ИНФОРМАТИКА, ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И УПРАВЛЕНИЕ

УДК 004.4
doi:10.21685/2072-3059-2021-3-1

Концепция виртуального программиста в самоадаптивной системе управления организацией

А. С. Бождай[1], Ю. И. Евсеева[2], А. А. Гудков[3]

[1,2,3]Пензенский государственный университет, Пенза, Россия

[1]bozhday@yandex.ru, [2]shymoda@mail.ru, [3]alexei-ag@yandex.ru

Аннотация. *Актуальность и цели.* Рассматривается проблема создания специализированного компонента для системы управления организацией, предназначенного для решения задач ее самоадаптации и сопровождения. В число таких задач входят: генерация модификаций программы на основе технического задания, составленного на естественном языке; автоматический перенос исходного кода системы и созданных для нее пользовательских доработок на новые версии языков программирования, библиотек и интерфейсов прикладного программирования; возможности гибкой самоадаптивной подстройки программы под требования различных категорий пользователей. Поскольку все перечисленные функции в настоящее время выполняются программистами вручную, принято решение назвать идею и методы их автоматизации концепцией виртуального программиста. *Материалы и методы.* В качестве основных методов применяются методы интеллектуального распознавания текста и генерации программного кода, основанные на нейросетевых технологиях. *Результаты.* К основным результатам работы относятся: обзор и классификация существующих программных систем управления организацией; концепция виртуального программиста на основе специализированного программного компонента с элементами искусственного интеллекта; описание процесса обучения виртуального программиста автоматической генерации кода на основе технического задания, составленного на естественном языке. *Выводы.* Практическая реализация предложенной концепции позволит снизить ресурсозатраты на модификацию систем управления организацией, оптимизировать жизненный цикл этой системы и повысить качество ее функционирования.

Ключевые слова: самоадаптивное программное обеспечение, программная инженерия, виртуальный программист, нейронные сети, искусственный интеллект, автоматическая генерация кода

Для цитирования: Бождай А. С., Евсеева Ю. И., Гудков А. А. Концепция виртуального программиста в самоадаптивной системе управления организацией // Известия высших учебных заведений. Поволжский регион. Технические науки. 2021. № 3. С. 3–13. doi:10.21685/2072-3059-2021-3-1

The concept of a virtual programmer in a self-adaptive organization management system

A.S. Bozhday[1], Yu.I. Evseeva[2], A.A. Gudkov[3]

[1,2,3]Penza State University, Penza, Russia

[1]bozhday@yandex.ru, [2]shymoda@mail.ru, [3]alexei-ag@yandex.ru

Abstract. *Background*. The article is devoted to the problem of creating a specialized component of an organization's management system, capable of solving the following tasks associated with automated revision and maintenance of such a system: generate program modifications based on a technical specification drawn up in a natural language; automatically transfer the source code of the system and custom modifications created for it to new versions of programming languages, libraries and application programming interfaces; endow the generated software components with the property of self-adaptability - that is, the ability to flexibly adjust to the needs of each category of users. Since the functions that this component should implement are currently performed by programmers, the authors of the article decided to call the concept of its development the concept of a virtual programmer. *Materials and methods*. Methods of intelligent text recognition and program code generation based on neural network technologies are used as the main methods. *Results*. The main results of the work should include: review and classification of existing software management systems of the organization; the concept of a virtual programmer, the main purpose of which is to determine exactly what functions and how a specialized component of artificial intelligence of a software system should be performed, designed for its automatic completion; description of the learning process of a component (virtual programmer) of automatic code generation based on a technical assignment written in a natural language. *Conclusions*. The practical implementation of the concept proposed by the authors will reduce the resource costs for modifying the organization's management systems and improve the quality of their functioning.

Keywords: self-adaptive software, virtual programmer, software engineering, neural networks, artificial Intelligence, automatic code generation

For citation: Bozhday A.S., Evseeva Yu.I., Gudkov A.A. The concept of a virtual programmer in a self-adaptive organization management system. *Izvestiya vysshikh uchebnykh zavedeniy. Povolzhskiy region. Tekhnicheskie nauki = University proceedings. Volga region. Engineering sciences.* 2021;(3):3–13. (In Russ.). doi:10.21685/2072-3059-2021-3-1

Введение

Системы управления организацией широко распространены в настоящее время и используются как небольшими, так и крупными компаниями. В составе современной системы управления организацией можно выделить следующие базовые компоненты.

