ISSN 1992-6502 (Print) Вестник УГАТУ ISSN 2225-2789 (Online)
2026. Т. 30, № 1 (111). С. 65-74 http://journal.uust.ru/index.php/Vestnik

УДК 004.41 doi 10.54708/19926502_2026_30111165
Код ГРНТИ 55.05.19

Графовые модели и операции при оценке микросервисных решений

Э.М. Чембарисов*, О.Н. Сметанина, Е.Ю. Сазонова

ФГБОУ ВО «Уфимский университет науки и технологий» (УУНиТ), г. Уфа, Россия

Аннотация. Архитектура микросервисов сыграла важную роль в качестве доминирующего фактора в различных разработках программных систем. Она обеспечивает масштабируемость и возможность обслуживания. Возрастающая сложность систем микросервисов сталкивается со значительными проблемами в понимании и оценке. В статье предлагается формальное представление архитектуры микросервисов с использованием теории графов, где каждый микросервис представлен как вершина, а каждая связь или зависимость между двумя микросервисами – направленное ребро. Введена математическая модель, основанная на матрицах смежности, исследованы алгоритмы теории графов. Изучены метрики графа для анализа структурных свойств, поиска критических узлов и обнаружения потенциальных точек отказа. Также исследуется ряд операций, такие как анализ пути, поиск и обнаружение циклов и оценка связности. Указанные операции важны для тестирования, оценки надежности и отказоустойчивости в распределенных системах. Предлагаемый подход тестируется на примере гипотетической системы микросервисов. Рекомендации предполагают, что теоретико-графовые модели обеспечивают основу для формального анализа, мониторинга и оптимизации архитектур микросервисов.

Ключевые слова: *архитектура микросервисов, теория графов, матрица смежности, вершины, ребра*.

***lawluker@gmail.com**

Введение

Повышение сложности программных решений при расширении спектра цифровизации деятельности человека связано с изменениями в технологической сфере и возрастающими запросами пользователей. Необходимость трансформации IT ландшафта на предприятиях возникает при переходе на отечественное программное обеспечение, потребности в наращивании функционала имеющихся систем или их интеграции. Немаловажным становится подход к разработке программных решений на основе микросервисной архитектуры, позволяющей поочередно вводить новые модули. Это обеспечивает возможность обновления ряда блоков, масштабируемость системы и др.

Пользу и специфику микросервисной архитектуры для создания программных решений отмечают многие специалисты [1–10]. Авторы статьи [1] определяют микросервисы как архитектурные и организационные подходы к разработке программного обеспечения (ПО), представленного небольшими (независимыми или связанными) сервисами или системами по выполнению определенных функций, взаимодействующих через четко определенные API.

В статье [2] авторами отмечены такие особенности микросервисной архитектуры (MSA), как упрощение масштабирования и ускорение разработки приложений с возможностью внедрения инноваций и вывода новых функций на рынок. Также приведены примеры организаций (Amazon, Netflix и Guardian), использующих MSA для разработки и непрерывной поставки больших и сложных приложений, обеспечивающих гибкость и разнообразие технологического стека. Помимо возможности структурирования разработки систем, принципы проектирования микросервисной архитектуры используются для миграции систем с традиционным архитектурным стилем в MSA.

Несмотря на довольно широкое распространение MSA при увеличении масштабов и сложности создаваемых систем, например веб-сервисов, возникает потребность в эффективных

способах моделирования их структуры, учета взаимодействия между микросервисами и возможного влияния одних микросервисов на другие.

В статье описана проблема взаимосвязи между микросервисами, возможного влияния одних компонентов системы на другие, современное состояние проблемы, предлагаемое решение на основе теоретико-графового подхода, проведение эксперимента на примере вебсервиса, анализ архитектурных решений, формализованных на основе теории графов, и обсуждение полученных результатов.

