Министерство образования Республики Беларусь

Учреждение образования
Белорусский государственный университет
информатики и радиоэлектроники

УДК 519.685.1

Рябинкин
Герман Максимович

Архитектура и дизайн объектно-ориентированных информационных систем при использовании парадигмы метапрограммирования

АВТОРЕФЕРАТ
диссертации на соискание степени магистра

по специальности 1-40 80 04 – Информатика и технологии программирования

Минск 2024

Работа выполнена на кафедре информатики учреждения образования «Белорусский государственный университет информатики и радиоэлектроники»

Научный руководитель: **Боброва Наталья Леонидовна,** кандидат технических наук, доцент, доцент кафедры информатики учреждения образования «Белорусский государственный университет информатики и радиоэлектроники»

Рецензент: **Нестеренков Сергей Николаевич,** кандидат технических наук, доцент, доцент кафедры программного обеспечения информационных технологий учреждения образования «Белорусский государственный университет информатики и радиоэлектроники»

Защита диссертации состоится «28» июня 2024 года в $10^{\underline{00}}$ часов на заседании Государственной экзаменационной комиссии по защите магистерских диссертаций в учреждении образования «Белорусский государственный университет информатики и радиоэлектроники» по адресу: 220013, Минск, ул. Платонова, 39, копр. 5, ауд. 308, тел. 293-85-91, e-mail: inform@bsuir.by.

С диссертацией можно ознакомиться в библиотеке учреждения образования «Белорусский государственный университет информатики и радиоэлектроники».

ВВЕДЕНИЕ

В современном мире разработки программного обеспечения архитектура приложения играет ключевую роль, определяя не только эффективность и масштабируемость создаваемых систем, но и их способность адаптироваться к постоянно меняющимся требованиям бизнеса и технологий. Архитектурные подходы, заложенные в основу многих современных приложений и широко применяющиеся и в настоящее время, были разработаны 10-20 лет назад, отражали существовавшие на тот момент тенденции и были, в том числе, обусловлены характерными техническими ограничениями зачастую неактуальными на сегодняшний день.

Однако, с течением времени, в условиях стремительного развития инфраструктуры и появления новых инструментов разработки, архитектура программного обеспечения претерпевает значительные изменения. Эта эволюция необходима для поддержания высокой производительности, безопасности и удобства использования систем в динамично развивающейся среде современных технологий.

Сегодня особенно актуальным становится вопрос адаптации архитектуры программного обеспечения к изменениям в окружении в свете внедрения в процесс разработки инструментов на основе больших языковых моделей и нейросетей. Эти инновационные технологии предлагают принципиально новые возможности для автоматизации труда программиста и оптимизации процессов разработки, что требует переосмысления традиционных архитектурных решений и может привести к радикальному изменению подходов к проектированию программного обеспечения.

Рост доступности аппаратного обеспечения, широкое распространение технических решений, задействующих облачную инфраструктуру, внедрение в разработку программного обеспечения практик и связанных с ними инструментов, меняющих подход к организации исходного кода приложения и процесса его написания – все это несомненно оказывает влияние на весь процесс создания и поддержки программ . Изменение этих факторов, кроме того, требует от программистов и пересмотра подходов к архитектуре программного обеспечения.

Архитектура программного обеспечения является важным практическим аспектом разработки, играющим одну из ключевых ролей в этом процессе. При этом не существует общепринятого подхода к сравнению эффективности различных архитектурных решений, равно как и согласия в том, какие именно характеристики системы можно отнести к первичным, определяющим ее архитектуру, а какие назвать вторичными.

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Цель и задачи исследования

Целью диссертационной работы является изучение и анализ изменения подходов к архитектуре и дизайну в условиях использования практик, характерных для парадигмы метапрограммирования.