Подсистема управления производственными процессами. Конкретная техническая реализация данной подсистемы определяется деятельностью компании, но самым распространенным вариантом является система управления взаимоотношениями с клиентами (customer relationship management, CRM). CRM – модель взаимодействия, основанная на приоритетной важности клиента в любых бизнес-процессах. Соответственно главными направлениями деятельности компании являются меры по обеспечению эффективного маркетинга, продаж и обслуживания клиентов. Реализация этой модели включает сбор, хранение и анализ информации о потребителях, поставщиках, партнерах, а также о внутренних процессах компании [1].

Социальная сеть – программно-техническая подсистема, предназначенная для взаимодействия сотрудников в рамках организации. Базовыми компонентами социальной сети являются чаты, группы и новостная лента.

Подсистема модернизации представляет собой внутреннюю программную среду для разработчиков. Направления деятельности и подход к организационным процессам у различных компаний могут различаться между собой столь сильно, что сложно создать систему управления организацией, которая отвечала бы всем требованиям. По этой причине многие современные системы подобного направления имеют ряд встроенных возможностей по дальнейшей модернизации и расширению. К таким возможностям, в частности, относится модернизация с помощью сторонних приложений, использующих специализированный интерфейс прикладного программирования для взаимодействия с системой (REST API), а также написание программных бизнес-процессов.

Прочие подсистемы, в число которых могут входить управление виртуальным хранилищем данных системы, подсистема дистанционного обучения, интеллектуального анализа и визуализации данных.

Наибольший интерес в рамках тематики статьи представляет подсистема модернизации. В настоящий момент ситуация такова, что доработка функциональности системы под нужды компании требует привлечения квалифицированных специалистов в своей области и определенных временных и финансовых затрат. Альтернатива – покупка сторонних приложений. Как правило, пользующиеся большим спросом и коммерчески успешные системы управления организацией имеют свой внутренний интерфейс прикладного программирования, позволяющий программировать расширения для системы в рамках терминов предметной области.

Одно из современных перспективных направлений исследований в области программной инженерии – разработка механизмов автоматической генерации кода – дает возможность более быстрой и менее ресурсозатратной доработки функциональности системы управления организацией. Наибольший интерес представляет возможность создания работающих программных расширений на основе технических заданий, которые могут быть составлены человеком, не являющимся специалистом в области разработки программного обеспечения.

Другой проблемой, часто возникающей как перед создателями систем, так и перед компаниями, использующими их продукцию, является перенос старой функциональности программы под новые версии используемых языков программирования, библиотек и интерфейсов прикладного программирования. В настоящий момент данная проблема решается разработчиками без вспомогательных средств, что также отнимает время и ресурсы.

Самоадаптивность – необходимое свойство многих современных программных систем. Решения, разрабатываемые в качестве новых функций системы, должны быть не просто эффективны, но и иметь возможность гибкой подстройки под нужды пользователей. Ввод в эксплуатацию новой функциональности не всегда сопровождается положительной реакцией пользователей системы, и возможность гибкой самонастройки позволит значительно снизить возможные негативные последствия интеграции нововведений.

1. Обзор существующих систем управления организацией

Существует немало систем управления организацией, включающих в себя различные инструменты для доработки собственной функциональности. Существует три типа расширяемых систем управления организацией: системы с открытым кодом, платные системы с платформами и собственным набором средств разработки, облачные системы с собственным интерфейсом прикладного программирования.

Примером системы с открытым исходным кодом является Sugar CRM [2]. Система разработана на языке PHP, использует в качестве СУБД MySQL или Microsoft SQL Server, содержит свой собственный модуль для визуальной разработки, с помощью которого пользователь, не знакомый с программированием, может создавать новые модули и вносить изменения в старые. Открытый исходный код дает широкие возможности по доработке стандартных функций системы.

