ОСНОВНЫЕ ПОЛОЖЕНИЯ ПАРАДИГМЫ ЯЗЫКА ЕДИНОГО ПРЕДСТАВЛЕНИЯ ПРОГРАММНОГО КОДА В ИНТЕРЕСАХ ПОИСКА В НЕМ СРЕДНЕ- И ВЫСОКОУРОВНЕВЫХ УЯЗВИМОСТЕЙ

Буйневич М.В.[1], Израилов К.Е.[2], Покусов В.В.[3]

Цель исследования: *повышение эффективности работы эксперта по поиску среднеуровневых (в алгоритмах) и высокоуровневых (в архитектуре) уязвимостей в программном коде за счет инновационной парадигмы языка его представлению.*

Метод: *заключается в анализе релевантных работ на предмет подходов, способов и нотаций представления алгоритмов и архитектуры программного обеспечения с выделением сильных и слабых сторон решений, синтезе парадигмы представления программного кода и качественной оценки эффективности каждого из положений парадигмы (методом от противного); под эффективностью понимается совокупность трех ее показателей: количество ошибок I и II рода, время поиска и когнитивное напряжение эксперта.*

Полученные результаты: *описание идеи и 7-ми основных положений парадигмы языка псевдокода для единого описания алгоритмов и архитектуры с максимально необходимой и минимально достаточной степенью формализации; главной практической значимостью получаемых таким образом представлений программного кода является их предназначенность для анализа экспертом по информационной безопасности на предмет наличия средне- и высокоуровневых уязвимостей; также для каждого положения установлено их качественное влияние на показатели эффективности поиска уязвимостей экспертом.*

Ключевые слова: *информационная безопасность, программное обеспечение, уязвимость, язык представления программного кода, парадигма.*

DOI:10.21681/2311-3456-2021-6-78-89

1. Введение

Важнейшей проблемой современного цифрового мира считается безопасность данных,в гигантском объеме циркулирующих в инфосфере любого государства [1]. Один из источников информационных угроз – небезопасность программного обеспечения, как раз обрабатывающего большую часть этих данных. Причиной «опасного» (ошибочного или злонамеренного) функционирования программного обеспечения являются уязвимости в его коде, часть которых может быть обнаружена автоматическими средствами поиска [2, 3], а для другой – требуется привлечение ручного труда экспертов [4, 5]. В интересах последнего необходимо представление анализируемого кода в человеко-ориентированном виде, отражающем все необходимые для поиска уязвимостей признаки. Ситуация усложняется тем, что уровень «внедрения» уязвимостей в структуру программы (например, на уровне алгоритмов, архитектуры и выше) может оказаться отличным от уровня классического представления программы (т.е. в виде исходного кода, детализирующего реализацию подпрограмм), что не позволит эксперту эффективно (т.е. с минимальным количеством ошибок I и II рода, адекватными временными затратами и низкой когнитивной нагрузкой) обнаруживать ошибки в алгоритмах, *backdoor*-ы в модулях, концептуальные просчеты в общем построении программной системы и т.п. Также особо следует отметить, что большинство автоматических средств поиска уязвимостей работает именно с низкоуровневыми представлениями программ – в виде исходного кода, что естественным образом приводит к пропуску более высокоуровневых ошибок. Таким образом, задача представления кода, в котором бы отражались и алгоритмы, и архитектура программы, и при этом ориентированного на работу с ним эксперта (гипотетически применяющего вспомогательные средства анализа) является по-настоящему актуальной. И первым шагом в направлении ее решения должна стать разработка общей парадигмы языка описания такого представления, основные положения

1 Буйневич Михаил Викторович, доктор технических наук, профессор, профессор кафедры прикладной математики и информационных технологий Санкт-Петербургского университета государственной противопожарной службы МЧС России. Санкт-Петербург, Россия.
ORCID: https://orcid.org/0000-0001-8146-0022. Scopus Author ID: 56122749800. E-mail: bmv1958@yandex.ru

2 Израилов Константин Евгеньевич, кандидат технических наук, доцент кафедры защищенных систем связи Санкт-Петербургского государственного университета телекоммуникаций им. проф. М.А. Бонч-Бруевича, старший научный сотрудник лаборатории проблем компьютерной безопасности Санкт-Петербургского Федерального исследовательского центра Российской академии наук. Санкт-Петербург, Россия.
ORCID: https://orcid.org/0000-0002-9412-5693. Scopus Author ID: 56123238800. E-mail: konstantin.izrailov@mail.ru

3 Покусов Виктор Владимирович, Председатель Казахстанской ассоциации информационной безопасности. Алматы, Казахстан.
ORCID: https://orcid.org/0000-0002-5251-3452. Scopus Author ID: 57195494534. E-mail: v@victor.kz

которого и будут предложены, доказаны и продемонстрированы далее.

2. Обзор релевантных работ

Произведем обзор научных работ, затрагивающих область представления алгоритмов и архитектуры программного обеспечения, на предмет используемых в них с этой целью подходов, способов и нотаций.

Работа Журова Д.П. и соавт. [6] посвящена применению для описания алгоритмов специального псевдокода на базе внешнего предметно-ориентированного языка *DSL ANTLR* (*аббр. от англ.* Domain Specific Language ANother Tool for Language Recognition), позволяющего затем генерировать их графическое представление в виде блок-схем.

В исследовании Шевелёва И.А. и Федяева О.И. [7] псевдокод используется для описания «кейсов», участвующих в автоматическом тестировании; показана высокая эффективность такого подхода.

В учебном пособии Князевой М.Д. [8] дается общее представление об «алгоритмике»; в частности, выделяются следующие свойства алгоритмов: дискретность, детерминированность, результативность, конечность, массовость, эффективность, компактность. В качестве способов записи алгоритмов приводятся такие, как словесная запись, псевдокод и графическая схема; среди структур алгоритмов выделяются линейная, разветвленная и циклическая. Примерно аналогичные сведения приводятся и в учебном пособии Саратовского государственного технического университета [9].

Статья Кащей В.В. в сборнике научных трудов [10] посвящена способам записи алгоритмов с учетом парадигм структурного программирования, а также формализации их записи в интересах автоматизация процесса. Рассматриваются такие вопросы предметной области, как графическая запись алгоритмов, использование в псевдокоде собственно вводимых элементов, а также необходимость ознакомления разработчика алгоритмов с тезаурусом используемого языка.

Бузовский О.В. и соавт. в своей статье [11] описывают систему, использующую графические схемы алгоритмов для генерации кода на различных языках программирования. Для унификации операций, проводимых над компонентами граф-схем алгоритмов, выделена абстракция, содержащая общие для различных типов элементов схем свойства и методы их обработки.

Автор в статье [12] предлагает расширение классического языка *C*, позволяющего более безопасно описывать алгоритмы программного кода. В частности, данное расширение может использоваться в задачах реверс-инжиниринга машинного кода.

Публикация Зобнина Р.Е. [13] представляет собой свидетельство о государственной регистрации программы для ЭВМ, предназначенное для работы с диаграммами Насси-Шнейдермана, используемыми как альтернатива при графической записи схем алгоритмов. Особенностью диаграмм является высокая компактность и «заточенность» исключительно на структурное программирование (т.е. в котором отсутствуют безусловные переходы, типа *GOTO*).

