УДК 004.853

Применение подхода «архитектура как код» при формировании крупных экосистем

К. А. Терещенко, Р. В. Мальчева
Донецкий национальный технический университет
E-mail: raisa.malcheva@yandex.ru

Аннотация
Рассмотрены основные методологии реализации процесса развития проекта. Сформулированы проблемы контроля и развития системной архитектуры в рамках основных рассматриваемых методологий. Рассмотрен новый подход к реализации артефактов системной архитектуры в процессе проектной работы. В результате анализа сделан вывод, что для крупных систем внедрение отдельной команды и набор соответствующих специалистов является менее ресурсозатратной задачей, чем внедрение в каждую команду отдельного специалиста по архитектуре или разрешение постоянно всплывающих инцидентов.

Введение

При формировании крупных экосистем актуальной проблемой является проектирование их эффективной архитектуры с учетом таких ключевых требований как масштабируемость, открытость, неоднородность, безопасность, доступность, и производительность [1-5]. В [6] выполнен анализ особенностей и области применения шаблонов системной архитектуры с точки зрения эффективности их применения в развернутых экосистемах.

Целью данной статьи является анализ основных методологий реализации процесса развития проекта и рассмотрение нового подхода к реализации артефактов системной архитектуры в процессе проектной работы.

Общая постановка проблемы

Формирование крупной IT экосистемы - не тривиальная задача. Для того, чтобы привлечь пользователей, современные компании пытаются не просто предоставить им набор разнообразных обособленных продуктов, а сформировать совокупный ландшафт интегрированных друг с другом решений. Это делается с целью повышения степени автоматизации сопряженных друг с другом процессов, улучшения отчетности, снижения нагрузки на пользователя и сохранения его в экономическом контуре компании. В итоге рождается задача контроля, необходимая для выполнения условий функционирования, масштабирования и развития множества связанных между собой проектов. Также возникает набор новых ролей, одной из которых является роль архитектора систем.

Системный архитектор — это специалист, который отвечает за создание и разработку общей структуры и дизайна информационной системы или программного продукта. Он определяет ключевые компоненты системы, их взаимодействие и общую архитектуру, учитывая требования заказчика, бизнес-процессы и технические аспекты. Роль системного архитектора включает определение архитектурных решений, разработку технических спецификаций, управление техническими рисками и обеспечение соответствия архитектуры стандартам и требованиям проекта.

Ресурсы, выделяемые архитекторам на работу с проектом, тяжело планировать и балансировать. В крупных компаниях, где существует значительное число команд, постоянное согласование изменений с архитектором может существенно замедлить процесс разработки. В свою очередь, увеличение числа архитекторов, напротив, может создать ситуацию простаивания ресурсов, так как архитектор участвует лишь в некоторых этапах создания системы (планирование, согласование архитектурно значимых изменений).

В отличие от процесса разработки, где все построено на коде, архитектура системы требует создания графических артефактов, для которых используются визуальные редакторы.

Объединение версий диаграмм, актуализация данных, плановые изменения расположенных в документации решений часто приводит к значительным затратам времени, так как даже определение расположения всех связанных с проектом артефактов в масштабных системах становится комплексной и развернутой задачей. Тратятся значительные ресурсы на механическую и бюрократическую деятельность, что может стать причиной возникновения неактуальной документации, замедления процессов согласования, расхождения в понимании концептов реализации

взаимодействия систем и сервисов в разных командах этих систем, и, как следствие, приведет к появлению ошибок в интеграционном взаимодействии и возникновению инцидентов.

Анализ существующих методологий

Существуют разные методологи для решения вышеописанной проблемы. Использование одного из популярных концептов ведения разработки потенциально позволяет организовывать процесс более эффективным образом, что решает множество проблем, таких как возникновение неактуальной документации, расхождения в системных контрактах взаимодействия и других, связанных с реализацией системной архитектуры.

Можно выделить несколько основных металогических подходов к разработке.

Agile-методологии (рисунок 1) - семейство методологий управления проектами и разработки программного обеспечения, которые основаны на принципах Agile Manifesto [7]. Они предполагают итеративное и инкрементальное развитие продукта, постоянное взаимодействие с заказчиком, гибкое реагирование на изменения в требованиях и фокус на командной работе. Применение данного шаблона могло бы позволить сократить объем концентрации ответственности на системном архитекторе, распределяя ее по проектным командам, которые берут на себя часть функций в рамках производственного цикла.