Современное состояние проблемы

Актуальность данного исследования заключается в том, что комплекс взаимосвязей и взаимозависимостей между компонентами системы, построенной на микросервисной архитектуре, может вызвать трудности при выявлении зависимостей между микросервисами при оценке влияния изменений одного микросервиса на другие, при выявлении неисправностей и др. Прежде всего, взаимосвязанность и взаимозависимость микросервисов может привести к неконтролируемой передаче данных, искажениям информации и, как следствие, к нарушениям целостности функционирования всей системы. Отсутствие формального и систематизированного представления таких взаимосвязей существенно затрудняет анализ общей архитектуры системы и выявление узких мест.

Авторы статьи [3, 14] исследуют приложения микросервисной архитектуры и возможность появления архитектурных антишаблонов при ее проектировании. Для выявления такого рода проблем проведен подробный анализ архитектуры, в рамках которого были применены концепции и алгоритмы теории графов. Авторами предложено инструментальное решение по использованию трассируемых данных системы для построения графов зависимостей и последующего извлечения метрик, отражающих потенциальные недостатки архитектуры. Предложенное решение позволило проанализировать тестовую микросервисную систему на предмет наличия антишаблонов. Для дополнительной валидации удобства и применимости решения группа разработчиков провела анализ системы с открытым исходным кодом, используя предложенный инструмент.

Авторы [11] в своем исследовании провели динамический анализ (как способ визуализации архитектуры микросервисной системы), что позволило оценить возможность построения системы, а впоследствии показать ее поведение во время функционирования. Авторы определили методы, соответствующие инструменты и модели, которые способны генерировать эти практики. Так же рассматривается возможность возникновения архитектурных антишаблонов в микросервисных приложениях. Для выявления таких проблем авторы предлагают инструмент, основанный на использовании трассируемых данных и алгоритмов теории графов, позволяющий строить графы зависимостей и извлекать архитектурные метрики.

Однако в рассмотренных работах недостаточное внимание уделено комплексному анализу надежности и отказоустойчивости системы на основе графовых метрик, а также формальной оценке вероятности отказа всего сервиса или его критических подсистем в зависимости от структуры взаимодействий. Кроме того, не в полной мере исследованы вопросы оптимизации архитектуры на основе анализа графовых моделей с целью упреждающего устранения потенциальных узких мест и циклических зависимостей.

Постановка задачи и концептуальные положения ее решения

Настоящее исследование направлено на устранение указанной проблемы посредством применения аппарата теории графов, позволяющего наглядно и структурировано описывать межкомпонентные связи, выявляя критические элементы и зоны потенциального риска в архитектуре микросервисов.

Теоретико-множественное описание микросервисной архитектуры системы $V = \{v_1, v_2, ..., v_n\}$ может быть представлено через множество отдельных микросервисов v_i, где $I = 1, n$. Каждый из микросервисов имеет свой функционал. Отношения между микросервисами E могут быть как прямыми, так и косвенными и иметь ряд свойств, например

рефлексивность, антиссиметричность, транзитивность и др. Отношение для двух микросервисов могут быть представлены как $E \subseteq V \times V$, где $(v_i, v_j) \in E$.

При описании архитектуры с помощью графа задача будет сведена к определению свойств отношений E и изучению структуры графа зависимостей, возникающего на множестве V.

В основу концептуальных положений для анализа микросервисной архитектуры при проектировании системы с целью выявления возможных проблемных ситуаций в процессе тестирования положена теория графов. Микросервисы (вершины графа), взаимодействуют между собой через специальные интерфейсы (ребра графа). При этом микросервисная архитектура может быть описана разными видами графов: структурным (физические/ логические связи), поведения (управляющие связи), передачи данных (информационные связи). В статье акценты делаются на графы передачи данных и поведения.

Графы служат основным инструментом анализа и проектирования микросервисных систем, обеспечивая глубокое понимание и прогнозирование поведения системы в различных условиях эксплуатации [16].

Данные передаются в одном направлении при описании микросервисной архитектуры, представленной ориентированным графом, и в двух направлениях – в неориентированном графе [12, 13, 18].