В работах Пароджанова В.Д. [14, 15] описывается авторский графический язык представления алгоритмов любого назначения и области применения – ДРАКОН. Особенностью языка является совмещение в нем как классического (структурного, процедурного и пр.) представления программ, так и использование конечного автомата состояний.

В исследовании [16] излагается авторская концепция проектирования архитектуры современных систем защиты с выделением следующих основных ее принципов: искусственность декомпозиции, необходимость и достаточность подсистем, структурная инвариантность, универсализация взаимодействия, единое информационное пространство, – которые могут применяться и при описании архитектуры программного кода.

В статьях [17-19] приводится авторская утилита, применяемая для поиска уязвимостей в коде телекоммуникационных устройств. В аспекте решаемой задачи архитектура утилиты описывается по трем слоям: функциональному (взаимосвязь модулей и их функционала), информационному (преобразование данных) и модульно-алгоритмическому (схемы алгоритмов модулей), – в том числе с применением диаграммы Насси-Шнейдермана для анализа структурированных алгоритмов.

В работе Гордиенко А.В. и соавт. [20] описывается архитектура систем, взаимодействие в которых осуществляется посредством данных. Особенностями такой архитектуры является модифицируемость, расширяемость и устойчивость. Также обоснована возможность применения компонентных технологий при разработке таких архитектур.

Яичный И.В. и Клименко А.Я. в своей работе [21] приводят архитектуру для создания автоматизированных коммерческо-технологических систем; для описания архитектуры используется граф, связывающий инструментальные средства, модули и их потребителей. Также указывается на использование шаблонов проектирования, соответствующих решению некоторой распространенной проблемы. Приводится модель архитектуры (в виде древовидного графа), представляющая процесс расширения функционала новыми модулями.

Подход к созданию формального метаязыка среды описания предметно-ориентированных (т.е. «заточенных» на конкретную предметную область или задачу) языков моделирования рассматривается Суховым А.В. в статье [22]. Для этого им используются грамматики на основе графов; даются определения конструкций метаязыка. По мнению автора, применение формального определения на основе графовых моделей позволяет разрабатывать алгоритмы горизонтальной и вертикальной трансформации метамоделей и созданных на их основе моделей информационных систем, а также их предметных областей.

В статье Зеленовых С.В. и С.А. [23] обосновывается применения *AADL* (*аббр. от англ.* Architecture Analysis and Design Language – язык анализа и проектирования архитектуры) в аспекте безопасности создаваемых критических программных систем; для этого, в частности, применяется *EMA* (*аббр. от англ.* Error Model Annex – приложение к архитектурной модели ошибок). Такой

подход призван позволить универсализировать описание требований безопасности к создаваемой системе, где традиционно для этого используются марковские цепи и логико-вероятностные функции.

Доклад Левина И.И. и соавт. на Международной суперкомпьютерной конференции [24] посвящен развитию разработанного им еще в 1987 г. языка программирования *COLAMO* (*аббр. от англ.* Common Oriented Language for Architecture of Multi Objects – общеориентированный язык описания архитектуры многоцелевых объектов), одной из особенностей которого является поддержка параллелизма, а также описание структурной и процедурной компонент программы. В докладе представлены его следующие расширения, повышающие эффективность цифровой обработки сигналов: виртуальная переменная, операция выделения бита и поддержка внешней константы. Суть первого расширения заключается в переменной, присваивание которой происходит только первый раз; второго – в выделении в переменной только необходимого диапазона бит; а третьего – в модификации констант для упрощения разработки параллельных программ.

В своих статьях [25, 26] Михеевой В.Д. приводятся общие подходы к расширению языков программирования, а именно: создание библиотек на другом языке с предоставлением доступа по *API*, (*аббр. от англ.* Application Programming Interface – программный интерфейс приложения, интерфейс прикладного программирования), реализация нового функционала на подмножестве текущего языка программирования (возможно, с подключением библиотек), выполнение кода на другом языке программирования внутри текущего с помощью интерпретаторов, добавление в язык программирования новых конструкций.

Э. Юан в [27] предлагает использовать стандарт *WOL* (*аббр. от англ.* Web Ontology Language – язык описания Веб-онтологий) для представления архитектуры программного обеспечения. Так, хотя основным предназначением *WOL* является описание веб-страниц, тем не менее, сфера его применения (за счет достаточно высокого уровня абстракции) оказывается более широкой, что было доложено им на конференции ECASE-2017.

В. Чен с соавт. в докладе на Первом международном семинар по образовательным технологиям и информатике [28] представили модель, описывающую архитектуру в виде слоев, используя при этом специальные абстрактные типы данных. В модели выделяются следующие четыре слоя: система, компоненты, взаимосвязь классов и собственно классы.

На Международной конференции по архитектуре программного обеспечения (ICSA) в 2017 был представлен доклад [29], в котором В. Фам и соавт. предложили упростить синхронизацию между программным кодом и его архитектурой путем применения диаграмм составной структуры и конечного автомата из состава *UML*. Докладчики также представили предварительные экспериментальные результаты, подтверждающие эффективность конечных *UML*-автоматов и модели составной структуры для проектирования архитектуры.

Из результатов анализа релевантных работ можно сделать следующие предварительные выводы.

Во-первых, основными способами представления алгоритмов является его словесное описание, псевдокод и графическая схема. К последней относятся классическая блок-схема [6], диаграмма Насси-Шнейдермана [13, 17–19] и представление ДРАКОН [14, 15].

Во-вторых, какого-либо общепринятого, достаточно универсального и потому широко используемого подхода для представления архитектуры программ найдено не было, а в подавляющей части работ приводится ее графическое описание [6, 8–11, 14, 15, 21, 29] и предлагается использование специальных нотаций [6–12, 22–27].

И, в-третьих, попытки совмещения в едином представлении, как алгоритмов, так и архитектуры в релевантных работах отсутствуют в принципе.

Таким образом, какого-либо удовлетворительного решения по единому представлению алгоритмов и архитектуры найдено не было. Поэтому потребуется разработка собственного представления, суммируя преимущества и нивелируя недостатки существующих подходов, способов и нотаций, освещенных в проанализированных научных работах.

3. «Классические» парадигмы и идея новой парадигмы

В интересах создания будущей парадигмы для единого представления алгоритмов и архитектуры программного кода выделим принципиальные положения «классических» (наиболее часто используемых при разработке программного обеспечения) парадигм программирования [30]:

Для императивного программирования – это запись инструкций, которые выполняются последовательно; в частности, парадигма детализируется на следующие:

- для структурного программирования – это представление программы в виде иерархической структуры блоков;
- для процедурного программирования – последовательно выполняемые операторы собираются в подпрограммы;
- для объектно-ориентированного программирования – программа представляется в виде совокупности объектов, каждый из которых является экземпляром определённого класса, которые образуют иерархию наследования;
- для декларативного программирования – это задание спецификации решения задачи, то есть описание того, что представляет собой проблема и ожидаемый результат; в частности, парадигма детализируется на следующие:
- для функционального программирования – процесс вычисления трактуется как вычисление значений функций в математическом понимании;
- для логического программирования – программа основывается на автоматическом доказательстве теорем, а также разделе дискретной математики, изучающей принципы логического вывода информации.