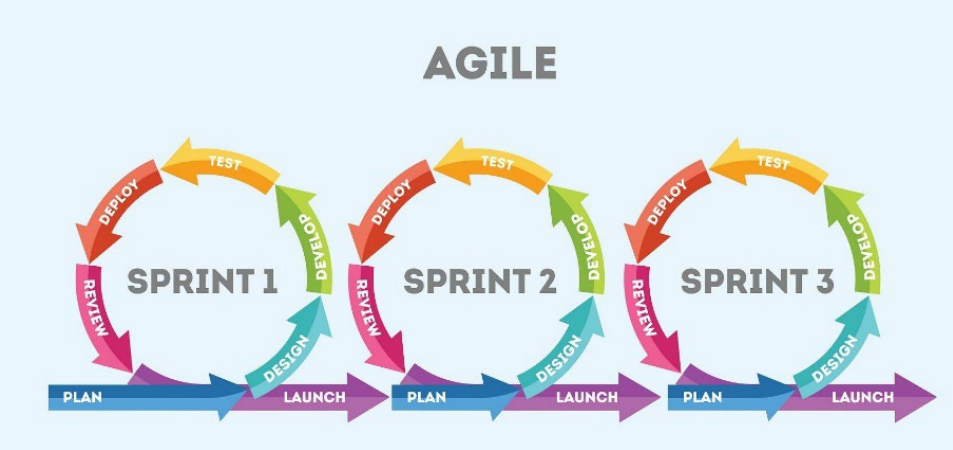

Рисунок 1 – Agile-методологии

Рассмотрим некоторые из проблем применения подхода в крупных компаниях.

Необходимость стандартизации. В крупных корпорациях часто имеются строгие архитектурные стандарты и процедуры, которым должны следовать все проекты. Это касается как применяемых технологий с целью унификации подхода к мониторингу и обмену данными, например, внедрения определенных брокеров сообщений или стандартизированных API интерфейсов, так и формирования конкретизированных архитектурных шаблонов, предназначенных, например, для ограничения хранения критически важных данных в полном множестве проектов. Agile-методологии, напротив, поощряют гибкость и адаптацию к изменяющимся требованиям. Это может привести к конфликтам со стандартами и процедурами корпорации и дальнейшим проблемам.

Масштабирование. Agile-методологии изначально были разработаны для небольших команд, что может вызвать сложности для масштабирования их до уровня системной архитектуры крупной корпорации. При значительном числе штаба разбиение на малые команды может затруднить коммуникацию, а расширение состава команд стать препятствием к реализации уже самого манифеста.

Waterfall-модель (рисунок 2) [8]. Это традиционный подход к управлению проектами, который предполагает последовательное выполнение этапов проекта: анализ требований, проектирование, реализацию, тестирование и внедрение. Этот подход хорошо подходит для проектов с четко определенными требованиями и стабильной средой разработки. Несмотря на то, что waterfall модель в определенной степени решает проблемы Agile методологии, отсутствие гибкости порождает ряд новых.

*Высокая стоимость ошибок. Е*сли ошибки были допущены на ранних стадиях проекта, то исправление их может занять много времени и ресурсов, так как waterfall не предполагает возвращение к ранним этапам. Это особенно болезненно для крупных и развернутых систем, где цена простоя крайне высока.

Проблемы с коммуникацией. При использовании Waterfall-модели необходимо четко определить все требования к проекту на начальном этапе.

