-2010.** Code for design of concrete structures. National standard of People's Republic of China. China architecture & building press, China, 2010, 441 p.

8. **Chauskin A. Y.** Evaluation of the reliability of a monolithic reinforced concrete building under the influence of the maximum probable earthquake, Moscow, 2017, 157 p. (in Russian).

9. **Abaqus** User Subroutines Reference Manual. Dassault Systemes. France, 2014, 647 p.

ИНФОРМАЦИЯ

В рамках Международного конгресса «Суперкомпьютерные дни в России» **Нижегородский государственный университет им. Н. И. Лобачевского** проводит международную конференцию и молодежную школу «Математическое моделирование и суперкомпьютерные технологии». Проведение конференции (13—14 ноября) и молодежной школы (15—16 ноября) будет содействовать дальнейшему повышению синергетического эффекта от взаимовыгодного взаимодействия двух приоритетных направлений развития науки и техники — математического моделирования и суперкомпьютерных технологий.

Тематика конференции и молодежной школы охватывает широкий круг задач математического моделирования сложных процессов и численных методов их исследования и проблемы разработки методов суперкомпьютерных вычислений для решения актуальных задач в различных областях науки, промышленности, бизнеса и образования.

Конференция и молодежная школа проводятся под эгидой **Суперкомпьютерного консорциума университетов России** при партнерстве с Национальным центром НТИ **«Технологии машинного обучения и когнитивные технологии»** (СПбГУ ИТМО) и Национальным центром НТИ **«Технологии хранения и анализа больших данных»** (МГУ).

Организаторы: **Научно-образовательный математический центр «Математика технологий будущего»** и **Приволжский научно-образовательный центр суперкомпьютерных технологий** ННГУ им. Н. И. Лобачевского.

Рабочие языки: русский и английский.

Подробности: http://agora.guru.ru/hpc2023

МЕЖДУНАРОДНЫЙ КОНГРЕСС
Суперкомпьютерные дни в России
23 СЕНТЯБРЯ — 30 СЕНТЯБРЯ 2023 г.

Конгресс объединяет научную конференцию, научные школы суперкомпьютерной академии, серию специализированных научных семинаров, экскурсии в ведущие суперкомпьютерные центры и множество других событий, проводимых на различных площадках Москвы и России.

Международная научная конференция

Суперкомпьютерные дни в России

25.09 — 26.09.2023

Конференция проводится в течение двух дней и включает множество параллельных мероприятий: доклады мировых лидеров HPC-индустрии, научные секции, тренинги и семинары, постерную секцию, конференцию молодых ученых, выставку суперкомпьютерных технологий.

Научные школы

Суперкомпьютерная Академия

23.09 — 30.09.2023

Научные школы Суперкомпьютерной Академии — специализированные мероприятия по актуальным направлениям развития науки и технологий.

- Технологии параллельного программирования MPI и OpenMP.
- Высокопроизводительные вычисления на кластерах с использованием графических ускорителей NVIDIA.
- Применение платформы Python для высокопроизводительных вычислений.
- Квантовая информатика.
- Реализация глубоких нейросетей на высокопроизводительных кластерах.

Семинары и мастер-классы

23.09 — 30.09.2023

Темой семинара может быть любая тема, связанная с высокопроизводительными вычислениями, усиливающая или дополняющая тематику конгресса.
Форма семинара определяется его руководителем. Семинары могут проводиться на площадке научной конференции, на одной из площадок конгресса или же в формате телеконференции.
Предложения по проведению семинаров принимаются на конкурсной основе.

Выставка суперкомпьютерных технологий

25.09 — 26.09.2023

На выставке демонстрируются новейшие российские и зарубежные программные и аппаратные решения и технологии для высокопроизводительных вычислений от лидеров рынка HPC.

Подробности: https://congress.russianscdays.org/

ООО "Издательство "Новые технологии". 107076, Москва, ул. Матросская Тишина, д. 23, стр. 2
Технический редактор *Е. В. Конова*. Корректор *А. В. Чугунова*.

Сдано в набор 04.07.2023 г. Подписано в печать 02.08.2023 г. Формат 60×88 1/8. Заказ PI823
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**

Рисунок к статье **Б. А. Позина, А. Е. Гараниной, Е. А. Ивановой**
«АВТОМАТИЗИРОВАННАЯ ОЦЕНКА ПРОГРЕССА И «ЗДОРОВЬЯ» ПРОЕКТА НА ОСНОВЕ СТАНДАРТА OMG ESSENCE» (продолжение)