Ряд приведенных особенностей рассмотренных парадигм, как хорошо зарекомендовавших себя в практике поиска средне- и высокоуровневых уязвимо-

стей, положим в основу создаваемой парадигмы единого языка представления программного кода. В качестве идеи парадигмы возьмем следующую, поделенную на части (отмеченные соответствующими числами): программа представляет собой единый псевдокод (1), описывающий как ее алгоритмы, так и архитектуру (2), обладающий максимально необходимой и минимально достаточной степенью формализации (3), при этом хорошо понятной эксперту (4). Таким образом, общая идея парадигмы состоит из композиции 4 частных идей.

Используя подходы, «экстрагированные» в результате проведенного аналитического обзора (см. п. 2), классические принципы парадигм программирования (см. п. 3), а также опираясь на значительный опыт научного коллектива в моделировании, проектировании, разработке и применении различных концептов для сферы информационной безопасности [31-41], декларируем следующие положения, лежащие в основе парадигмы единого языка представления алгоритмов и архитектуры программного кода.

Эти положения содержат в себе условия, следствием соблюдения которых является гипотетический рост эффективности работы эксперта по поиску средне- и высокоуровневых уязвимостей в программном коде. Поэтому их доказательство будет лежать в плоскости прогнозируемого (в случае соблюдения требований) изменения показателей ее составляющих, указанных во введении, а именно: минимального количества ошибок I и II рода (суммарное количество ошибок поиска – R), адекватных временных затрат (время поиска – T) и низкой когнитивной нагрузки (умственное напряжение или усталость при поиске – C), – и последующего сравнения с желаемыми (R ↓, T ↓ и C ↓).

Ввиду отсутствия на данном этапе развития парадигмы достаточно строгой математической модели количественной оценки эффективности работы эксперта построим доказательство методом «от противного» и в качественном измерении.

4. Основные положения новой парадигмы языка

Положение 1. Для описания алгоритмов и архитектуры программного кода используется единый псевдокод (т.е. применяется некоторая общая нотация).

Доказательство. Предположим, для описания алгоритмов и архитектуры программы применяются различные нотации; например, псевдокод подпрограмм vs схема взаимодействия модулей. Тогда, помимо несоответствия представления частной идее (1) новой парадигмы, эксперт не всегда сможет корректно интерпретировать назначение подпрограмм (поскольку оно зависит от вызываемого контекста – архитектуры); что приведет к росту суммарного количества ошибок поиска (R ↑).

Положение 2. В псевдокоде отсутствуют мелкие детали, несущественные для понимания архитектуры и алгоритма – условие достаточной детализации (например, упраздняются типы данных, формулы для вычислений, работа с битами и пр.).

Доказательство. Предположим, для описания алгоритмов и архитектуры программы в псевдокоде присутствуют мелкие, несущественные детали; например, у переменных указаны их типы. Тогда, помимо несоответствия преставления частной идее (3) новой парадигмы, эксперт будет неоправданно дольше работать с информацией, излишней для поиска среднеуровневых уязвимостей (таких, как условия переходов в логике выполнения); что автоматически приведет к росту времени поиска (T ↑).

Положение 3. В псевдокоде присутствует подробное описание архитектурных элементов, связывающее последних с алгоритмическими элементами – условие необходимой детализации (например, помимо взаимосвязи модулей описывается логика работы последних).

Доказательство. Предположим, в псевдокоде отсутствует описание архитектуры в связке с алгоритмами ее модулей; например, не указана организация подпрограмм в модули. Тогда, помимо несоответствия преставления частной идее (2) новой парадигмы, эксперту будет сложнее и не всегда возможно обнаруживать высокоуровневые уязвимости (такие, как небезопасное взаимодействие модулей), поскольку целевое назначение архитектурных конструкций понимается на основании задач, решаемых подпрограммами и их алгоритмами; что приведет к росту суммарного количества ошибок поиска (R ↑) и когнитивной перегрузке за счет роста умственного напряжения (C ↑).

Положение 4. Для упрощения описания используются шаблоны проектирования, заменяющие собой стандартные структуры как архитектуры, так и ее алгоритмов.

Доказательство. Предположим, для описания алгоритмов и архитектуры программы в псевдокоде не используются шаблоны проектирования; например, код описывает алгоритмы подпрограмм и их взаимные вызовы без выделения часто применяемых (т.к. ставших классическими) конструкций частей архитектуры или алгоритма. Тогда, помимо несоответствия представления частной идее (4) новой парадигмы, эксперт затратит более существенные усилия на распознавание уязвимостей, связанных с отличием реализации программы от уже многократно проверенных на практике подходов и решений; что приведет к росту усталости при поиске (C ↑).

Положение 5. При невозможности формализации элементов алгоритмов и архитектуры используется словесное описание (например, в виде комментариев).

Доказательство. Предположим, в случае невозможности нотационного описания алгоритмов и архитектуры программы в псевдокоде не используются комментарии – т.е. некоторые их элементы попросту опускаются. Тогда, помимо несоответствия представления частной идее (3) новой парадигмы, эксперту будет предоставлена информация о программе не в полном объеме, что с большой вероятностью приведет к росту суммарного количества ошибок поиска (R ↑).

Положение 6. Обеспечивается возможность расширения как способов описания алгоритмов, так и архитектуры для «наклонения» (т.е. корректировки) нотации к заданной цели (например, путем добавления элементов информационной безопасности такой, как потенциальные уязвимости или механизма их недопущения).

DOI:10.21681/2311-3456-2021-6-78-89

Доказательство. Предположим, псевдокод алгоритмов и архитектуры программы не предназначен для предметно-ориентированного расширения описания в интересах отдельных задач; например, отсутствуют механизмы отображения уязвимостей, найденных автоматически при генерации псевдокода. Тогда, помимо несоответствия представления частной идее (3) новой парадигмы, эксперт может пропустить потенциальные уязвимости или затрачивать на это дополнительное время; что приведет к росту времени (T ↑) и суммарного количества ошибок поиска (R ↑).

Положение 7. Обеспечивается иерархичность слоев описания как алгоритмов программы, так и ее архитектуры, что позволит масштабировать их представление (например, дроблением модулей на подмодули, подподмодули и т.д., а алгоритмов на метаалгоритмы и т.п.).

Доказательство. Предположим, для описания алгоритмов и архитектуры программы в псевдокоде не используются метаалгоритмы и объединения модулей в отдельные структуры; например, код представляет собой лишь линейную последовательность алгоритмов, разделенных на отдельные модули. Тогда, помимо несоответствия представления частной идее (4) новой парадигмы, эксперту будет сложнее распознавать уязвимости ошибочного вызова подпрограмм (например, подпрограмма очистки объектов вызывается дважды) и небезопасного проектирования (например, когда модули работы с локальными секретными ключами тесно взаимодействуют с модулями работы через открытые сети); что приведет к когнитивной перегрузке за счет роста умственного напряжения (C ↑).

Сводная таблица влияния несоблюдения каждого из положений новой парадигмы на эффективность поиска в псевдокоде средне- и высокоуровневых уязвимостей экспертом представлена в табл. 1.

Табличный анализ влияния (см. табл. 1) позволяет сделать выводы, что любое отклонение псевдокода от положений новой парадигмы, так или иначе, снижает эффективность поиска средне- и высокоуровневых уязвимостей экспертом. При этом в случае отклонения наиболее подверженным негативному эффекту является количество ошибок I и II рода, а наименее подверженным – время работы эксперта с псевдокодом.