Исходя из цели, в рамках работы были поставлены следующие *задачи*:

– определить подходы и практики в архитектуре программного обеспечения и дизайне объектно-ориентированных информационных систем, выделить факторы, влияющие на их развитие;

– выявить влияние используемой парадигмы программирования на архитектуру программного обеспечения;

– оценить перспективы использования генеративных языковых моделей при разработке программного обеспечения, в том числе в качестве средства поддержки парадигмы метапрограммирования;

– сформировать требования к перспективным инструментам разработки программного обеспечения и оценить их влияние на архитектуру и дизайн объектно-ориентированных информационных систем.

Объектом исследования является совокупность подходов и практик, составляющих архитектуру и дизайн объектно-ориентированных информационных систем.

Предметом исследования является влияние внешних факторов на изменение названых подходов и их трансформация при использовании парадигмы метапрограммирования.

Основной *гипотезой*, положенной в основу диссертационной работы, является утверждение, что использование парадигмы метапрограммирования способно в упростить процесс разработки и сопровождения программного обеспечения (на примере информационных систем), а также оказать значительное влияние не архитектуру разрабатываемого приложения.

Связь работы с приоритетными направлениями научных исследований и запросами реального сектора экономики

Сегодня особенно актуальным становится вопрос адаптации архитектуры программного обеспечения к изменениям в окружении в свете внедрения в процесс разработки инструментов на основе больших языковых моделей и нейросетей. Эти инновационные технологии предлагают принципиально новые возможности для автоматизации труда программиста и оптимизации процессов разработки, что требует переосмысления традиционных

архитектурных решений и может привести к радикальному изменению подходов к проектированию программного обеспечения.

Личный вклад соискателя

Основные результаты, изложенные в диссертации, получены лично соискателем. Вклад научного руководителя Н.Л. Бобровой заключается в постановке задач, участии в обсуждении методов решения и анализе полученных результатов.

Апробация результатов исследования

Результаты исследований, вошедшие в диссертацию, докладывались и обсуждались на международной научной конференции «Информационные технологии и системы» (г. Минск, Беларусь, 2022 г.), XI Международной научно-методической конференции «Высшее техническое образование : проблемы и пути развития» (г. Минск, Беларусь, 2022 г.), 59-й научно-технической конференции аспирантов, магистрантов и студентов БГУИР (г. Минск, Беларусь, 2023 г.).

Публикация результатов исследования

Основные результаты диссертации опубликованы в пяти печатных работах. В том числе статьях в сборниках трудов и материалов международных научных конференций и тезисах докладов на международных научных конференциях.

Структура и объем диссертации

Диссертация состоит из введения, общей характеристики работы, трех глав, заключения, библиографического списка.

В первой главе работы дается общая характеристика архитектуры объектно-ориентированных информационных систем и предлагается методика оценки сложности внесения изменений в архитектуру приложения.

Во второй главе предлагается классификация отношения языка и парадигмы программирования, рассматриваются особенности разработки информационных систем с использованием парадигмы метапрограммирования, влияние парадигмы программирования на архитектуру приложения.

В третей главе рассматривается влияние средств искусственного интеллекта на процесс разработки программного обеспечения, предлагаются требования к перспективным средствам автоматизации труда программиста.

Общий объем работы составил 66 страниц, их которых основного текста – 55 страниц, пять иллюстраций на двух страницах, библиографический список на пяти страницах из 49 именований, пять из которых – публикации соискателя.

ОСНОВНОЕ СОДЕРЖАНИЕ РАБОТЫ

В введении освещаются современное состояние и тенденции развития архитектуры программного обеспечения. Обозначаются проблемы, связанные с устареванием существующих практик ввиду изменения факторов, влияющих на архитектуру приложений: используемой инфраструктуры и аппаратного обеспечения, распространения технологий облачных вычислений, внедрения в процесс разработки инструментов на основе искусственного интеллекта в контексте использования парадигмы метапрограммирования. Обоснована актуальность темы работы, определены цели и задачи исследования.