Второй группе систем принадлежит, например, Salesforce CRM [3]. Данный программный продукт поставляется заказчикам только по модели облачных вычислений. Для доработки используется собственный Java-подобный язык Apex и собственное средство проектирования Visualforce, предназначенное для генерации пользовательских интерфейсов. Подписчики могут размещать разработанные решения в специализированном каталоге. К преимуществам системы можно отнести простоту средств доработки, сочетаний богатого набора готовых решений и хороших инструментов разработки, высокую модульность и кастомизацию.

Облачные версии с собственным интерфейсом прикладного программирования имеют системы Odoo и Битрикс24. ERP- и CRM-система Odoo [4], в базовую функциональность которой входит бухгалтерский учет, CRM, управление персоналом, производство, продажи, закупки, управление складом, управление проектами, управление транспортом, управление претензиями, POS, интеграция с социальными сетями, написана на языке программирования Python, клиент-серверное взаимодействие реализовано посредством протокола XML-RPC. В качестве системы управления базами данных для серверной части используется PostgreSQL. Также Odoo является одной из самых популярных в мире платформ для бизнес-приложений и имеет собственный онлайн-магазин, в который в настоящий момент загружено около 30 тысяч доработок. Помимо модификации с помощью приложений, возможна также доработка системы с помощью модулей.

Битрикс24 [5] имеет проработанный интерфейс прикладного программирования, а также хорошую документацию по его изучению. Как и в случае с Odoo, имеется возможность создания встраиваемых приложений, а также онлайн-каталог таких приложений. К преимуществам обеих систем можно отнести относительно низкий порог вхождения в разработку, а также широкие возможности по работе с интерфейсом прикладного программирования и гибкость.

Недостатком всех описанных систем является необходимость привлечения специалистов в области программирования для доработки, модификации и сопровождения. Возможное решение данной проблемы следует искать в области автоматизированного программного синтеза.

Синтез программ представляет собой задачу перевода неполной спецификации (например, текста на естественном языке) в программный код,

наиболее этой спецификации удовлетворяющий. В последнее десятилетие сфера программного синтеза значительно продвинулась за счет новых нейросетевых технологий. Однако, несмотря на существующие успехи в данной области, проблема синтеза программного обеспечения, которое могло бы применяться на практике для решения реальных задач, остается открытой. Кроме того, плохо изучена возможность автоматизированного синтеза самоадаптивных программных компонентов.

2. Концепция виртуального программиста

Учитывая перечисленные проблемы и задачи, можно сформулировать концепцию виртуального программиста в системе управления организацией, основывающуюся на следующих принципах:

– генерация исходного кода программных расширений системы на основе технических заданий, составленных на естественном языке;

– включение в функционал сгенерированных расширений возможностей самоадаптации, связанных с самонастройкой параметров и режимов функционирования для различных типов и ролей пользователей;

– возможность автоматического переноса стандартной функциональности системы и разработанных для нее расширений на новые версии языков программирования, программных библиотек и интерфейсов прикладного программирования.

Обучение и развитие виртуального программиста будет включать в себя несколько этапов, среди которых майнинг идиом в репозитории исходного кода и сопоставление выделенных идиом с техническими заданиями, выделение пар «элемент технического задания – идиома», нахождение вариативных элементов в идиомах (переменных).

Под идиомами в языках программирования понимаются часто встречающиеся в различных программах устойчивые синтаксические фрагменты. Каждая идиома имеет собственную, не меняющуюся в зависимости от проекта смысловую роль. Примерами простейших идиом являются код обмена значениями двух переменных, конструкция бесконечного цикла и т.д. Отличительной особенностью идиом в языках, используемых для доработки систем управления организацией, будет их высокоуровневость и предметно-ориентированность, поскольку сами внутренние языки, библиотеки и интерфейсы прикладного программирования таких систем оперируют сущностями более высокого уровня, чем «чистые» языки программирования.

Работа виртуального программиста связана с выполнением следующих основных задач:

1. Генерация исходного кода нового расширения программной системы на основе текста технического задания и составленного в процессе обучения словаря идиом.