5. Пример

Используя положения, лежащие в основе новой парадигмы языка представления, приведем следующий гипотетический пример описания алгоритмов и архитектуры программы типового калькулятора (на гипотетическом псевдокоде). Соответствие строк псевдокода примера каждому из положений парадигмы представлено в табл. 2.

Псевдокод дает описание архитектуры и алгоритмов программы посредством следующих объектов, их взаимосвязей и свойств:

1) *Calculator* – главный модуль калькулятора, работающий по шаблону консольного приложения (*ConsoleTemplate*) и содержащий подпрограмму ввода исходных операндов (через подпрограмму *Operands()* в модуле *Input*) и операции (через подпрограмму *Operation()* в модуле *Input*) вычислений (через соответствующие подпрограммы в модуле *Output*) и возврата результата (через подпрограмму *Result()* в модуле *Output*).

2) *Operation* – функциональный модуль (т.е. ядро) калькулятора для операций сложения (подпрограмма Add()), вычитания (подпрограмма *Sub()*), умножения (подпрограмма *Mul()*) и деления (подпрограмма *Div()*); при этом, в реализации последней присутствует «неправильная» проверка на ошибку «Деление на 0», приводящую к выводу соответствующего сообщения (подпрограмма *Error()* у модуля *Output*) и выходу (подпрограмма *Exit()*); уязвимость в данном случае заключается в том, что согласно условию ошибка выведется при делении на любое число, кроме положительного, хотя деление на отрицательные числа корректно;

3) *Input* – модуль ввода пользовательских данных, работающий по шаблону стандартного ввода (*InputTemplate*) и содержащий подпрограмму *Operands()* для ввода операндов и *Operation*() – для вывода операций;

4) *Output* – модуль вывода пользовательских данных и ошибок (работающий по шаблону стандартного ввода – *OutputTemplate*) и содержащий подпрограмму *Result*() для вывода результата вычисления калькулятора.

Также в коде присутствуют дополнительные комментарии, обозначаемые с помощью стандартной C/C++ конструкции «//».

Таблица 1

Влияние несоблюдения положений новой парадигмы на эффективность поиска уязвимостей

Показатели эффективности	Положения новой парадигмы						
	1	2	3	4	5	6	7
Количество ошибок I и II рода (R)	↑		↑		↑	↑	
Временные затраты (T)		↑				↑	
Когнитивная нагрузка (C)			↑	↑			↑

Таблица 2

Соответствие строк псевдокода примера каждому из положений новой парадигмы

Строки псевдокода	Положения новой парадигмы						
	1	2	3	4	5	6	7
MODULE Program {							×
// Главный модуль Калькулятора					×		
MODULE Calculator : ConsoleTemplate {	×			×			
Main() {			×				
(X, Y) = Input.Operands();		×					
Op = Input.Operands();		×					
Z = SWITCH (Op) {		×					
‘+’: Operation.Add(X, Y);							
‘-’: Operation.Sub(X, Y);							
‘*’: Operation.Mul(X, Y);							
‘/’: Operation.Div(X, Y);							
}							
Output.Result(Z);							
}							
}							
// Ядро Калькулятора					×		
MODULE Operation {	×						
Add (X, Y) {		×					
RETURN X + Y;							
}							
Sub (X, Y) {		×					
RETURN X - Y;							
}							
Mul (X, Y) {		×					
RETURN X * Y;							
}							
Div (X, Y) {		×					
VULNERABILITY (Medium, InvalidCondition);						×	
// Верная запись: “IF (Y == 0) {”					×		
IF (Y <= 0) {							
// Вывод ошибки: «Деление на 0»					×		
Output.Error(“Divided by zero”);				×			
EXIT;							
}							
RETURN X / Y;							
}							
}							
// Модуль ввода пользовательских данных					×		
MODULE Input : InputTemplate {	×			×			
Operands () {							
X = Parent.Read;		×		×			
Y = Parent.Read;		×		×			
RETURN (X, Y);							
}							
Operation () {							

DOI:10.21681/2311-3456-2021-6-78-89

Строки псевдокода	Положения новой парадигмы						
	1	2	3	4	5	6	7
Op = Parent.Read;		×		×			
RETURN Op;							
}							
}							
// Модуль вывода пользовательских данных					×		
MODULE Output : OutputTemplate {	×						
Result (Z) {		×					
Parent.Write(Z);				×			
}							
Error (S) {		×					
Parent.WriteError(S);				×			
}							
}							
}							×

Обоснуем соответствие псевдокода каждому из положений парадигмы.

Соответствие Положению 1 обуславливается наличием в псевдокоде ключевого слова *MODULE*, объединяющего модуль архитектуры и входящие в него подпрограммы.

Соответствие Положению 2 обуславливается тем, что алгоритмы подпрограмм записаны при помощи переменных без явного указания их типа.

Соответствие Положению 3 обуславливается наличием подпрограммы *Main*, реализующей основную ветку выполнения, вызывающую все остальные подпрограммы (ближайшего уровня вложенности).

Соответствие Положению 4 обуславливается использованием в архитектуре шаблонов консольного приложения (*ConsoleTemplate*) и ввода/вывода данных (*InputTemplate, OutputTemplate*), а в алгоритмах – операций работы с пользовательской консолью (*Read*, *Write*, *WriteError*).

Соответствие Положению 5 обуславливается наличием комментариев, замена которых на какие-либо встроенные языковые конструкции невозможна по причине их высокой специализированности (очевидно, что языковые конструкции обладают возможностями обобщения описания программы).

Соответствие Положению 6 обуславливается появлением в коде конструкции *VULNERABILITY*, сигнализирующей о потенциально неверном условии в коде (*InvalidCondition*) – среднеуровневой уязвимости (*Medium*). В данном случае это считается расширением псевдокода, поскольку изначально в идею новой парадигмы было заложено лишь отображение алгоритмов и архитектуры, по которым эксперт сможет искать средне- и высокоуровневые уязвимости, а новая конструкция позволяет отображать потенциальные уязвимости, найденные автоматическими средствами.

Соответствие Положению 7 обуславливается наличием основного модуля программы *Program*, в который входя модули калькулятора *Calculator*, *Operation*, *Input* и *Output*.

Таким образом, можно утверждать, что приведенный псевдокод примера соответствует новой парадигме языка единого представления программного кода в интересах поиска в нем средне- и высокоуровневых уязвимостей.

6. Заключение

В результате обзора научных работ, близких к решаемой задаче представления алгоритмов и архитектуры программного обеспечения (в рамках единой парадигмы), было установлено, что все они лишь частично затрагивают данную область. Вследствие этого была создана авторская парадигма представления, основанная на 7-ми положениях – единый псевдокод, условия необходимой и достаточной детализации, введение шаблонов, возможность словесного описания, расширяемость и иерархичность. Приведенный гипотетический пример подтвердил обоснованность положений и работоспособность новой парадигмы.

Полученные результаты в виде идеи и положений новой парадигмы являются мощным теоретико-практическим (а отчасти и методологическим) аппаратом, связывающим машинно-ориентированную область кодирования программ (представление программы в некоторой нотации) с человеко-ориентированной областью осознания гетерогенной информации экспертом (поиск в программе сущностей, не отраженных явно, таких как уязвимость). Качественная оценка положений позволяет утверждать, что любое представление программы, преобразованное в псевдокод согласно предложенной парадигме, может быть проанализировано экспертом на предмет наличия средне- и высокоуровневых уязвимостей с большей эффективностью, чем в случае использования существующих классических представлений программного кода таких, как блок-схемы, языки программирования и др.