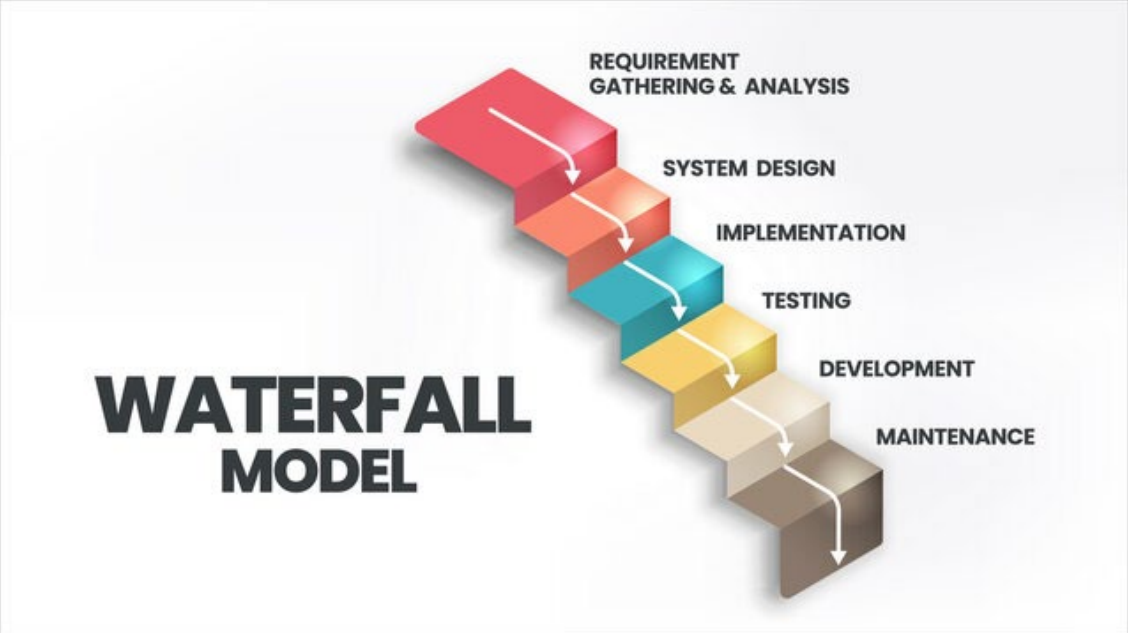

Рисунок 2 – Waterfall-модель

Однако, в крупных проектах часто возникают сложности с коммуникацией между различными отделами и заинтересованными сторонами, что нередко приводит к рассинхронизации данных, нарушению системных контрактов и запозданию в локализации ошибок.

Model-driven engineering (MDE) [9]. Данная методология в области разработки программного обеспечения основывается на использовании моделей для проектирования и создания систем. В своей структуре она включает формирование необходимых артефактов системной архитектуры. Более того, в MDE модели играют центральную роль, представляя систему и ее компоненты на различных уровнях абстракции.

Разработчики используют модели для автоматизации процессов создания кода, тестирования, документирования и сопровождения ПО, что может решить ряд проблем с коммуникацией, так как мутабельность моделей автоматизируема и позволяет отслеживать все необходимые изменения в взаимосвязанных технически артефактах документации.

Тем не менее данный подход тяжело применим, так как:

- требует использования или разработки специализированных инструментов автоматизации формирования артефактов;
- предлагает лишь набор стандартизированных принципов, а не конкретные решения, что в контексте применения автоматизации и моделирования, по сути, ставит перед проектом задачу реализации подхода.

Это привело к разработки его модификаций.

Архитектура как код

Данная архитектура фактически является локализацией **MDE,** определяющей более конкретный процесс создания архитектурных артефактов системы, который потенциально может быть расширен на весь процесс разработки.

Концептуально «архитектура как код» сопоставляет процесс формирования моделей, системных контрактов и другой документации с процессами внедрения изменений в рабочий код компании [10].

Артефакты создаются путем написания кода с использованием определенных инструментов, позволяющих это делать, например, PlantUML.

Затем процесс фиксируется через инструмент контроля версий, например Git, в общем «репозитории» проекта.

В случае внесения изменений, автор создаёт отдельную «ветку», в которую вносит необходимые доработки (рисунок 3) и отправляет ее «на слияние» (рисунок 4) с корневым проектом.

Ответственные за архитектуру проверяют доработки и согласовывают их, либо возвращают автору с комментариями, какие дополнительные изменения требуется произвести.