Ключевые аспекты теории графов, такие как петля, некорректно сформированный подграф, параллельные ребра, высокие степень захода и исхода, неудачно сформированный ориентированный путь, ориентированный цикл, ориентированный ациклический граф, сильно связные вершины, сильно связная компонента, определяют ситуации и возможные проблемы в архитектуре микросервисов, когда после их обнаружения необходимо применять соответствующие методы для корректировки (Табл. 1).

Таблица 1. Возможные ситуации и проблемы при наличии определенных элементов графа.

Элемент графа	Описание возможных ситуаций и проблем
Наличие петли	– зацикливание вызовов и бесконечная рекурсия, – нарушение порядка выполнения операций, – несвоевременное выполнение задач, – увеличение сетевого трафика, – повторное выполнение операций, – трудности с отладкой и анализом системы
Выделение или игнорирование важных подграфов	– усложнение управления и мониторинга, – недостаточная оптимизация производительности, – риск возникновения бутылочных горлышек (повышенная нагрузка), – сложность эффективного масштабирования, – снижение отказоустойчивости, – ошибки в проектировании, – сложность с миграцией
Наличие параллельных ребер	– дублирование вызовов и перегрузка ресурсов, – затруднение анализа и контроля системы, – перенасыщение очереди запросов, – ошибки в проектировании, – сложности с восстановлением после сбоев и масштабируемостью
Наличие высокой степени захода	– централизации нагрузки, – зависимость и каскадные сбои, – усложнение обновления и масштабирования, – рост задержки и ухудшение производительности, – затруднение мониторинга и анализа системы, – уязвимость к атакам и угрозам безопасности, – препятствия для миграции и реструктуризации

Продолжение Табл. 1

Высокая степень исхода	– комплексность и трудность поддержки, – зависимость от внешних сервисов, – риск возникновения лавинных сбоев, – повышенная уязвимость к сетевым задержкам, – рост нагрузки на ресурсы, – сложность с оптимизацией производительности, – проблемы с масштабированием, – возникновение временной несогласованности
Неудачно-сформированный ориентированный путь	– увеличение времени отклика, – снижение производительности, – высокий риск отказов, – зависимость от удаленных сервисов, – проблемы с мониторингом и наблюдаемостью, – отсутствие гибкости, – поверхностное тестирование, – ограничения масштабируемости
Наличие ориентированного цикла	– бесконечные рекурсивные вызовы, – высокая нагрузка на систему, – истощение ресурсов, – проблемы с производительностью (замедление реакции, перегрузка сети), – сложности с отладкой и диагностикой, – зависимость между микросервисами, – проблемы с масштабированием, – перекрестные блокировки и deadlock, – трудности с эксплуатацией и поддержкой (падение надежности)
Неправильное использование или реализация ориентированного ацикличного графа	– ограничения в топологии вызовов, – ограничение функциональности, – потеря гибкости, – длительное время отклика, – падение производительности и сложности с масштабированием
Наличие сильно связных вершин	– ухудшение отказоустойчивости, – сложности с масштабированием, – замедление разработки и ухудшение управляемости системы
Наличие сильно-связанных компонент (максимально связный подграф)	– сложность масштабирования, – низкая отказоустойчивость, – замораживание системы при блокировке, – ограниченная независимость микросервисов, – проблемы с производительностью, – сложности с управлением и пониманием архитектуры, – увеличивается риск образования блоков

Различные элементы графа могут быть сведены к свойствам отношений. Например, наличие петли в графе соответствует такому свойству отношений, как рефлексивность; наличие сильно-связанных вершин – транзитивность; наличие ориентированного цикла в графе – цикличность.

Таким образом, концептуальные положения оценки микросервисной архитектуры могут быть представлены с использованием теоретико-множественного описания как множество

элементов графа $G = (V, E)$ с матрицей смежности Am, приводящих к сложным ситуациям S; множество методов и алгоритмов A оценки этих ситуаций; множество метрик оценивания M структурной сложности, связности, надежности и производительности системы в целом: $CP = \{V, E, Am, S, A, M, Pr, Res, Cond\}$. Концептуальные положения расширены принципами Pr, ограничениями Res и условиями $Cond$. Графовое представление делает архитектуру системы измеримой и пригодной для анализа. Формальное определение проблемных ситуаций позволяет выявлять сценарии, требующие повышенного внимания при тестировании. Наличие в концепции методов и алгоритмов для оценки ситуаций позволяет научно обосновать результаты тестирования. Принципы, условия и ограничения предлагаемой концепции оценки микросервисной архитектуры представлены в Табл. 2.