Исходя из сделанного обзора релевантных работ установлено, что способы записи алгоритмов и архитектуры в едином представлении, как и адаптация представлений для поиска средне- и высокоуровневых уязвимостей, отсутствуют принципе, и, следовательно, предложенная парадигма обладает безусловной новизной. Способ же ее косвенной оценки (с позиции использования псевдокода экспертом) через установление влияния положений парадигмы на показатели эффективности может считаться авторским, поскольку в большинстве случаев классические парадигмы вводились *de-facto* (т.е. на веру разработчикам программного обеспечения) без формального обоснования необходимости их применения, а любые оценки применялись лишь к конкретному синтаксису языка программирования [42] или вариантам программного кода.

Продолжение работы авторы видят в следующем. Во-первых, необходимо апробирование парадигмы псевдокода единого представления алгоритмов и архитектуры на практике и, в частности, его применимость для поиска реальных уязвимостей. Во-вторых, потребуется синтез и разработка моделей, алгоритмов и программных средств для генерации такого псевдокода по различным первоначальным представлениям программ, наиболее важным среди которых с точки зрения безопасности является машинное (имеющее бинарную и слабоструктурированную форму, не подходящую для глубокого экспертного анализа). И, в-третьих, для повышения автоматизации процесса поиска уязвимостей экспертом (в основном за счет снижения рабочего времени и когнитивной нагрузки) существует востребованность в создании специализированных алгоритмов выявления признаков средне- и высокоуровневых уязвимостей с отображением соответствующей информации в псевдокоде. Частично это планируется сделать в продолжении текущей научно-исследовательской работы.

Работа выполнена в рамках реализации проекта AP06851400 «Разработка способа и автоматизация поиска уязвимостей в машинном коде телекоммуникационных устройств» (при финансовой поддержке Министерства образования и науки Республики Казахстан)

Литература

1. Шадрин Д.В. Актуальность угроз информационной безопасности для информационных систем // NovaInfo.Ru. 2016. Т. 3. № 53. С. 41-44. eLIBRARY ID: 27185592
2. Несов В.С., Маликов О.Р. Автоматический поиск уязвимостей в больших программах // Информационное противодействие угрозам терроризма. 2006. № 7. С. 281-291. eLIBRARY ID: 9572348
3. Аветисян А., Белеванцев А., Бородин А., Несов В. Использование статического анализа для поиска уязвимостей и критических ошибок в исходном коде программ // Труды Института системного программирования РАН. 2011. Т. 21. С. 23-38. eLIBRARY ID: 17103027
4. Благодаренко А.В. Методика автоматизированного поиска уязвимостей в программном обеспечении без исходных кодов // Системы высокой доступности. 2011. Т. 7. № 2. С. 65-69. eLIBRARY ID: 17098464
5. Буйневич М.В., Израилов К.Е., Мостович Д.И. Сравнительный анализ подходов к поиску уязвимостей в программном коде // Актуальные проблемы инфотелекоммуникаций в науке и образовании (АПИНО-2016): сборник научных статей V международной научно-технической и научно-методической конференции. СПбГУТ. 2016. С. 256-260. eLIBRARY ID: 27296306
6. Журов Д.П., Пятых С.О., Сосинская С.С. Использование внешнего DSL ANTLR для разработки программного обеспечения построения блок-схем по псевдокоду // Austrian Journal of Technical and Natural Sciences. 2015. № 5-6. С. 49-50. eLIBRARY ID: 25620176
7. Шевелёв И.А., Федяев О.И. Формализация представления тест-кейсов с помощью тест-ориентированного псевдокода // Информатика, управляющие системы, математическое и компьютерное моделирование (ИУСМКМ-2020): сборник материалов XI Международной научно-технической конференции в рамках VI Международного Научного форума Донецкой Народной Республики. Донецк. 2020. С. 82-86. eLIBRARY ID: 42907950
8. Князева М.Д. Алгоритмика: от алгоритма к программе: учебное пособие. Сер. Информатика. Москва: КУДИЦ-ОБРАЗ, 2006. 192 с. . eLIBRARY ID: 23878311
9. Степанов А.М., Степанов М.Ф. Основы алгоритмизации и программирования на языке С: учебное пособие. Саратов: Саратовский государственный технический университет, 2016. 88 с. eLIBRARY ID: 27462813
10. Кащей В.В. Методические особенности использования средств автоматизации записи алгоритмов и программ в курсе информатики // Конференциум АСОУ: сборник научных трудов и материалов научно-практических конференций. 2016. № 2. С. 720-733. eLIBRARY ID: 27683379
11. Бузовский О.В., Алещенко А.В., Подрубайло А.А. Система автоматической генерации кодов по графическим схемам алгоритмов // Вісник Національного технічного університету України «Київський політехнічний інститут». Серія: Інформатика, управління та обчислювальна техніка. 2009. № 51. С. 208-215. eLIBRARY ID: 22791201
12. Израилов К.Е. Расширение языка «С» для описания алгоритмов кода телекоммуникационных устройств // Информационные технологии и телекоммуникации. 2013. Т. 1. № 2. С. 21-31. eLIBRARY ID: 21133498
13. Зобнин Р.Е. Графический редактор для отображения диаграмм Несси-Шнейдермана: свидетельство о государственной регистрации программы для ЭВМ RU № 2015619448 от 04.09.2015. eLIBRARY ID: 39336508

