

Unified Availability Model (UAM)

Методика расчёта доступности разнородных ИТ-систем

Алексей Алексеевич Неклюдов

Руководитель исследовательских проектов

AstraVerge Research Lab

WWW: <https://astraverge.ru>

E-mail: a.nekliudov@astraverge.ru

25 ноября 2025

Содержание

1	Введение	6
2	Основные определения	7
2.1	Доступность системы	7
2.2	Метрика	7
2.3	Нормализация	7
2.4	Вес метрики	7
2.5	Доступность контура	7
3	Общая модель Unified Availability Model	7
3.1	Формула доступности системы	8
3.2	Формула доступности контура	8
4	Нормализация метрик	8
4.1	Стабильный baseline	8
4.2	Пороги SLA	9
4.3	Известные физические границы	9
4.4	Допустимые отклонения	10
4.5	Итоговая нормализация	10
5	Типовые стратегии нормализации	10
5.1	Линейная нормализация	10
5.2	Нормализация по SLA	10
5.3	Нормализация с saturating-функцией	11
5.4	Пример нормализации latency	11
6	Примеры применения нормализации для различных типов систем	11
6.1	API-сервисы и платёжные шлюзы	11
6.2	Криптографические сервисы (HSM, подпись)	12
6.3	Batch-процессы, пачки и отчёты	12
6.4	ERP / 1C:ERP	13
6.5	ERP / SAP S/4HANA	14
6.6	ERP / Oracle EBS	14
6.7	SAP IDoc / EDI интеграции	15
6.8	Сравнительная таблица нормализационных подходов	16
7	Типы систем и рекомендуемые метрики	16
7.1	API-сервисы и платёжные шлюзы	17
7.2	Банковские API	17

7.3	Криптографические сервисы	17
7.4	Batch-процессы	17
7.5	ERP / 1C:ERP	18
7.6	ERP / SAP S/4HANA	18
7.7	ERP / Oracle EBS	18
7.8	SAP IDoc / EDI Интеграции	18
8	Доступность контура	19
8.1	Определение контура	19
8.2	Состав контура	19
8.3	Весовой коэффициент системы в контуре	20
8.4	Формула доступности контура	20
8.5	Пример структуры контура	21
8.6	Почему доступность контура важнее доступности отдельных систем	21
9	Сквозной пример расчёта доступности контура	21
9.1	Метрики подсистем	22
9.1.1	Web-сайт	22
9.1.2	Банк-клиент API	23
9.1.3	1C ERP	23
9.2	Расчёт доступности контура	24
9.3	Интерпретация	24
10	Реализация в мониторинге	24
10.1	Prometheus	24
10.2	Zabbix	25
10.3	Grafana	25
11	Иерархическая модель когерентности уровней (расширение к Unified Availability Model)	25
11.1	Введение	25
11.2	Архитектура уровней	25
11.3	Проблема фиксированных весов и динамическая перенормировка	26
11.4	Мягкая иерархия уровней (Multiplicative Coherence Model)	26
11.5	Жёсткая иерархия (Foundational Limit Model)	27
11.6	Динамическая логика весов внутри уровней	27
12	Заключение	27

Appendix A. Fuzzy-логика и Нейросети	28
Список литературы	31

Abstract

The Unified Availability Model (UAM) proposes a formal, extensible framework for evaluating the availability of heterogeneous information systems whose operational characteristics, performance indicators, and failure modes differ fundamentally across layers and functional domains. Unlike traditional SRE- and API-centric approaches that assume homogeneous metric spaces and rely primarily on latency–error abstractions, UAM introduces a generalizable normalization methodology capable of transforming diverse metric types into a unified, comparable scale.

The model is based on four foundational constructs — statistical baselines, SLA thresholds, physical system limits, and acceptable deviation ranges — each represented as a mathematically defined normalization operator. By combining these operators through a weighted linear composition, UAM enables consistent availability scoring for systems as diverse as REST/gRPC microservices, payment gateways, cryptographic HSM-signing modules, batch/ETL pipelines, ERP platforms (Oracle EBS, SAP, 1C), and EDI/IDoc integration channels.

At the higher level, UAM introduces the concept of a business-process contour — a structured composite of heterogeneous subsystems — and defines contour availability as a weighted aggregation of subsystem availability scores. This provides a unified and interpretable indicator of end-to-end business operability, compatible with real-world observability tools such as Prometheus, Zabbix, Grafana, and VictoriaMetrics.