В первой главе работы приводится общая характеристика архитектуры программного обеспечения и определяется понятие архитектуры. Рассматриваются основные элементы составляющие понятие и рассматривается их взаимосвязь.

Далее обосновывается актуальность проблемы внесения изменений в архитектуру приложения, для чего предлагается модель, описывающая изменение сложности добавления нового функционала в систему:

$$f_v(t) = \frac{a}{t+b} + c, \tag{1}$$

где a, b, c – некоторые постоянные коэффициенты, зависящие от достаточности или нехватки ресурсов в соответствии с потребностями, оптимальности используемых средств разработки и других внешних факторов,

t – время затраченное на разработку, прошедшее с начала проекта.

Кроме того, описывается и обосновывается зависимость роста количества компонентов приложения от времени при условии неизменности задействованных в процессе разработки ресурсов:

$$f_k(t) = a \cdot \ln(t+b) + ct + C. \tag{2}$$

Заключается, что полученная модель, описывающая рост количества компонентов приложения, а следовательно и рост объема исходного кода приложения и его функционала в зависимости от времени согласуется с результатами, наблюдаемыми на практике и позволяет применять результаты ее математического исследования к описываемому процессу реального мира. Так, исходя из полученной модели, можно достоверно утверждать, что не существует фундаментального предела роста объема приложения, т.к. $\lim\limits_{t\to\infty} f_k(t) = +\infty$.

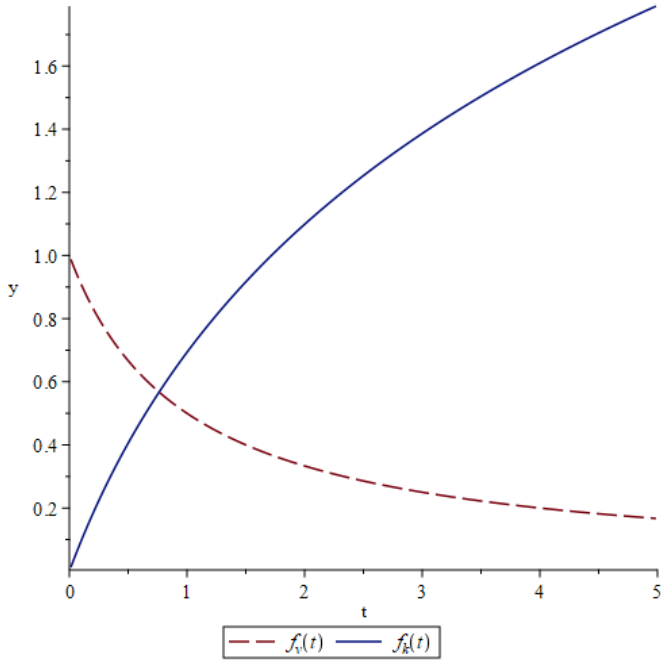

Рисунок 1 – Графики зависимости скорости добавления компонентов и роста количества компонентов от времени

Вместе с этим, ввиду неограниченного снижения значения функции (1) можно утверждать, что существует такой момент времени t_l, после которого скорость добавления новых компонентов (при условии сохранения доступных ресурсов) будет настолько мала ввиду высокой сложности расширения функционала приложения, что такая операция становится неоправданно затратной и на практике останавливает развитие системы.

Помимо добавления в систему новых компонентов, на практике часто возникает необходимость внесения изменений в уже существующие части приложения. Причиной для этого может стать изменение требований к системе, выявление ранее допущенных ошибок, в том числе на этапе проектирования, в ходе принятии архитектурных решений, при реализации различных компонентов и т.п. При этом, учитывая практическую неизбежность таких изменений, важной характеристикой системы становится возможность внесения в нее таковых за минимально допустимое время.