14. Пароджанов В.Д. Дружелюбные алгоритмы, понятные каждому, как улучшить работу ума без лишних хлопот. Москва: ДМК Пресс, 2010. 464 с. eLIBRARY ID: 20242517
15. Parondzhanov V.D. Graphic syntax of the DRACON language // Programmirovanie. 1995. Т. 21. № 3. С. 45-62. eLIBRARY ID: 12754683
16. Буйневич М.В., Израилов К.Е., Покусов В.В., Ярошенко А.Ю. Основные принципы проектирования архитектуры современных систем защиты // Национальная безопасность и стратегическое планирование. 2020. № 3 (31). С. 51-58. DOI: 10.37468/2307-1400-2020-3-51-58
17. Буйневич М.В., Израилов К.Е. Утилита для поиска уязвимостей в программном обеспечении телекоммуникационных устройств методом алгоритмизации машинного кода. Часть 1. Функциональная архитектура // Информационные технологии и телекоммуникации. 2016. Т. 4. № 1. С. 115-130. eLIBRARY ID: 26191560
18. Израилов К.Е. Утилита для поиска уязвимостей в программном обеспечении телекоммуникационных устройств методом алгоритмизации машинного кода. Часть 2. Информационная архитектура // Информационные технологии и телекоммуникации. 2016. Т. 4. № 2. С. 86-104. eLIBRARY ID: 27468771
19. Израилов К.Е., Покусов В.В. Утилита для поиска уязвимостей в программном обеспечении телекоммуникационных устройств методом алгоритмизации машинного кода. Часть 3. Модульно-алгоритмическая архитектура // Информационные технологии и телекоммуникации. 2016. Т. 4. № 4. С. 104-121. eLIBRARY ID: 29041144
20. Гордиенко А.В., Золотов О.И., Коган В.Е. Архитектура программ моделирования // Современное образование: содержание, технологии, качество. 2010. Т. 2. С. 167-168. eLIBRARY ID: 26598715
21. Ячный И.В., Клименко А.Я. Описание архитектуры инструментальных средств для создания автоматизированных коммерческо-технологических систем // Технические науки – от теории к практике. 2013. № 23. С. 37-44.
22. Сухов А.О. Теоретические основы разработки DSL-инструментария с использованием графовых грамматик // Информатизация и связь. 2011. № 3. С. 35-37. eLIBRARY ID: 17086901
23. Зеленов С.В., Зеленова С.А. Моделирование программно-аппаратных систем и анализ их безопасности // Труды Института системного программирования РАН. 2017. Т. 29. № 5. С. 257-282. DOI: 10.15514/ISPRAS-2017-29(5)-13
24. Левин И.И., Дордопуло А.И., Гудков В.А., Гуленок А.А. Расширение языка программирования высокого уровня COLAMO для битовой обработки // Научный сервис в сети Интернет: все грани параллелизма: труды Международной суперкомпьютерной конференции. Новороссисйк. 2013. С. 371-374. eLIBRARY ID: 22446876
25. Михеева В.Д. Методы расширения языков программирования (часть 1) // Информационно-управляющие системы. 2010. № 4 (47). С. 46-52. eLIBRARY ID: 15002700
26. Михеева В.Д. Методы расширения языков программирования (часть 2) // Информационно-управляющие системы. 2010. № 5 (48). С. 37-42. eLIBRARY ID: 15235777
27. Yuan E. Towards Ontology-Based Software Architecture Representations // Processings of the 1st International Workshop on Establishing the Community-Wide Infrastructure for Architecture-Based Software Engineering (ECASE). Buenos Aires, Argentina. 2017. PP. 21-27. DOI: 10.1109/ECASE.2017.5
28. W. Chen, T. Li and J. Li. Algebraic Model and Formal Description Language of Software Architecture // Processings of the 1st International Workshop on Education Technology and Computer Science. United States. 2009. PP. 659-665. DOI: 10.1109/ETCS.2009.407.
29. Pham V., Radermacher A., Gerard S., Li S. Bidirectional Mapping between Architecture Model and Code for Synchronization // Processings of International Conference on Software Architecture (ICSA). 2017. Pp. 239-242. DOI: 10.1109/ICSA.2017.41
30. Шилов Н.В. Заметки о трёх парадигмах программирования // Компьютерные инструменты в образовании. 2010. № 2. С. 24-37. eLIBRARY ID: 15002994
31. Буйневич М.В., Покусов В.В., Ярошенко А.Ю., Хорошенко С.В. Категориальный подход в приложении к синтезу архитектуры интегрированной системы обеспечения безопасности информации // Проблемы управления рисками в техносфере. 2017. № 4 (44). С. 95-102. eLIBRARY ID: 32601997
32. Покусов В.В. Формализация и определение корректности протокола информационно-технического взаимодействия(на примере интегрированной системы защиты информации) // Информатизация и связь. 2021. № 2. С. 55-68. DOI: 10.34219/2078-8320-2021-12-2-55-68
33. Буйневич М.В., Израилов К.Е. Антропоморфический подход к описанию взаимодействия уязвимостей в программном коде. Часть 1. Типы взаимодействий // Защита информации. Инсайд. 2019. № 5 (89). С. 78-85. eLIBRARY ID: 41221386
34. Буйневич М.В., Израилов К.Е. Антропоморфический подход к описанию взаимодействия уязвимостей в программном коде. Часть 2. Метрика уязвимостей // Защита информации. Инсайд. 2019. № 6 (90). С. 61-65. eLIBRARY ID: 41494732
35. Израилов К.Е., Обрезков А.И., Курта П.А. Подход к выявлению последовательности одноцелевых сетевых атак с визуализацией их прогресса эксперту // Методы и технические средства обеспечения безопасности информации. 2020. № 29. С. 68-69. eLIBRARY ID: 44017276
36. Буйневич М.В., Владыко А.Г., Израилов К.Е., Щербаков О.В. Архитектурные уязвимости моделей телекоммуникационных сетей // Научно-аналитический журнал «Вестник Санкт-Петербургского университета Государственной противопожарной службы МЧС России». 2015. № 4. С. 86-93. eLIBRARY ID: 25294888
37. Израилов К.Е. Модель прогнозирования угроз телекоммуникационной системы на базе искусственной нейронной сети // Вестник ИНЖЭКОНа. Серия: Технические науки. 2012. № 8 (59). С. 150-153. eLIBRARY ID: 18244835
38. Израилов К.Е., Васильева А.Ю. Язык описания модели безопасности телекоммуникационной сети // Новые информационные технологии и системы (НИТиС-2012): труды X Международной научно-технической конференции. Пенза. 2012. С. 272-275. eLIBRARY ID: 28771694

39. Буйневич М.В., Израилов К.Е., Щербаков О.В. Структурная модель машинного кода, специализированная для поиска уязвимостей в программном обеспечении автоматизированных систем управления // Проблемы управления рисками в техносфере. 2014. № 3(31). С. 68-74. eLIBRARY ID: 22553713
40. Буйневич М.В., Израилов К.Е., Щербаков О.В. Модель машинного кода, специализированная для поиска уязвимостей // Вестник Воронежского института ГПС МЧС России. 2014. № 2 (11). С. 46-51. eLIBRARY ID: 21574302
41. Буйневич М.В., Израилов К.Е. Аналитическое моделирование работы программного кода с уязвимостями // Вопросы кибербезопасности. 2020. № 3 (37). С. 2-12. DOI: 10.21681/2311-3456-2020-03-02-12
42. Израилов К.Е. Система критериев оценки способов поиска уязвимостей и метрика понятности представления программного кода // Информатизация и связь. 2017. № 3. С. 111-118. eLIBRARY ID: 29108491

LANGUAGE OF THE PROGRAM CODE UNIFORM PRESENTATION FOR SEARCHING MEDIUM- AND HIGH-LEVEL VULNERABILITIES: THE BASIC PROVISIONS OF PARADIGM

Buinevich M.V.[4], Izrailov K.E.[5], Pokussov V.V.[6]

Purpose of the study: *increasing the efficiency of an expert in searching for medium-level (in algorithms) and high-level (in architecture) vulnerabilities in the program code due to the innovative paradigm of the language for its presentation.*

Method: *consists in the analysis of relevant works on the subject of approaches, methods and notations for representing algorithms and software architecture with highlighting the strengths and weaknesses of solutions, synthesizing the paradigm for the presentation of the program code and qualitatively assessing the effectiveness of each of the provisions of the paradigm (by contradiction method); efficiency is understood as a combination of its three indicators: the number of type I and II errors, the search time and the cognitive stress of the expert.*

The results obtained: *description of the idea and 7 main provisions of the paradigm of the pseudocode language for a unified description of algorithms and architecture with the maximum necessary and minimum sufficient degree of formalization; the main practical significance of the representations of the program code obtained in this way is their intended use for analysis by an information security expert for the presence of medium and high-level vulnerabilities; also, for each position, their qualitative influence on the performance indicators of vulnerability search by an expert was established.*