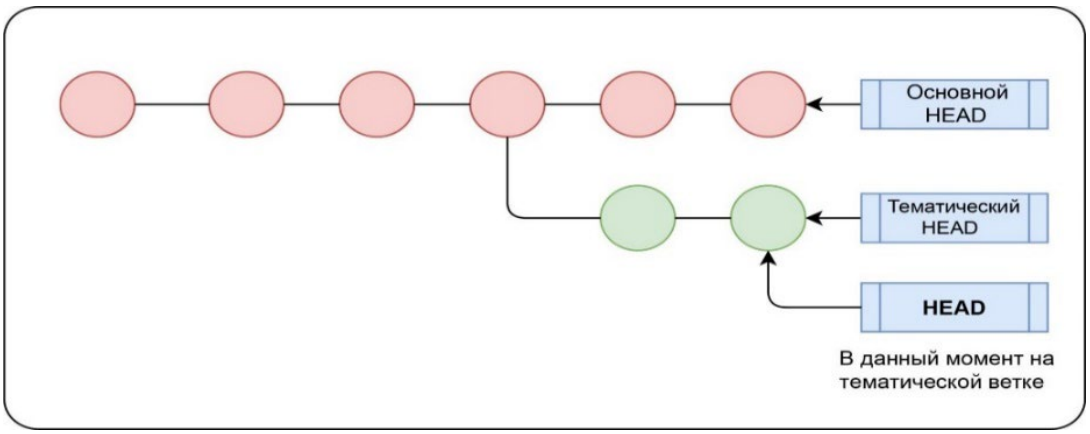

Рисунок 3 - Создание тематической ветки (HEAD)

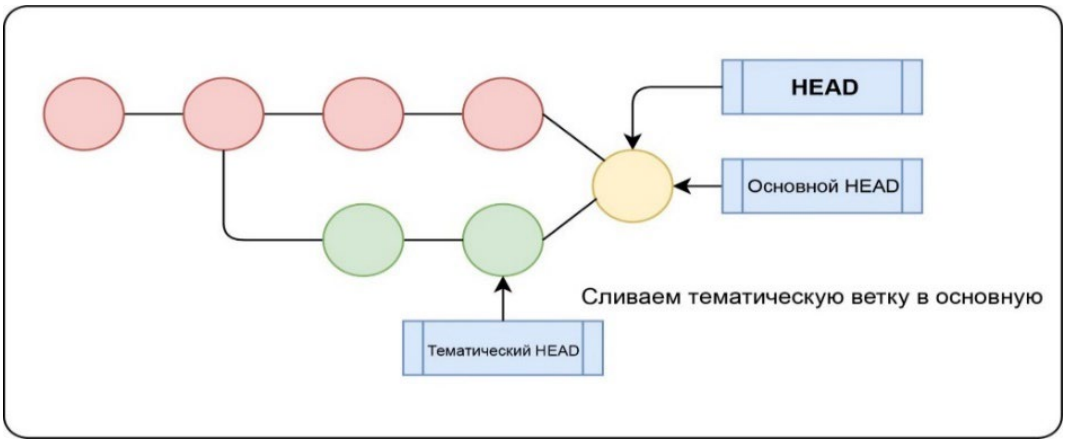

Рисунок 4 - Слияние веток

Помимо внутренней ядра проекта есть корпоративный центральный репозиторий (рисунок 5), который содержит концептуальную архитектуру, доступную всем для ознакомления с планами развития.

Изменения в репозиторий вносятся также через «Pull requests», а визуализация происходит средствами какого-либо инструмента, например, DocHub.

Далее изменения рассматривают управляющие проектами. Это способствует обмену опытом, установке эффективного взаимодействия и общему пониманию архитектурных решений.

Описание текущей архитектуры хранится в отдельных командах, что дает им возможность автономно принимать быстрые решения.

Централизованный репозиторий размещен рядом с командами, позволяя им оперативно вносить обоснованные изменения в продукт, и, в случае необходимости, фиксирует отклонения как инциденты.

Выводы

Реализация концепции «архитектура как код» имеет преимущества, которые частично разрешают противоречия, возникающие при применении популярных методологий Agile и Waterfall, а также дают локализацию и конкретику в применении методологии MDE.

Применение подхода накладывает определенные требования, например, дополнительными компетенциями на соответствующих участников процесса, использование отдельной системы для контроля системной архитектуры и другие.

В результате анализа сделан вывод, что для крупных систем внедрение отдельной команды и набор соответствующих специалистов является менее ресурсозатратной задачей, чем внедрение в каждую команду отдельного специалиста по архитектуре или разрешение постоянно всплывающих инцидентов.