2. Генерация пользовательского интерфейса на основе графических макетов.

3. Автоматический перенос имеющегося исходного кода системы и расширений на новые версии используемых языков программирования, библиотек и интерфейсов прикладного программирования.

4. Сбор и анализ мнений сотрудников о разработанном решении, индивидуальная автоматическая настройка решения.

При этом важнейшей задачей в рамках предлагаемой концепции является задача составления словаря идиом. Наибольшую трудность представляет поиск соответствия каждой идиоме необходимой естественно-языковой конструкции. Возможное решение проблемы лежит в области методов искусственного интеллекта, в частности, нейролингвистических методов.

3. Процесс генерации кода по техническому заданию

Анализируя ряд имеющихся работ в области программного синтеза, в частности работу [6], можно дать следующую формулировку задачи программного синтеза.

Предположим, что L – некоторый язык программирования. Тогда каждую программу $P \in L$ можно представить либо последовательностью токенов языка $x_1 \dots x_{|P|}$, либо, что эквивалентно, абстрактным синтаксическим деревом T, построенным в соответствии с контекстно-свободной грамматикой G языка L. Модель программного синтеза $f : \mu \to P$ генерирует программу P на основе спецификации μ так, чтобы условная вероятность $\Pr(P|\mu)$ была максимальной, т.е. программа P являлась наиболее вероятной для заданной спецификации μ. Спецификация представляет собой описание задачи на естественном языке, т.е. последовательность слов $Y = y_1 \dots y_{|Y|}$.

Согласно работе [7] идиомы можно определить как фрагменты корректных абстрактных синтаксических деревьев T в контекстно-свободной грамматике G, т.е. деревья терминальных и нетерминальных символов из G, которые могут быть поддеревьями корректных синтаксических деревьев в G. Грамматика G, расширенная множеством идиом, будет представлять собой грамматику подстановки деревьев. Наиболее интересным и перспективным способом майнинга идиом является способ, основанный на применении вероятностных грамматик подстановки деревьев [7].

Вероятностная контекстно-свободная грамматика позволяет задать распределение строк контекстно-свободного языка. Вероятностная контекстно-свободная грамматика определяется как $G = (E, N, S, R, J)$, где E – множество терминальных символов, N – множество нетерминальных символов, $S \in N$ – корневой нетерминальный символ, R – множество порождающих правил вида $X \to Y$, где $X \in N$ и $Y \in (E \vee N)^*$. Множество J – это множество вероятностных распределений $P(r \mid c)$, где $c \in N$ – нетерминальный символ и $r \in R$ – порождающее правило с нетерминалом c в левой части. Синтаксическое дерево на основе вероятностной контекстно-свободной грамматики начинает строиться с символа S, и каждое раскрытие нетерминального символа c осуществляется в соответствии с распределением $P(r \mid c)$. Вероятность генерации дерева T на основе этой процедуры равна произведению вероятностей правил вывода, участвующих в генерации данного дерева. Вероятность $P(x)$ генерации строки $x \in E^*$ равна сумме вероятностей генерации деревьев T, порождающих x.

Грамматику подстановки деревьев можно представить кортежем $G = (E, N, S, R)$, где E, N, S имеют тот же смысл, что и в определении вероятностной контекстно-свободной грамматики, однако каждое порождающее

правило $r \in R$ принимает форму $X \to \beta_X$, где β_X – фрагмент дерева. Для генерации строки на основе грамматики подстановки деревьев используется то же последовательное рекурсивное расширение, начиная с символа *S*, что и в случае с контекстно-свободной грамматикой, с той разницей, что некоторые правила могут увеличивать высоту дерева более чем на один уровень. Вероятностная грамматика подстановки деревьев расширяет обычную грамматику подстановки деревьев подобно тому, как вероятностная контекстно-свободная грамматика расширяет обычную контекстно-свободную грамматику. Вероятностная грамматика подстановки деревьев определяется как $G = (E, N, S, R, J)$, где J – множество вероятностных распределений $P(\beta_X \mid X)$ для всех $X \in N$, каждый элемент которого является распределением по множеству всех правил $X \to \beta_X$ из множества *R*, имеющих в левой части *X*.