Keywords: *information security, software, vulnerability, programming language, paradigm.*

References

1. Shadrin D.V. Aktual`nost` ugroz informacionnoj bezopasnosti dlya informacionny`x sistem // NovaInfo.Ru. 2016. T. 3. № 53. S. 41-44. eLIBRARY ID: 27185592
2. Nesov V.S., Malikov O.R. Avtomaticheskij poisk uyazvimostej v bol`shix programmax // Informacionnoe protivodejstvie ugrozam terrorizma. 2006. № 7. S. 281-291. eLIBRARY ID: 9572348
3. Avetisyan A., Belevancev A., Borodin A., Nesov V. Ispol`zovanie staticheskogo analiza dlya poiska uyazvimostej i kriticheskix oshibok v isxodnom kode programm // Trudy` Instituta sistemnogo programmirovaniya RAN. 2011. T. 21. S. 23-38. eLIBRARY ID: 17103027
4. Blagodarenko A.V. Metodika avtomatizirovannogo poiska uyazvimostej v programmnom obespechenii bez isxodny`x kodov // Sistemy` vy`sokoj dostupnosti. 2011. T. 7. № 2. S. 65-69. eLIBRARY ID: 17098464

4 Mikhail V. Buinevich, Dr.Sc., Professor, Saint-Petersburg University of State Fire Service of EMERCOM of Russia. Saint-Petersburg, Russia. E-mail: bmv1958@yandex.ru ORCID: https://orcid.org/0000-0001-8146-0022. Scopus Author ID: 56122749800.

5 Konstantin E. Izrailov, Ph.D., The Bonch-Bruevich Saint-Petersburg State University of Telecommunications, Saint Petersburg Scientific Center RAS. Saint-Petersburg, Russia. E-mail: konstantin.izrailov@mail.ru ORCID: https://orcid.org/0000-0002-9412-5693. Scopus Author ID: 56123238800.

6 Victor V. Pokussov, Kazakhstan Information Security Association. Almaty, Kazakhstan. E-mail: v@victor.kz ORCID: https://orcid.org/0000-0002-5251-3452. Scopus Author ID: 57195494534.

5. Bujnevich M.V., Izrailov K.E., Mostovich D.I. Sravnitel`ny`j analiz podxodov k poisku uyazvimostej v programmnom kode // Aktual`ny`e problemy` infotelekommunikacij v nauke i obrazovanii (APINO-2016): sbornik nauchny`x statej V mezhdunarodnoj nauchno-texnicheskoj i nauchno-metodicheskoj konferencii. SPbGUT. 2016. S. 256-260. eLIBRARY ID: 27296306
6. Zhurov D.P., Pyaty`x S.O., Sosinskaya S.S. Ispol`zovanie vneshnego DSL ANTLR dlya razrabotki programmnogo obespecheniya postroeniya blok-sxem po psevdokodu // Austrian Journal of Technical and Natural Sciences. 2015. № 5-6. S. 49-50. eLIBRARY ID: 25620176
7. Shevelyov I.A., Fedyaev O.I. Formalizaciya predstavleniya test-kejsov s pomoshh`yu test-orientirovannogo psevdokoda // Informatika, upravlyayushhie sistemy`, matematicheskoe i komp`yuternoe modelirovanie (IUSMKM-2020): sbornik materialov XI Mezhdunarodnoj nauchno-texnicheskoj konferencii v ramkax VI Mezhdunarodnogo Nauchnogo foruma Doneczkoj Narodnoj Respubliki. Doneczk. 2020. S. 82-86. eLIBRARY ID: 42907950
8. Knyazeva M.D. Algoritmika: ot algoritma k programme: uchebnoe posobie. Ser. Informatika. Moskva: KUDICz-OBRAZ, 2006. 192 s. eLIBRARY ID: 23878311
9. Stepanov A.M., Stepanov M.F. Osnovy` algoritmizacii i programmirovaniya na yazy`ke S: uchebnoe posobie. Saratov: Saratovskij gosudarstvenny`j texnicheskij universitet, 2016. 88 s. eLIBRARY ID: 27462813
10. Kashhej V.V. Metodicheskie osobennosti ispol`zovaniya sredstv avtomatizacii zapisi algoritmov i programm v kurse informatiki // Konferencium ASOU: sbornik nauchny`x trudov i materialov nauchno-prakticheskix konferencij. 2016. № 2. S. 720-733. eLIBRARY ID: 27683379
11. Buzovskij O.V., Aleshhenko A.V., Podrubajlo A.A. Sistema avtomaticheskoj generacii kodov po graficheskim sxemam algoritmov // Visnik Naczional`nogo texnichnogo universitetu Ukraïni «Kiïvs`kij politexnichnij institut». Seriya: Informatika, upravlinnya ta obchislyuval`na texnika. 2009. № 51. S. 208-215. eLIBRARY ID: 22791201
12. Izrailov K.E. Rasshirenie yazy`ka «C» dlya opisaniya algoritmov koda telekommunikacionny`x ustrojstv // Informacionny`e texnologii i telekommunikacii. 2013. T. 1. № 2. S. 21-31. eLIBRARY ID: 21133498
13. Zobnin R.E. Graficheskij redaktor dlya otobrazheniya diagramm Nessi-Shnejdermana: svidetel`stvo o gosudarstvennoj registracii programmy` dlya E`VM RU № 2015619448 ot 04.09.2015. eLIBRARY ID: 39336508
14. Parondzhanov V.D. Druzhelyubny`e algoritmy`, ponyatny`e kazhdomu, kak uluchshit` rabotu uma bez lishnix xlopot. Moskva: DMK Press, 2010. 464 s. eLIBRARY ID: 20242517
15. Parondzhanov V.D. Graphic syntax of the DRACON language // Programmirovanie. 1995. T. 21. № 3. S. 45-62. eLIBRARY ID: 12754683
16. Bujnevich M.V., Izrailov K.E., Pokusov V.V., Yaroshenko A.Yu. Osnovny`e principy` proektirovaniya arxitektury` sovremenny`x sistem zashhity` // Nacional`naya bezopasnost` i strategicheskoe planirovanie. 2020. № 3 (31). S. 51-58. DOI: 10.37468/2307-1400-2020-3-51-58
17. Bujnevich M.V., Izrailov K.E. Utilita dlya poiska uyazvimostej v programmnom obespechenii telekommunikacionny`x ustrojstv metodom algoritmizacii mashinnogo koda. Chast` 1. Funkcional`naya arxitektura // Informacionny`e texnologii i telekommunikacii. 2016. T. 4. № 1. S. 115-130. eLIBRARY ID: 26191560
18. Izrailov K.E. Utilita dlya poiska uyazvimostej v programmnom obespechenii telekommunikacionny`x ustrojstv metodom algoritmizacii mashinnogo koda. Chast` 2. Informacionnaya arxitektura // Informacionny`e texnologii i telekommunikacii. 2016. T. 4. № 2. S. 86-104. eLIBRARY ID: 27468771
19. Izrailov K.E., Pokusov V.V. Utilita dlya poiska uyazvimostej v programmnom obespechenii telekommunikacionny`x ustrojstv metodom algoritmizacii mashinnogo koda. Chast` 3. Modul`no-algoritmicheskaya arxitektura // Informacionny`e texnologii i telekommunikacii. 2016. T. 4. № 4. S. 104-121. eLIBRARY ID: 29041144
20. Gordienko A.V., Zolotov O.I., Kogan V.E. Arxitektura programm modelirovaniya // Sovremennoe obrazovanie: soderzhanie, texnologii, kachestvo. 2010. T. 2. S. 167-168. eLIBRARY ID: 26598715
21. Yachny`j I.V., Klimenko A.Ya. Opisanie arxitektury` instrumental`ny`x sredstv dlya sozdaniya avtomatizirovanny`x kommercheskotexnologicheskix sistem // Texnicheskie nauki – ot teorii k praktike. 2013. № 23. S. 37-44.
22. Suxov A.O. Teoreticheskie osnovy` razrabotki DSL-instrumentariya s ispol`zovaniem grafovy`x grammatik // Informatizaciya i svyaz`. 2011. № 3. S. 35-37. eLIBRARY ID: 17086901
23. Zelenov S.V., Zelenova S.A. Modelirovanie programmno-apparatny`x sistem i analiz ix bezopasnosti // Trudy` Instituta sistemnogo programmirovaniya RAN. 2017. T. 29. № 5. S. 257-282. DOI: 10.15514/ISPRAS-2017-29(5)-13
24. Levin I.I., Dordopulo A.I., Gudkov V.A., Gulenok A.A. Rasshirenie yazy`ka programmirovaniya vy`sokogo urovnya COLAMO dlya bitovoj obrabotki // Nauchny`j servis v seti Internet: vse grani parallelizma: trudy` Mezhdunarodnoj superkomp`yuternoj konferencii. Novorossisjk. 2013. S. 371-374. eLIBRARY ID: 22446876
25. Mixeeva V.D. Metody` rasshireniya yazy`kov programmirovaniya (chast` 1) // Informacionno-upravlyayushhie sistemy`. 2010. № 4 (47). S. 46-52. eLIBRARY ID: 15002700
26. Mixeeva V.D. Metody` rasshireniya yazy`kov programmirovaniya (chast` 2) // Informacionno-upravlyayushhie sistemy`. 2010. № 5 (48). S. 37-42. eLIBRARY ID: 15235777
27. Yuan E. Towards Ontology-Based Software Architecture Representations // Processings of the 1st International Workshop on Establishing the Community-Wide Infrastructure for Architecture-Based Software Engineering (ECASE). Buenos Aires, Argentina. 2017. PP. 21-27. DOI: 10.1109/ECASE.2017.5
28. W. Chen, T. Li and J. Li. Algebraic Model and Formal Description Language of Software Architecture // Processings of the 1st International Workshop on Education Technology and Computer Science. United States. 2009. PP. 659-665. DOI: 10.1109/ETCS.2009.407.
29. Pham V., Radermacher A., Gerard S., Li S. Bidirectional Mapping between Architecture Model and Code for Synchronization // Processings of International Conference on Software Architecture (ICSA). 2017. PP. 239-242. DOI: 10.1109/ICSA.2017.41