Далее в работе обосновывается возможность описания трудозатрат на внесение изменений в отдельный компонент как линейную зависимость от общего количества компонентов системы $f_{ck}(k)$:

$$f_{ck}(k) = a_{ck}k + b_{ck}. \quad (3)$$

Отсюда функция трудозатрат на переработку всех компонентов приложения $f_c(k)$ будет сложной функцией от k и $f_{ck}(k)$ (3), которая после ряда преобразований будет иметь следующий вид:

$$f_c(k) = ak^2 + bk + c. \quad (4)$$

Чтобы построить модель, выражающую зависимость сложности внесения изменений в архитектуру системы от времени $F(t)$, используются полученные выше формулы (2) и (4), в результате чего зависимость будет иметь вид:

$$F(t) = \ln(t+1)^2 + \ln(t+1). \quad (5)$$

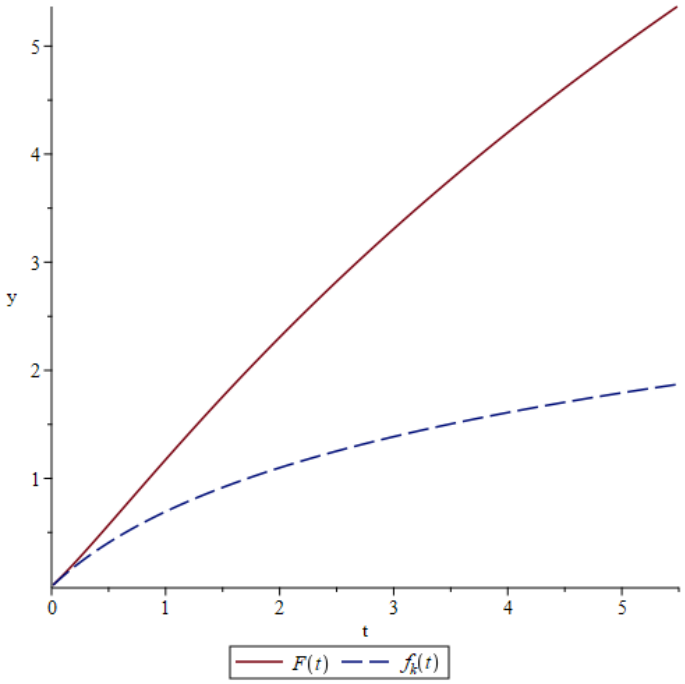

Рисунок 2 – График зависимости роста количества компонентов и сложности внесения изменений в архитектуру системы от времени

Из этого следует, что сложность внесения изменений в архитектуру приложения, представленная функцией более высокой степени, всегда будет расти опережающим темпом по отношению к росту самого приложения. Из чего можно сделать вывод, что некоторый момент времени t_{l1}, после которого задача внесения изменений в архитектуру приложения будет не достижима на практике наступит многим раньше чем рассмотренный ранее для функций (1) и (2) момент t_l, после которого становится практически невыполнимой задача добавления нового функционала в приложение. При этом, как многократно подтверждается практическими наблюдениям, момент времени t_{l1} для отдельных аспектов архитектуры для крупных корпоративных приложений наступает достаточно рано, и замкнутый период $(0, t_{l1})$ составляет незначительный отрезок в сопоставлении с периодом создания и поддержки системы.

Таким образом, можно говорить о принципиальной важности архитектуры приложения не только с точки зрения значимости определяемых ей аспектов системы, но и в контексте практической невозможности внесения изменений в архитектуру на значительном этапе жизни приложения $t_{l1} < t < t_l$. Это обстоятельство предъявляет высокие требования к архитекторам, и оправдывает пристальное внимание, уделяемое проблемам архитектуры в современной разработке программного обеспечения.

В заключительной части главы рассматривается процесс развития взглядов на архитектуру программного обеспечения в контексте изменения внешних факторов, оказывающих на нее влияние. В качестве наглядной демонстрации этого факта приводится процесс так называемой «облачной трансформации», связанной с ростом доступности инфраструктурных средств, аппаратного обеспечения, повышения вычислительных мощностей и распространения облачных вычислений.