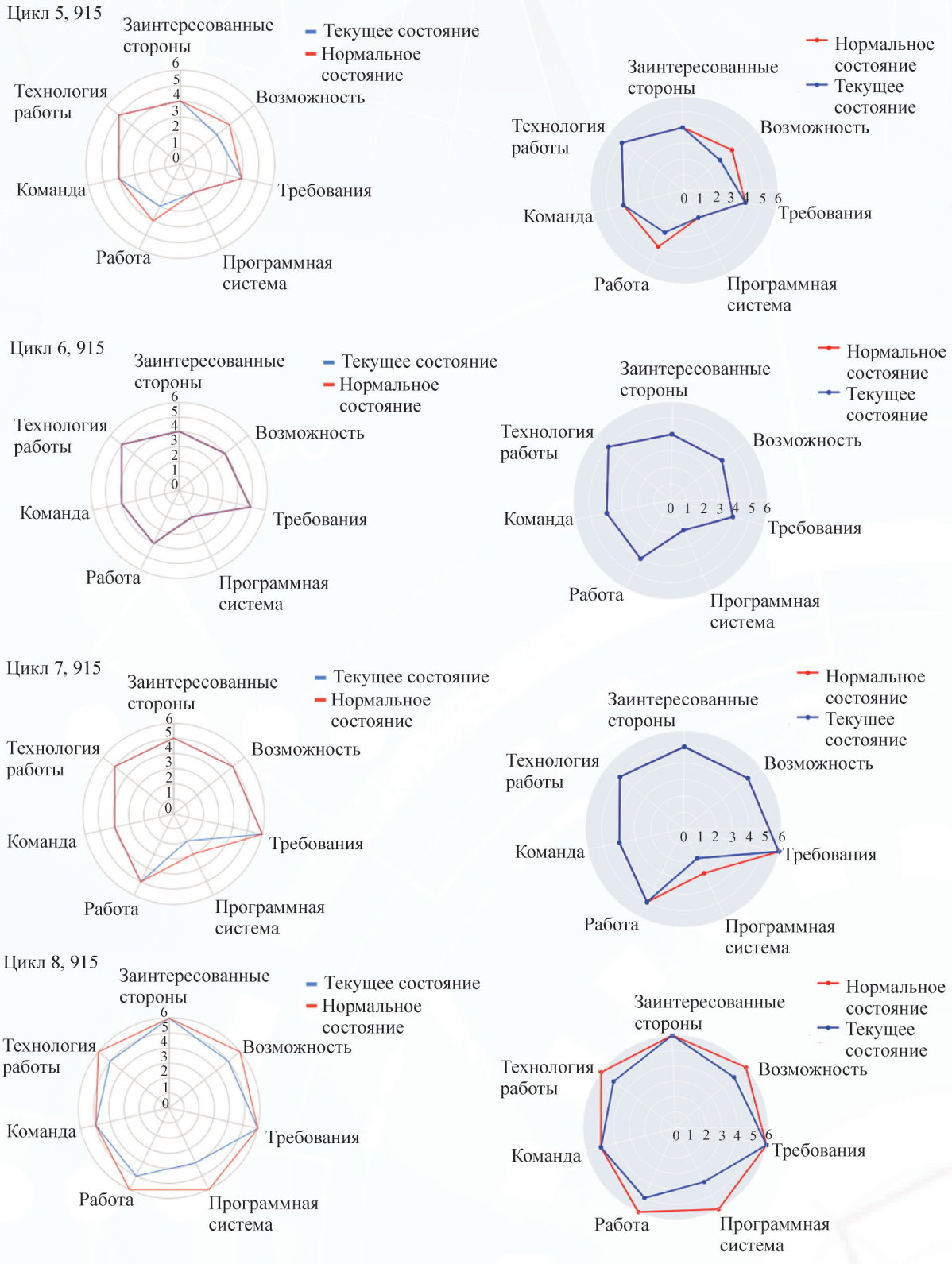


Рис. 3. Лепестковые диаграммы для циклов проекта (продолжение)

Рисунки к статье **А. И. Хвостова, С. И. Жукова, А. Ю. Чаускина**
«МОДЕЛИРОВАНИЕ СТОХАСТИЧЕСКИХ СВОЙСТВ КОНСТРУКЦИОННЫХ МАТЕРИАЛОВ С ПОМОЩЬЮ ПОЛЬЗОВАТЕЛЬСКИХ ПОДПРОГРАММ В SIMULIA ABAQUS»