30. Shilov N.V. Zametki o tryox paradigmax programmirovaniya // Komp`yuterny`e instrumenty` v obrazovanii. 2010. № 2. S. 24-37. eLIBRARY ID: 15002994
31. Bujnevich M.V., Pokusov V.V., Yaroshenko A.Yu., Xoroshenko S.V. Kategorial`ny`j podxod v prilozhenii k sintezu arxitektury` integrirovannoj sistemy` obespecheniya bezopasnosti informacii // Problemy` upravleniya riskami v texnosfere. 2017. № 4 (44). S. 95-102. eLIBRARY ID: 32601997
32. Pokusov V.V. Formalizaciya i opredelenie korrektnosti protokola informacionno-texnicheskogo vzaimodejstviya(na primere integrirovannoj sistemy` zashhity` informacii) // Informatizaciya i svyaz`. 2021. № 2. S. 55-68. DOI: 10.34219/2078-8320-2021-12-2-55-68
33. Bujnevich M.V., Izrailov K.E. Antropomorficheskij podxod k opisaniyu vzaimodejstviya uyazvimostej v programmnom kode. Chast` 1. Tipy` vzaimodejstvij // Zashhita informacii. Insajd. 2019. № 5 (89). S. 78-85. eLIBRARY ID: 41221386
34. Bujnevich M.V., Izrailov K.E. Antropomorficheskij podxod k opisaniyu vzaimodejstviya uyazvimostej v programmnom kode. Chast` 2. Metrika uyazvimostej // Zashhita informacii. Insajd. 2019. № 6 (90). S. 61-65. eLIBRARY ID: 41494732
35. Izrailov K.E., Obrezkov A.I., Kurta P.A. Podxod k vy`yavleniyu posledovatel`nosti odnocelevy`x setevy`x atak s vizualizaciej ix progressa e`kspertu // Metody` i texnicheskie sredstva obespecheniya bezopasnosti informacii. 2020. № 29. S. 68-69. eLIBRARY ID: 44017276
36. Bujnevich M.V., Vlady`ko A.G., Izrailov K.E., Shherbakov O.V. Arxitekturny`e uyazvimosti modelej telekommunikacionny`x setej // Nauchno-analiticheskij zhurnal «Vestnik Sankt-Peterburgskogo universiteta Gosudarstvennoj protivopozharnoj sluzhby` MChS Rossii». 2015. № 4. S. 86-93. eLIBRARY ID: 25294888
37. Izrailov K.E. Model` prognozirovaniya ugroz telekommunikacionnoj sistemy` na baze iskusstvennoj nejronnoj seti // Vestnik INZhE`KONa. Seriya: Texnicheskie nauki. 2012. № 8 (59). S. 150-153. eLIBRARY ID: 18244835
38. Izrailov K.E., Vasil`eva A.Yu. Yazy`k opisaniya modeli bezopasnosti telekommunikacionnoj seti // Novy`e informacionny`e texnologii i sistemy` (NITiS-2012): trudy` X Mezhdunarodnoj nauchno-texnicheskoj konferencii. Penza. 2012. S. 272-275. eLIBRARY ID: 28771694
39. Bujnevich M.V., Izrailov K.E., Shherbakov O.V. Strukturnaya model` mashinnogo koda, specializirovannaya dlya poiska uyazvimostej v programmnom obespechenii avtomatizirovanny`x sistem upravleniya // Problemy` upravleniya riskami v texnosfere. 2014. № 3(31). S. 68-74. eLIBRARY ID: 22553713
40. Bujnevich M.V., Izrailov K.E., Shherbakov O.V. Model` mashinnogo koda, specializirovannaya dlya poiska uyazvimostej // Vestnik Voronezhskogo instituta GPS MChS Rossii. 2014. № 2 (11). S. 46-51. eLIBRARY ID: 21574302
41. Bujnevich M.V., Izrailov K.E. Analiticheskoe modelirovanie raboty` programmnogo koda s uyazvimostyami // Voprosy` kiberbezopasnosti. 2020. № 3 (37). S. 2-12. DOI: 10.21681/2311-3456-2020-03-02-12
42. Izrailov K.E. Sistema kriteriev ocenki sposobov poiska uyazvimostej i metrika ponyatnosti predstavleniya programmnogo koda // Informatizaciya i svyaz`. 2017. № 3. S. 111-118. eLIBRARY ID: 29108491