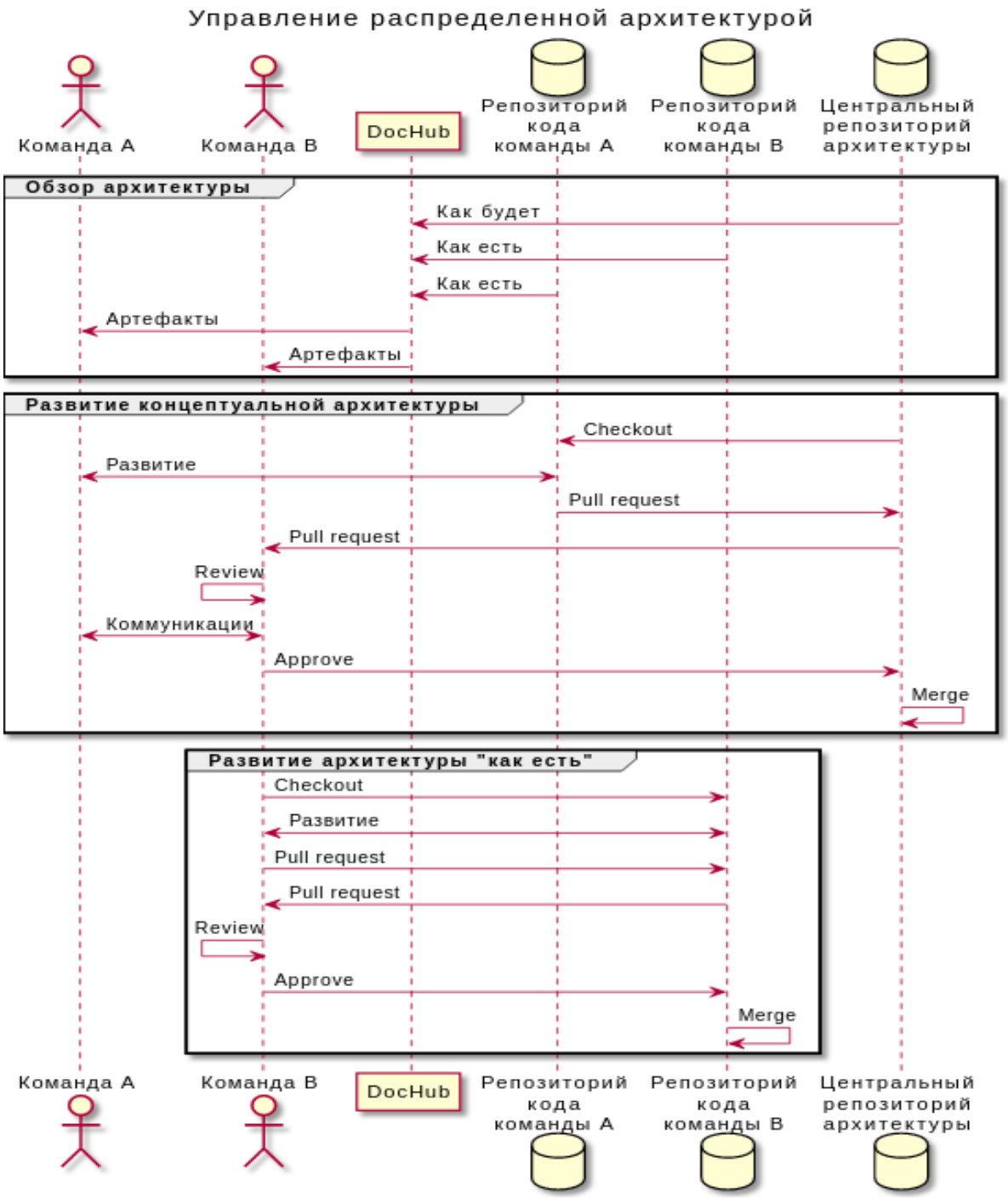

Рисунок 5 – Управление распределенной архитектурой

Литература

1. Мальчева, Р. В. Компьютерные технологии – основа цифровой экономики // Бизнес-инжиниринг сложных систем: модели, технологии, инновации. Сборник материалов III международной научно-практической конференции. – Донецк: ДОННТУ, 2018. - С. 102-105.