Таблица 2. Описание принципов, условий и ограничений концепции оценки микросервисных решений.

Принципы/ условия/ ограничения	**Описание**
Принципы	
причинно-следственной связи	Проблемная ситуация должна быть связана с конкретной структурной характеристикой графа, которая ее вызывает
верифицируемости	Методы и метрики должны быть основаны на общепризнанных или четко определенных алгоритмах и давать воспроизводимые, объективные результаты
иерархичности	Для обеспечения анализа на разных уровнях абстракции
однозначной декомпозиции	Элемент программного решения должен быть однозначно представлен как вершина или как ребро без дублирования или пересечения ролей для обеспечения четкости и полноты графовой модели
ориентированного ациклического графа	Позволяет формально описать зависимости между отдельными компонентами, гарантируя, что не возникнет круговых зависимостей, которые могут привести к взаимоблокировкам или непредсказуемому поведению системы
Ограничения	
полноты данных	Модель будет работать корректно только при условии, что граф полностью и актуально отражает реальную архитектуру системы. Если какие-то скрытые взаимодействия не учтены в графе, результаты оценки будут неверны
на вычислительную сложность	Применение некоторых алгоритмов на очень больших графах может быть NP-полной задачей и потребовать неприемлемо больших вычислительных ресурсов или времени
применимости метрик	Метрики структурной сложности (например, цикломатическая сложность) не всегда напрямую коррелируют с бизнес-метриками (например, удовлетворенность клиента)
на динамические изменения	Статическая матрица смежности или граф плохо моделируют динамическое поведение системы
Условия	
декомпозиции системы	Каждый компонент реальной программной системы должен быть представлен ровно одной вершиной в графе, каждая вершина соответствует одному компоненту

Продолжение Табл. 2

направленности связей	Если модель описывает архитектуру микросервисов (потоки запросов), то все ребра должны быть ориентированными, что соответствует принципу ориентированного ациклического графа
полноты связей	Множество ребер должно включать все существующие взаимодействия между вершинами
причинности ситуаций	Каждая сложная ситуация должна иметь поддающийся оценке и идентифицируемый источник в графе, который может быть проанализирован с помощью методов
применимости алгоритмов	Выбранные методы и алгоритмы должны быть адекватно применимы к типу графа, который моделирует систему
измеримости метрик	Все элементы множества метрик должны иметь четкое, количественное определение, которое может быть вычислено на основе данных графа
валидации	Результаты, полученные с помощью концепции, должны быть верифицированы на реальной работающей системе

Проведение эксперимента

Предлагаемую для эксперимента систему – вебсервис – можно представить моделью в виде ориентированного графа, поскольку она содержит следующие микросервисы (Табл. 3).

Таблица 3. Микросервисы и их функции.

Название микросервиса	Функция
A	Аутентификация
B	Пользовательский сервис
C	Сервис товаров
D	Платежный сервис
E	Биллинг
F	Почтовый сервис
G	Сервис доставки

Представим эту систему в виде ориентированного графа следующим образом:

$$G = (V, E), \tag{1}$$

где V – множество вершин (сервисов), $E \subseteq V \times V$ – множество ориентированных ребер (взаимодействий между сервисами).

Примеры взаимодействий:

$$\begin{gathered} A \to B; \\ B \to C; \\ C \to D; \\ D \to E; \\ E \to F; \\ D \to G. \end{gathered} \tag{2}$$

Матрица смежности

Представим архитектуру в виде матрицы смежности МС [7], где значение элемента матрицы смежности при наличии и отсутствии связи между вершинами принимают значения 1 и 0 соответственно:

$$\mathrm{MC}_{ij} = \begin{cases} 1, \text{если существует связь из } i \text{ в } j \\ 0, \text{иначе} \end{cases} \quad \mathrm{MC} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{3}$$

Операции над графом

– *Поиск путей*: для решения может быть использован Breadth First Search (BFS) / Depth First Search (DFS), в частности для поиска всех возможных путей от A до G или между любыми двумя службами. Количество путей отражает вариативность сценариев использования [8].