Во второй главе рассматривается влияние используемой парадигмы программирования на архитектуру программного обеспечения и процесс его разработки. В первом разделе обосновывается утверждение о возможности переключения между парадигмами в рамках одного языка программирования, а также о возможности расширения возможностей языка программирования за счет внесения в него средств поддержки сторонних парадигм. Предлагаются основания для классификация отношения языка программирования к той или иной парадигме.

В качестве подтверждения названных идей описывается практика использования объектно-ориентированной парадигмы в языке программирования Си, а также приводятся примеры применения нестандартных для используемых языков парадигм, при написании исходного кода приложения как на языках ассемблера, так и при использовании высокоуровневых языков, таких как Java.

Далее описывается влияние, оказанное внесением средств поддержки парадигмы функционального программирования в язык Java на используемы при создании бизнес-ориентированных приложений архитектурные стили. Из представленных примеров следует наличие устойчивой связи между архитектурой приложения и используемыми при его разработке парадигмами программирования.

В третьем разделе подробно рассматривается парадигма метапрограммирования, приводятся основные подходы к ее использованию в различных языках. Отмечается ее влияние не только на архитектуру приложения, но и на принципиальный характер зависимости (5), описанной в первой главе. Для обоснования этого факта подробно рассматривается случай использования парадигмы метапрограммирования при разработке

информационной системы с веб-интерфейсом, реализованной в объектно-ориентированном стиле с помощью высокоуровневого языка программирования.

В частности, описывается проблема избыточности информации в исходном коде таких систем, которую можно описать следующей моделью. Пусть матрица D размера $m \times 1$, представляет собой определения объектов доменной модели:

$$D = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_{m-1} \\ d_m \end{bmatrix}. \quad (6)$$

При чем каждый элемент матрицы D, является определением некоторого объекта доменной модели. В свою очередь, матрица K размера $1 \times n$, представляет собой метаинформацию с определением структуры функциональных компонентов системы:

$$K = [k_1 \quad k_2 \quad \cdots \quad k_{n-1} \quad k_n]. \quad (7)$$

При чем каждый элемент матрицы K, является определением некоторого функционального компонента системы. В таком случае, все компоненты системы, можно представить в виде элементов произведения матриц $M = D \times K$:

$$M = D \times K = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_{m-1} \\ d_m \end{bmatrix} \times [k_1 \quad k_2 \quad \cdots \quad k_{n-1} \quad k_n] =$$

$$= \begin{bmatrix} d_1k_1 & d_1k_2 & \cdots & d_1k_{n-1} & d_1k_n \\ d_2k_1 & d_2k_2 & \cdots & d_2k_{n-1} & d_2k_n \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ d_{m-1}k_1 & d_{m-1}k_2 & \cdots & d_{m-1}k_{n-1} & d_{m-1}k_n \\ d_mk_1 & d_mk_2 & \cdots & d_mk_{n-1} & d_mk_n \end{bmatrix}. \quad (8)$$

При этом информацию, содержащуюся в каждом компоненте системы m_{ij} (8), можно определить как производную информацию от определения некоторого объекта d_i (6) и определения некоторого компонента k_j (7).

Таким образом, для любой информационной системы очевидна избыточность информации, определяющей каждый из ее компонентов по отдельности. В случае, если разработка приложения ведется в парадигме

объектно-ориентированного программирования, без применения средств кодогенерации и парадигмы метапрограммирования, определение некоторого компонента k_j, может быть выражено как в виде технической документации, так и в виде соглашения между программистами, участвующими в разработке приложения. При этом внесение изменений в архитектуру приложения можно трактовать как внесение изменений в определение компонента k_j, что, в свою очередь, приведет к необходимости внесения изменения как минимум в m компонентов, определение которых зависит от k_j, где m – количество строк матрицы D, соответствующее числу реализованных в системе объектов доменной модели. Для данного случая остаются справедливыми зависимости $f_k(t)$ и $F(t)$ описанные формулами (2) и (5) соответственно.