Основной причиной того, что для майнинга идиом используется вероятностная грамматика подстановки деревьев, является то, что каждый фрагмент дерева β_X можно рассматривать как описание набора контекстно-свободных правил, которые обычно используются совместно, т.е. фрагменты β_X потенциально являются идиомами. Для решения задачи нахождения идиом необходимо вывести такую вероятностную грамматику подстановки деревьев, в которой каждый фрагмент синтаксического дерева будет представлять собой идиому в случае, если его глубина больше единицы, или правило исходной грамматики языка, если глубина равна единице.

Далее следует рассмотреть задачу нахождения вероятностной грамматики подстановки деревьев. Ее можно рассматривать как задачу расширения контекстно-свободной грамматики. Входными данными является множество деревьев $T_1 \ldots T_N$ из контекстно-свободной грамматики $G_0 = (E_0, N_0, S_0, R_0)$. Задача расширения контекстно-свободной грамматики заключается в том, чтобы получить вероятностную грамматику подстановки деревьев G_1, расширяющую G_0 и «хорошо объясняющую» обучающее множество $T_1 \ldots T_N$.

Для решения задачи расширения контекстно-свободной грамматики целесообразно использовать методы машинного обучения. В любой задаче машинного обучения важно контролировать сложность модели. Например, многие известные методы кластеризации требуют заранее указывать количество кластеров. Для задачи расширения контекстно-свободной грамматики ключевым фактором, определяющим сложность модели, является число правил вывода фрагментов для каждого нетерминального символа. Если это число будет слишком большим, то модель будет просто запоминать обучающее множество. Если же оно будет слишком малым, то модель не сможет найти полезные паттерны. Эффективно решить данную проблему можно с помощью непараметрических байесовских методов. Подробное описание этих методов можно найти в работах [8, 9].

После того как получено множество идиом $I = \{I_1, \ldots, I_N\}$, следующей задачей, требующей решения, является обучение модели программного синтеза *f*, которая будет выдавать целостные идиомы I_j, а не отдельные узлы абстрактного синтаксического дерева, включающие I_j. Решение данной задачи связано с двумя ключевыми вопросами.

Во-первых, поскольку идиомы представлены как фрагменты абстрактных синтаксических деревьев без конкретного синтаксиса, то программный синтез также будет работать наилучшим образом, если модель *f* будет структурной, т.е. будет генерировать абстрактные синтаксические деревья программ, а не их синтаксис.

Во-вторых, не совсем очевидно, как применить модель *f* к паттернам использования идиом. Один из возможных подходов заключается в расширении исходной грамматики именованными операторами $\mathrm{op}_I\left(l_1,\ldots,l_k\right)$ для каждой идиомы *I*, замене в исходных данных каждого вхождения *I* на op_I и обучении модели синтеза на переписанном наборе данных. Однако это не позволит модели *f* обучаться на определениях (телах) идиом. Кроме того, вхождения идиом часто перекрывают друг друга, и любая подобная стратегия перезаписывания автоматически исключает некоторые идиомы из рассмотрения, тем самым ограничивая применимость модели.

Более эффективный подход по обучению модели для генерации кода основан на выборе наиболее полезного подмножества идиом и наилучшем представлении каждой программы в терминах этих идиом [10]. Это достигается следующим образом: в обучающем множестве (в исходном коде) выполняется маркировка вхождений идиом *I*; в процессе обучения модель выдает либо целиком идиому, либо ее определение (тело) для каждого потенциального вхождения идиомы в абстрактное синтаксическое дерево; в процессе вывода, после генерации идиомы *I*, состояние модели изменяется на состояние, которое она имела бы, если бы генерировала тело идиомы *I* пошагово.

Резюмируя, можно выделить следующие этапы обучения и работы генератора кода в составе модуля виртуального программиста:

1. Извлечение наиболее часто используемых идиом в исходном наборе данных.

2. Все вхождения идиом в исходном наборе данных (программах, использовавшихся при обучении) помечаются как необязательные альтернативные операторы грамматики.

3. Набор данных с отмеченными вхождениями идиом используется для обучения нейросетевой модели.