2. Паньшин, Б. Цифровая экономика: особенности и тенденции развития [Электронный ресурс] / Б. Паньшин. – Режим доступа: https://cyberleninka.ru/article/n/tsifrovaya-ekonomika-osobennosti-i-tendentsii-razvitiya

3. Мальчева, Р. В. Проектирование архитектуры системы электронных денег / Р. В. Мальчева, К. А. Терещенко // Информатика и кибернетика. – Донецк: ДонНТУ, 2023. - № 1(31). – С. 23-29.

4. Терещенко, К. А. Проектирование распределенной архитектуры компьютерной сети банка / К. А. Терещенко, Р. В. Мальчева // Информационное пространство Донбасса: проблемы и перспективы : материалы V Респ. С междунар. Участием науч.-практ. Конф., 27 окт.

2022 г. – Донецк : ГОУ ВПО «ДонНУЭТ», 2022. – С. 131 -134.

5. Терещенко, К. А. Анализ особенностей функционирования и развития объектов и процессов компьютерной сети банка / К. А. Терещенко, Р. В. Мальчева // Информатика, управляющие системы, математическое и компьютерное моделирование (ИУСМКМ-2022): Материалы XIII Международной научно-технической конференции в рамках VIII Международного Научного форума Донецкой Народной Республики. Донецк, 2022. – Донецк: ДонНТУ, 2022. – С.392-295.

6. Терещенко К. А. Анализ специфики и области применения архитектурных шаблонов в развернутых экосистемах / К. А. Терещенко, Р. В. Мальчева // Информатика и кибернетика. – Донецк: ДонНТУ, 2023. - № 4(34). – С. 49-59.

7. Agile: что это такое и где используется, принципы методологии [Электронный ресурс]. – Режим доступа: https://practicum.yandex.ru/blog/metodology-agile/

8. Барулина, В. В. Сравнительный анализ гибкой и каскадной методологии разработки программного обеспечения [Электронныйресурс] / В. В. Барулина. – Режим доступа: https://cyberleninka.ru/article/n/sravnitelnyy-analiz-gibkoy-i-kaskadnoy-metodologii-razrabotki-programmnogo-obespecheniya/viewer

9. Alberto Rodrigues da Silva. Model-Driven Engineering: A Survey Supported by a Unified Conceptual Model [Электронный ресурс] / INESC-ID / Instituto Superior Técnico, Universidade de Lisboa. – Режим доступа: http://isg.inesc-id.pt/alb/static/papers/2015/r13-as-2015-COMLAN-v2.1b.pdf

10. Brown, S. A minimal approach to software architecture documentation [Электронный ресурс] / S. Brown. – Режим доступа: https://dev.to/simonbrown/a-minimal-approach-to-software-architecture-documentation-4k6k

Терещенко К.А., Мальчева Р.В. Применение подхода «архитектура как код» при формировании крупных экосистем. *Рассмотрены основные методологии реализации процесса развития проекта. Сформулированы проблемы контроля и развития системной архитектуры в рамках основных рассматриваемых методологий. Рассмотрен новый подход к реализации артефактов системной архитектуры в процессе проектной работы. В результате анализа сделан вывод, что для крупных систем внедрение отдельной команды и набор соответствующих специалистов является менее ресурсозатратной задачей, чем внедрение в каждую команду отдельного специалиста по архитектуре или разрешение постоянно всплывающих инцидентов.*

Ключевые слова: *экоситема, шаблон, архитектура, критерии, система оценок*

Tereshchenko K.A., Malcheva R.V. The application of the "architecture as code" approach in the formation of large ecosystems. *The main methodologies for the implementation of the project development process are considered. The problems of control and development of the system architecture within the framework of the main methodologies under consideration are formulated. A new approach to the implementation of system architecture artifacts in the process of project work is considered. As a result of the analysis, it was concluded that for large systems, the introduction of a separate team and the recruitment of appropriate specialists is a less resource-intensive task than the introduction of a separate architecture specialist into each team or the resolution of constantly emerging incidents.*

Key words: *ecosystem, pattern, architecture, criteria, evaluation system*

Статья поступила в редакцию 15.04.2024
Рекомендована к публикации профессором Зори С. А.