В случае, когда в рассматриваемой системе используются инструменты кодогенерации и применяется парадигма метапрограммирования, в системе должны быть реализованы генераторы кода, которые будут создавать шаблонный код на основе некоторых исходных данных. При этом, возможно создание отдельного генератора для каждого из функциональных компонентов k_j. В таком случае, реализация генератора для отдельного функционального компонента может считаться определением этого компонента, а определение объекта доменной модели d_i будет выступать в роли входных данных для этого генератора.

Кроме очевидной экономии времени после этапа реализации кодогенераторов, главным преимуществом такого подхода является простота внесения изменений в архитектуру системы. Так, пусть в ходе разработки приложения возникла необходимость изменить порядок работы с базой данных. В принятой выше терминологии это означает, что будут внесены изменения в определение некоторого функционального компонента k_j, отвечающего за выполнение подлежащих изменению операций. Для внесения этих изменений в систему, программисту нет необходимости вносить изменения в m компонентов, определение которых зависит от k_j, а достаточно модифицировать исходный код генератора, представляющего собой определение k_j. То есть независимо от размеров системы и количества объектов доменной модели, которое имеет тенденцию к росту на протяжении всего жизненного цикла приложения, затраты на внесение изменений в архитектуру системы будет принимать постоянное значение. Таким образом, зависимость сложности внесения изменений в архитектуру системы от времени при использовании парадигмы метапрограммирования можно выразить как:

$$F_m(t) = a_m. \tag{9}$$

Где a_m, является некоторой константой. Очевидно, что при любом значении a_m, будет существовать момент времени t_m, после которого будет выполняться равенство:

$$F_m(t) < F(t). \tag{10}$$

При этом, предел разности значений этих функций будет стремиться к бесконечности при неограниченном удалении значении t вправо вдоль оси абсцисс.

$$\lim_{t \to +\infty} \big(F(t) - F_m(t)\big) = +\infty. \tag{11}$$

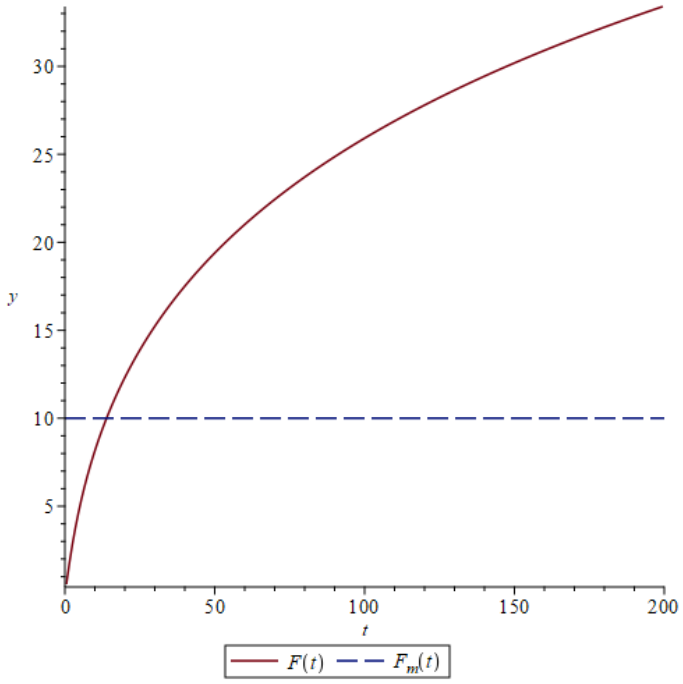

Рисунок 3 – Графики зависимости сложности внесения изменений в архитектуру от времени для систем, разрабатываемых с использованием разных парадигм программирования

Таким образом, использование парадигмы метапрограммирования решает проблему роста сложности внесения изменений в архитектуру приложения с ростом количества компонентов приложения, и, в таком случае, не наступает описанный выше момент времени t_{l1}, когда внесение изменений в некоторые аспекты архитектуры системы на практике становится невыполнимым ввиду нецелесообразности затраты ресурсов на такие изменения.