4. В процессе вывода модель генерирует программы, содержащие имена идиом; код идиом встраивается в программы перед их выполнением.

Следует отметить, что предлагаемая в статье концепция тесно связана с другими нашими работами в направлении создания самоадаптивного программного обеспечения. Особый интерес здесь представляет рассмотрение методов самоадаптации, основанных на интеллектуальном анализе данных и технологии моделирования изменчивости [11], анализе текстовой информации и технологии моделирования изменчивости [12], алгоритмах машинного обучения [13].

Заключение

Предложена концепция автоматизации работы прикладного программиста в задачах, связанных с доработкой самоадаптивных систем управления организацией. Наибольшее внимание уделено методам автоматической генерации кода на основе предложений естественного языка из текста техниче-

ских заданий. Показано, что большими перспективами в разработке таких методов обладает математический аппарат вероятностных контекстно-свободных грамматик. Определены основные направления практической работы в рамках реализации предложенной концепции.

Список литературы

1. Абрамова Е. А., Войнова М. Е. CRM-система как фактор успешной реализации бизнес-процессов в современной компании // Проблемы экономики, финансов и управления производством. 2019. № 44. С. 42–46.
2. Нечаева М. А., Зимина Л. В. Особенности информационных систем управления взаимодействием с клиентами // Экономическая среда. 2016. № 1 (15). С. 59–64.
3. Магомедов Р. М. Анализ возможностей использования платформы Saleforce CRM на российском рынке // Самоуправление. 2021. № 1 (123). С. 312–314.
4. Пашаева С. С., Кочеткова Н. В. Сравнительный анализ отечественных и зарубежных программ, используемых в управлении // Современные научные исследования в сфере экономики : сб. результатов науч. исследований. Киров, 2018. С. 821–824.
5. Наумова О. Г., Елистратова О. В., Герасименко Н. А. Битрикс24 как инновация и удобный сервис для осуществления научной деятельности студентов // Система менеджмента качества: опыт и перспективы. 2016. № 5. С. 262–265.
6. Shin R., Brocksshmidt M., Allamanis M., Polozov O. Program Synthesis with Learned Code Idioms // ICLR 2019 Conference Withdrawn Submission. 2019. № 4. P. 1–12.
7. Allamanis M., Sutton C. Mining idioms from source code // FSE 2014: Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering. Guimaraes, 2014. P. 472–483
8. Gelman A., Carlin J. B. [and etc.]. Bayesian data analysis. London : CRC Press, 2013. 676 p.
9. Gershman S. J., Blei D. M. A tutorial on Bayesian non-parametric models // Journal of Mathematical Psychology. 2012. № 56 (1). P. 1–12.
10. Neelakantan A., Le Q. V., Abadi M. [and etc.]. Learning a natural language interface with neural programmer // Proceedings of the 5th International Conference on Learning Representations (ICLR). 2017. P. 1332–1342.
11. Бождай А. С., Евсеева Ю. И. Метод рефлексивной самоадаптации программных систем // Известия высших учебных заведений. Поволжский регион. Технические науки. 2018. № 2. С. 74–86.
12. Бершадский А. М., Бождай А. С., Евсеева Ю. И., Гудков А. А. Самоадаптация обучающих программных систем на основе наблюдения за информационной средой // Открытое образование. 2019. № 3. С. 33–41.
13. Бождай А. С., Евсеева Ю. И., Артамонов Д. В. Использование машинного обучения с подкреплением в создании самоадаптивного программного обеспечения // Известия высших учебных заведений. Поволжский регион. Технические науки. 2019. № 3. С. 56–68.

References

1. Abramova E.A., Voynova M.E. CRM system as a factor in the successful implementation of business processes in a modern company. *Problemy ekonomiki, finansov i upravleniya proizvodstvom = Issues of economics, finance and production management*. 2019;(44):42–46. (In Russ.)
2. Nechaeva M.A., Zimina L.V. The features of customer relationship management information systems. *Ekonomicheskaya sreda = Economic environment*. 2016;(1):59–64. (In Russ.)