– *Обнаружение циклов*: для решения применим алгоритм Тарджана или Косарайджу. Циклы в архитектуре представляют собой опасные взаимозависимости, мешающие тестированию и масштабированию. Это покажет все возможные циклы, и в соответствии с этим можно будет принять множество решений [9, 17].

– *Надежность и отказоустойчивость*: для решения предположим, что каждый сервис/связь имеет вероятность успешного выполнения $P(i \to j)$. Тогда

$$P_{\text{общая}} = \prod_{i \to j \in \text{путь}} P\,(i \to j). \tag{4}$$

В этом случае матрица МС будет иметь значения, отличные от 1, например,

$$\mathrm{MC} = \begin{pmatrix} 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Это позволяет рассчитать вероятность успеха выполнения запроса по определенному пути.

Вычисление ключевых метрик

Степень важности узла:

$$\text{Степень}(v) = \deg^{+}(v) + \deg^{-}(v). \tag{5}$$

Сервисы с высокой степенью являются критическими точками, которым необходим тщательный мониторинг.

Сложность архитектуры:

$$T = |E|(\text{количество связей}). \tag{6}$$

Большое число связей указывает на сильную связанность, что затрудняет тестирование.

Метрика надежности:

$$R = \frac{\text{количество надежных путей}}{\text{общее число путей}}. \tag{7}$$

Малое значение указывает на наличие узких мест и единые точки отказа. Результаты эксперимента показывают:

– Сервис D имеет наибольшее количество исходящих ребер: он участвует в двух критических потоках – на биллинг и доставку.

– При сбое D нарушается сразу несколько функций – его степень важности максимальна.

– Модель позволяет визуализировать наиболее уязвимые компоненты системы.

Результаты и обсуждение

Результаты проведенного эксперимента на системе, основанной на микросервисной архитектуре, позволяют сделать выводы:

1. Система с микросервисами может быть представлена графом, основанным на ее системных деталях и описании.

2. Алгоритмы поиска в графе могут быть использованы для:

– поиска путей в графе, то есть обнаружения связей между микросервисами;

– поиска и обнаружения критических путей и узлов между всеми микросервисами;

– прогнозирования вероятности отказа между микросервисами.

3. Теория графов для представления микросервисов может быть использована для:

– оптимизации структуры любой системы микросервисов;

– тестирования и оценке автоматизации системы.

Предложенная концепция оценки микросервисных решений апробирована на примере вебсервиса, что позволило сделать вывод о ее применимости, в частности при описании системы графом как инструментом статики, не учитывающим динамику изменений [15]. Авторы продолжат свои исследования с целью разработки средств, необходимых при тестировании для обеспечения надежности и эффективности микросервисных архитектур при необходимости адаптации к изменениям.

Выводы

Теория графов с ее функциями и мощными инструментами используется для представления структуры системы микросервисов. Концептуальные положения включают представление микросервисного решения моделью графа, при оценке которой могут быть выявлены сложные ситуации, требующие разрешения для обеспечения надежности и производительности системы в целом.

Предложенное решение может быть использовано для оценки и тестирования структуры системы микросервисов, для мониторинга системы микросервисов в реальном времени.

Литература:

1. Босенко Т.М. Анализ микросервисной архитектуры в среде e-learning с многовариантным доступом к учебным материалам // Международный научно-исследовательский журнал. 2024. № 10(148). С. 65–71. [Bosenko T.M. Analysis of microservice architecture in e-learning environment with multivariate access to learning materials // Meždunarodnyj naučno-issledovatel'skij žurnal. 2024. No. 10(148). P. 65–71 (in Russian)]. DOI: 10.60797/IRJ.2024.148.65

2. Bural H., Koyuncu M., Guney S. A Systematic Literature Review on Microservices. In: Computational Science and Its Applications – ICCSA 2017. Springer, 2017. P. 203–217. DOI: 10.1007/978-3-319-62407-5_14.