Вместе с этим, реализация кодогенераторов для нужд конкретного приложения является нетривиальной задачей, также, учитывая уникальность совокупности используемых технологий, архитектурных решений, принципов и структуры для каждого приложения, не представляется возможным

реализовать универсальный фреймворк кодогенерации средствами классического программирования.

Данная задача должна решаться отдельно в рамках каждого проекта, и отсутствие возможности унифицировать процесс кодогенерации приемлемым для задач промышленного программирования образом является основным препятствием на пути к массовой адаптации этого подхода, несмотря на очевидные преимущества его использования в контексте построения объектно-ориентированных информационных систем.

Обозначенная проблема может быть решена с внедрением инструментов кодогенерации, основанных на моделях глубокого обучения работающих с естественным языком. В данном случае, сохраняются все преимущества использования парадигмы метапрограммирования и инструментов кодогенерации, и одновременно исчезает необходимость трудоемкой реализации кодогенераторов под нужды каждого конкретного приложения.

В третей главе рассматривается влияние средств искусственного интеллекта на процесс разработки программного обеспечения, а также предлагаются требования к перспективным средствам автоматизации труда программиста в контексте рассмотренного в предыдущей главе подхода к организации исходного кода приложения.

Проводится анализ внедрения в разработку программного обеспечения инструментов, основанных на искусственном интеллекте. Выделяется два основных подхода к данному процессу: интеграция названных систем в разрабатываемые приложения для обогащения их функционала, а также использование искусственного интеллекта для автоматизации труда программиста.

Интеграция систем искусственного интеллекта в приложения с целью расширения их функциональности вызывает ряд трудностей, вызванных, главным образом, динамическим характером и высоким темпом развития таких систем. Это накладывает определенные требования на приложение, в частности, возможность в кратчайшие сроки и с минимальными затратами адаптировать его к изминающейся среде. Данные требования отражаются на выборе используемых при разработке приложения архитектурных подходов.

Активно развиваются инструменты машинного обучения, предназначенные для автоматизации труда программиста. Перспективные пути их использования при разработке программного обеспечения предполагают в том числе изменение преобладающей парадигмы программирования. Данное обстоятельство способно в значительной степени оказать влияние на доминирующе взгляды на построение архитектуры программного обеспечения.

Исходя из результатов, полученных в первой и второй главах работы заключается, что на настоящем этапе видится перспективным вариант использования инструментов машинного обучения для решения проблем внедрения и распространения парадигмы метапрограммирования. Рассматриваемый подход позволяет наиболее эффективным образом задействовать возможности языковых моделей на основе нейронных сетей и оптимизировать процесс разработки программного обеспечения.

Вместе с этим, текущее состоящие инструментов кодогенерации на основе нейросетей и языковых моделей не позволяет в полной мере задействовать их в рамках парадигмы метапрограммирования. На основе проведенного выше анализа, к перспективным средствам кодогенерации, для возможности их интеграции в процесс разработки программного обеспечения в качестве средств поддержки метапрограммирования, выдвигаются следующие требования:

– качество работы, заключающееся в возможность генерировать компоненты приложения высокой сложности и объема, достигая при этом оптимальных показателей производительности полученного кода и достаточного уровня безопасности (отсутствия уязвимостей);

– стабильность работы, заключающаяся в достижении одного и того же результата при получении одних и тех же входных данных;

– прозрачность работы, заключающая в понятном механизме описания требований для генерации компонента и предсказуемом результате.

Далее в работе проводится анализ каждого из требований, оценивается текущее соответствие доступных систем кодогнерации на основе искусственного интеллекта названным требованиям, актуальные проблемы в рамках их выполнения, а также возможные пути решения этих проблем.