3. Magomedov R.M. Analysis of the possibilities of using the Saleforze CRM platform in the Russian market. *Samoupravlenie = Self-government*. 2021;(1):312–314. (In Russ.)
4. Pashaeva S.S., Kochetkova N.V. Comparative analysis of domestic and foreign programs used in management. *Sovremennye nauchnye issledovaniya v sfere ekonomiki: sb. rezul'tatov nauch. Issledovaniy = Modern scientific research in the field of economics: collection of scientific research's results*. Kirov, 2018:821–824. (In Russ.)
5. Naumova O.G., Elistratova O.V., Gerasimenko N.A. Bitrix24 as an innovation and convenient service for students' scientific activities. *Sistema menedzhmenta kachestva: opyt i perspektivy = Quality management system: experience and perspectives*. 2016;(5):262–265. (In Russ.)
6. Shin R., Brocksshmidt M., Allamanis M., Polozov O. Program Synthesis with Learned Code Idioms. *ICLR 2019 Conference Withdrawn Submission*. 2019;(4):1–12.
7. Allamanis M., Sutton C. Mining idioms from source code. *FSE 2014: Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*. Guimaraes, 2014:472–483
8. Gelman A., Carlin J.B. [et al.]. *Bayesian data analysis*. London: CRC Press, 2013:676.
9. Gershman S.J., Blei D.M. A tutorial on Bayesian non-parametric models. *Journal of Mathematical Psychology*. 2012;(56):1–12.
10. Neelakantan A., Le Q.V., Abadi M. [et al.]. Learning a natural language interface with neural programmer. *Proceedings of the 5th International Conference on Learning Representations (ICLR)*. 2017:1332–1342.
11. Bozhday A.S., Evseeva Yu.I. The method of reflexive self-adaptation of software systems. *Izvestiya vysshikh uchebnykh zavedeniy. Povolzhskiy region. Tekhnicheskie nauki = University proceedings. Volga region. Engineering sciences*. 2018;(2):74–86. (In Russ.)
12. Bershadskiy A.M., Bozhday A.S., Evseeva Yu.I., Gudkov A.A. Self-adaptation of training software systems based on monitoring the information environment. *Otkrytoe obrazovanie = Open education*. 2019;(3):33–41. (In Russ.)
13. Bozhday A.S., Evseeva Yu.I., Artamonov D.V. Using reinforcement learning to build self-adaptive software. *Izvestiya vysshikh uchebnykh zavedeniy. Povolzhskiy region. Tekhnicheskie nauki = University proceedings. Volga region. Engineering sciences*. 2019;(3):56–68. (In Russ.)

Информация об авторах / Information about the authors

Александр Сергеевич Бождай
доктор технических наук, доцент, профессор кафедры систем автоматизированного проектирования, Пензенский государственный университет (Россия, г. Пенза, ул. Красная, 40)

E-mail: bozhday@yandex.ru

Aleksandr S. Bozhday
Doctor of engineering sciences, associate professor, professor of the sub-department of computer-aided design system, Penza State University (40 Krasnaya street, Penza, Russia)

Юлия Игоревна Евсеева
кандидат технических наук, доцент кафедры систем автоматизированного проектирования, Пензенский государственный университет (Россия, г. Пенза, ул. Красная, 40)

E-mail: shymoda@mail.ru

Yuliya I. Evseeva
Candidate of engineering sciences, associate professor of the sub-department of computer-aided design system, Penza State University (40 Krasnaya street, Penza, Russia)

Алексей Анатольевич Гудков
кандидат технических наук, доцент,
доцент кафедры систем
автоматизированного проектирования,
Пензенский государственный
университет (Россия, г. Пенза,
ул. Красная, 40)

E-mail: alexei-ag@yandex.ru

Aleksey A. Gudkov
Candidate of engineering sciences, associate professor, associate professor
of the sub-department of computer-aided design system, Penza State University
(40 Krasnaya street, Penza, Russia)

Авторы заявляют об отсутствии конфликта интересов / The authors declare no conflicts of interests.

Поступила в редакцию / Received 16.07.2021

Поступила после рецензирования и доработки / Revised 25.09.2021

Принята к публикации / Accepted 11.10.2021