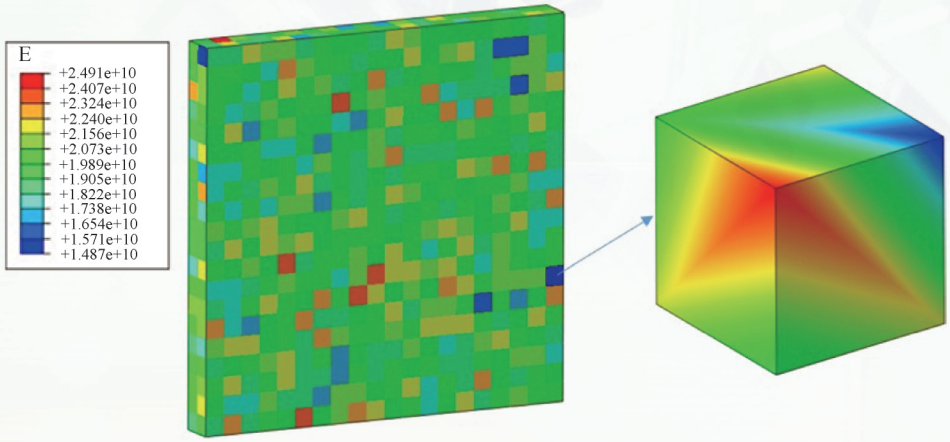
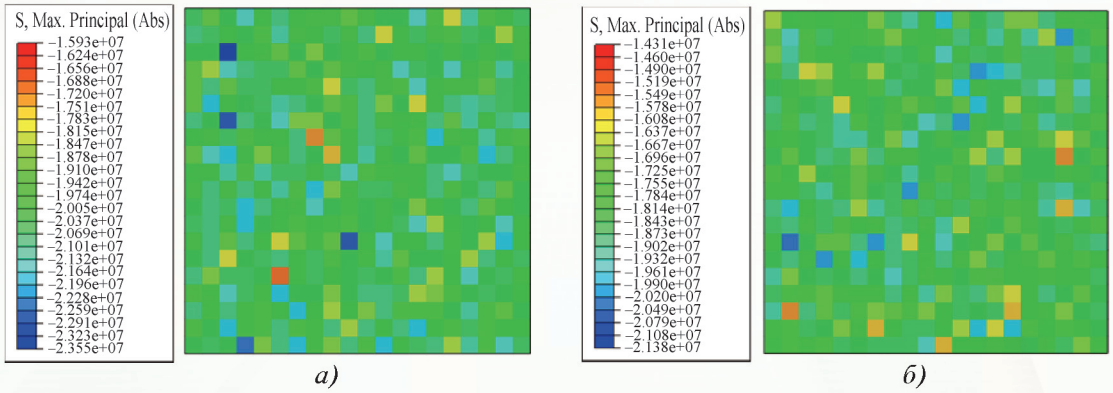


Рис. 3. Мозаика распределения модуля Юнга *E*, Па, для тестовой модели



а) *б)*

Рис. 4. Случайные расчетные реализации мозаики распределения наибольших по модулю главных напряжений:
а – случай отказа (максимальные сжимающие напряжения 23,55 МПа);
б – случай до отказа (максимальные сжимающие напряжения 21,38 МПа)

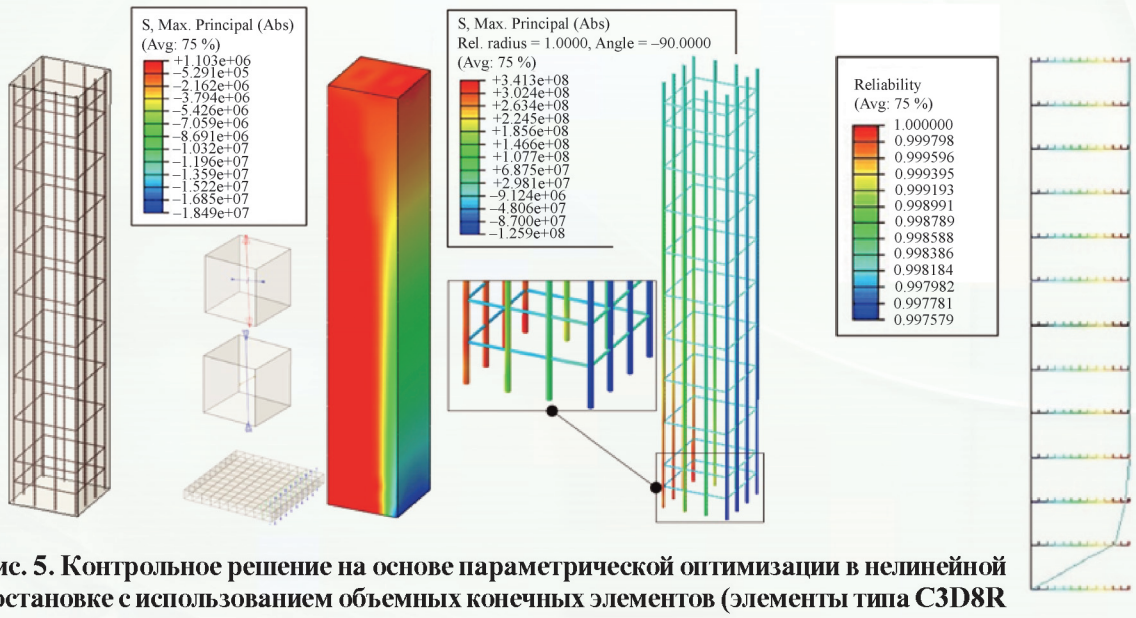


Рис. 5. Контрольное решение на основе параметрической оптимизации в нелинейной постановке с использованием объемных конечных элементов (элементы типа C3D8R и B31R); максимальное напряжение в арматуре 341 МПа, сжимающие напряжения в бетоне 18,5 МПа. Эпюра надежности, распределенная по высоте колонны

ISSN 2220-3397. Программная инженерия. 2023. Том 14. № 8. 365–416. Подписной индекс по Объединенному каталогу «Пресса России» – 22765