3. Ганьжа А.Ю., Карелова Р.А. Особенности микросервисной архитектуры событийно-ориентированных веб-приложений // Научное обозрение. Технические науки. 2021. № 6. С. 28–34. [Ganzha A.Yu., Karelova R.A. Features of the microservice architecture of event-oriented web-applications // Scientific Review. Technical science. 2021. No. 6. P. 28–34 (in Russian)].

4. Гольчевский Ю.В., Ермоленко А.В. Актуальность использования микросервисов при разработке информационных систем // Вестник Сыктывкарского университета. Сер. 1: Математика. Механика. Информатика. 2020. Вып. 2(35). С. 25–36. [Golchevskiy Yu.V., Yermolenko A.V. The relevance of using microservices in the development of information systems // Bulletin of Syktyvkar University. Series 1: Mathematics. Mechanics. Informatics. 2020. Iss. 2(35). P. 25–36 (in Russian)].

5. Городничев М.Г., Полонский Р.В. Оценка возможности использования микросервисной архитектуры при разработке пользовательских интерфейсов клиент-серверного программного обеспечения // Экономика и качество систем связи. 2020. № 3. С. 33–43. [Gorodnichev M., Polonsky R. Assessment of possibilities to apply microservice architecture approach for client-side development in client-server software // Ekonomika i Kachestvo Sistem Svyazi. 2020. No. 3. P. 33–43 (in Russian)].
6. Кравченко Д. А. Микросервисная архитектура // Интерактивная наука. 2022. Т. 4. № 69. С. 43–45. [Kravchenko D.A. Microservice architecture // Interaktivnaya Nauka. 2022. Vol. 4. No. 69. P. 43–45 (in Russian)].
7. Малыгин Д.С. Микросервисная архитектура в облачных системах: риски и возможности применения в 2024–2030 гг. // Моделирование, оптимизация и информационные технологии. 2024. Т. 12. № 2. [Malygin D.S. Microservice architecture in cloud systems: risks and application opportunities in 2024–2030 // Modeling, Optimization and Information Technology. 2024. Vol. 12. No. 2 (in Russian)]. DOI: 10.26102/2310-6018/2024.45.2.029.
8. Подгорная Ю.С., Гришанов Н.В., Суворов А.А. Матрица смежности и изоморфизм графов. В сб. Материалы 57-й Международной научной студенческой конференции. МНСК-2019. Новосибирск: Новосибирский национальный исследовательский государственный университет, 2019. С. 17. [Podgornaya Yu.S., Grishanov N.V., Suvorov A.A. Matritsa smezhnosti i izomorfizm grafov. In: Proceedings of the 57th International Scientific Student Conference. ISSC-2019. Novosibirsk: Novosibirsk National Research State University, 2019. P. 17 (in Russian)].
9. Резенов Г.В. Архитектура микросервисов и ее реализация с помощью технологии контейнеризации // Вестник РосНОУ. Серия «Сложные системы: модели, анализ и управление». 2020. № 3. С. 131–138. [Rezenov G.V. Architecture of microservices and its implementation with containerization technology // Vestnik of Russian New University. Series “Complex Systems: models, analysis and management”. 2020. No. 3 P. 131–138 (in Russian)].
10. Харазян А.А. Особенности микросервисной архитектуры для современных приложений // Электронные информационные системы. 2023. № 2(37). С. 45–50. [Kharazyan H.A. Specifi of microservice architecture for modern applications // Electronic Information Systems. 2023. No. 2(37). P. 45–50 (in Russian)].
11. Banerjee N. et al. Space efficient linear time algorithms for BFS, DFS and applications // Theory of Computing Systems. 2018. Vol. 62. P. 1736–1762.
12. Dilruksh T., Polyvyanyy A., Buyya R., Barros A., Fidge C. Microservices-based Software Systems Reengineering: State-of-the-Art and Future Directions // ACM Computing Surveys. 2024. – Vol. 1. No. 1. 40 p. arXiv:2407.13915v1
13. Edmonds C. Undirected graph theory // Archive of Formal Proofs. 2022. September.
14. Gamage I. U. P., Perera I. Using dependency graph and graph theory concepts to identify anti-patterns in a microservices system: A tool-based approach. In: 2021 Moratuwa Engineering Research Conference (MERCon). IEEE, 2021. P. 699–704.
15. Gortney M. E. et al. Visualizing microservice architecture in the dynamic perspective: A systematic mapping study // IEEE Access. 2022. Vol. 10. P. 119999–120012.
16. Li S. et al. Understanding and addressing quality attributes of microservices architecture: A Systematic literature review // Information and Software Technology. 2021. Vol. 131. Art. 106449.
17. Spek H. L. A. van der. Towards an Automatic Derivation of Tarjan's Algorithm for Detecting Strongly Connected Components in Directed Graphs: Thesis Master Computer Science. Leiden: Leiden University, 2006.
18. Zhao X. et al. UGRec: Modeling directed and undirected relations for recommendation. In: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval. New York: Association for Computing Machinery, 2021. P. 193–202.