ЗАКЛЮЧЕНИЕ

Основные научные результаты диссертации

1 Выполнен анализ подходов и сложившихся практик построения архитектуры программного обеспечения и дизайна объектно-ориентированных информационных систем. Выявлены основные факторы, влияющие на их развитие. Выявлены характер и степень влияния парадигмы программирования на архитектуру.

2 Установлен характер функциональных зависимостей сложности внесения изменений в архитектуру приложения и добавления новых компонентов от различных факторов. Предложена методика сравнения гибкости архитектуры на основе анализа установленных зависимостей.

3 Выполнен анализ перспективы использования генеративных языковых моделей при разработке программного обеспечения. Рассмотрены возможные варианты интеграции таких систем в процесс разработки и степень их влияния на архитектуру и дизайн объектно-ориентированных информационных систем. Сформированы требования к перспективным инструментам разработки программного обеспечения.

Рекомендации по практическому использованию результатов

Полученные результаты формируют теоретическую базу для адаптации подходов к архитектуре программного обеспечения в зависимости от изменения используемой инфраструктуры, доступных ресурсов, используемых при разработке парадигм программирования и других факторов. Они могут быть использованы для оценки принятых решений и поддержки их принятия, касающихся дизайна и внутренней организации новых и внесения изменений в существующие программные средства.

СПИСОК ПУБЛИКАЦИЙ СОИСКАТЕЛЯ

1. Rabinkin, H. M. Influence of software and hardware infrastructure on architectural styles in software development / H. M. Rabinkin // Актуальные вопросы экономики и информационных технологий : сборник тезисов и статей докладов 59-ой научной конференции аспирантов, магистрантов и студентов БГУИР, Минск, 17–21 апреля 2023 г. / Белорусский государственный университет информатики и радиоэлектроники. – Минск, 2023. – С. 355–359.

2. Деренчук, В. И. Использование чат-ботов для построения персонализированной траектории обучения = Using chatbots to build a personalized learning path / В. И. Деренчук, Г. М. Рябинкин // Высшее техническое образование : проблемы и пути развития = Engineering education: challenges and developments : материалы XI Международной научно-методической конференции, Минск, 24 ноября 2022 года / Министерство образования Республики Беларусь, Белорусский государственный университет информатики и радиоэлектроники. – Минск : БГУИР, 2022. – С. 58–64.

3. Деренчук, В. И. Классификация данных прецизионной моделью нейронной сети = Data classification with a precision neural network model / В. И. Деренчук, С. В. Болтак, Г. М. Рябинкин // Компьютерные системы и сети : сборник статей 59-й научной конференции аспирантов, магистрантов и студентов, Минск, 17–21 апреля 2023 г. / Белорусский государственный университет информатики и радиоэлектроники. – Минск, 2023. – С. 156–159.

4. Рябинкин, Г. М. Использование проблемно-эвристического подхода при обучении программированию = Using the problem-based heuristic approach in teaching of programming / Г. М. Рябинкин, С. В. Болтак // Высшее техническое образование : проблемы и пути развития = Engineering education: challenges and developments : материалы XI Международной научно-методической конференции, Минск, 24 ноября 2022 года / Министерство образования Республики Беларусь, Белорусский государственный университет информатики и радиоэлектроники. – Минск : БГУИР, 2022. – С. 148–153.

5. Рябинкин, Г. М. Расширение возможностей языка программирования за счёт добавления средств поддержки различных парадигм программирования / Г. М. Рябинкин, В. И. Деренчук // Информационные технологии и системы 2022 (ИТС 2022) = Information Technologies and Systems 2022 (ITS 2022) : материалы Международной научной конференции, Минск, 23 ноября 2022 / Белорусский государственный университет информатики и радиоэлектроники ; редкол.: Л. Ю. Шилин [и др.]. – Минск : БГУИР, 2022. – С. 41–42.