Об авторах:
ЧЕМБАРИСОВ Эмиль Марсович, аспирант третьего года обучения Уфимского университета науки и технологий, Россия, Республика Башкортостан, 450076 г. Уфа, ул. Заки Валиди, 32, lawluker@gmail.com.
СМЕТАНИНА Ольга Николаевна, доктор технических наук, доцент, профессор кафедры ВМиК Уфимского университета науки и технологий, Россия, Республика Башкортостан, 450076 г. Уфа, ул. Заки Валиди, 32, smoljushka@mail.ru.
САЗОНОВА Екатерина Юрьевна, кандидат технических наук, доцент, доцент кафедры ВМиК Уфимского университета науки и технологий, Россия, Республика Башкортостан, 450076 г. Уфа, ул. Заки Валиди, 32, rassadnikova_ekaterina@mail.ru

Metadata:
Title: Graph models and operations in evaluating microservice solutions
Author 1: Emil Marsovich Chembarisov, third year graduate at Ufa University of Science and Technology, 32 Zaki Validi st., 450076 Ufa, Republic of Bashkortostan, Russia, lawluker@gmail.com.
Author 2: Olga Nikolaevna Smetanina, Doctor of Technical Sciences, Docent, Professor of the Calculation Mathematics and Cibernetics Department at Ufa University of Science and Technology, 32 Zaki Validi st., 450076 Ufa, Republic of Bashkortostan, Russia, smoljushka@mail.ru.
Author 3: Yekaterina Yuryevna Sasonova, Candidate of Technical Sciences, Docent, Associate Professor of the CMaC Department at Ufa University of Science and Technology, 32 Zaki Validi st., 450076 Ufa, Republic of Bashkortostan, Russia, rassadnikova_ekaterina@mail.ru
Abstract: Microservice architecture played an important role as a dominant factor in different software system design; it offers scalability and the ability to maintenance. The increasing complexity of microservice-systems faces significant challenges in understanding and evaluation. This paper proposes a formal and mathematical representation of microservice architecture using graph theory, where each microservice is represented as a vertex and each communication or dependency between two microservices is represented as a directed edge. Mathematical model based on adjacency matrices was introduced, graph theory algorithms is explored. Graph metrics were studied to analyze structural properties, find critical nodes, and detect potential failure points. Other operations are also explored such as analysis of path, finding and detecting cycles, and connectivity evaluation, these operations are important for testing, reliability assessment, and fault tolerance in distributed systems. The proposed approach is tested on a case study of a hypothetical microservice system. The recommendations suggest that graph-theoretic modeling provides a powerful foundation for the formal analysis, monitoring, and optimization of microservice architectures.
Keywords: microservice architecture, graph theory, adjacency matrix, vertexes, edges.