The proposed model offers both theoretical novelty — by formalizing cross-system metric normalization — and practical applicability, providing engineering teams with a consistent methodology for measuring, comparing, and governing availability across complex IT landscapes. UAM constitutes a step toward a generalized mathematical foundation for reliability assessment in multi-layered enterprise systems.

Аннотация

Предлагаемая модель Unified Availability Model (UAM) формирует формальную и расширяемую методологическую рамку для оценки доступности разнородных информационных систем, существенно отличающихся по своим архитектурным свойствам, режимам работы и типам наблюдаемых метрик. В отличие от традиционных SRE- и API-центричных подходов, предполагающих однородность метрик и опирающихся главным образом на пары «latency–error», методика UAM вводит универсальную процедуру нормализации, позволяющую приводить различные типы метрик к единой, сопоставимой шкале.

Модель основывается на четырёх фундаментальных конструктах — статистическом baseline, SLA-порогах, физических границах системы и допустимых отклонениях — каждый из которых формализован как независимый оператор нормализации. Взвешенное объединение этих операторов обеспечивает единообразный расчёт доступности для широкого спектра систем: REST/gRPC API, платёжных шлюзов, криптографических HSM-

модулей, batch/ETL процессов, ERP-платформ (Oracle EBS, SAP, 1C), а также интеграционных каналов EDI/IDoc.

На верхнем уровне UAM вводит понятие бизнес-контура — структурной композиции разнородных подсистем — и определяет доступность контура как взвешенную агрегированную функцию доступностей отдельных компонентов. Это позволяет получать единый, интерпретируемый показатель работоспособности сквозного бизнес-процесса, совместимый с практическими средствами наблюдаемости (Prometheus, Zabbix, Grafana, VictoriaMetrics).

Представленная модель обладает как теоретической новизной — благодаря формализации межсистемной нормализации метрик, — так и высокой практической применимостью, обеспечивая инженерные команды унифицированным подходом к измерению, сравнению и управлению доступностью сложных корпоративных ИТ-ландшафтов. UAM формирует основу для развития обобщённой математической теории оценки надёжности многослойных enterprise-систем.

1. Введение

Современные ИТ-ландшафты представляют собой **разнородные и многослойные** системы, включающие (но не ограничиваясь) следующими категориями:

- API-сервисы и микросервисы,
- платёжные шлюзы,
- банковские интеграции,
- криптографические и HSM-сервисы,
- batch-процессы и ETL,
- ERP-системы (например, 1C:ERP, SAP S/4HANA, Oracle EBS),
- системы формирования отчетности.

Каждая из этих категорий имеет свои принципы работы, характер нагрузки и свои характерные метрики. Поэтому **единая универсальная метрика доступности невозможна**.

Однако возможно создать **единый методический подход**, при котором:

- каждая система оценивается по своим собственным метрикам,
- метрики нормализуются к единой шкале,
- итоговая доступность рассчитывается как взвешенная сумма.

Такой подход позволяет:

- корректно сравнивать разные системы,
- строить интегральный показатель доступности контура,
- унифицировать отчётность и мониторинг,
- легко расширять модель под новые классы систем.

2. Основные определения

2.1. Доступность системы

Доступность системы A_i — это нормализованная оценка состояния системы на интервале наблюдения, принимающая значение от 0 до 1 или в процентах от 0 до 100.

2.2. Метрика

Метрика M_{ij} — количественный показатель, характеризующий состояние системы i по параметру j .

2.3. Нормализация

Нормализация — преобразование метрики M_{ij} в значение $N_{ij} \in [0, 1]$ по правилам, заданным для категории системы.

2.4. Вес метрики

w_{ij} — весовой коэффициент, определяющий вклад метрики j в итоговую доступность системы i , при этом:

$$\sum_j w_{ij} = 1.$$

2.5. Доступность контура

Доступность контура A_{contour} — взвешенная сумма доступностей систем, входящих в цепочку бизнес-процесса.

3. Общая модель Unified Availability Model

Каждая система S_i имеет набор метрик:

$$\{M_{i1}, M_{i2}, \dots, M_{ik}\}.$$

Каждая метрика нормализуется:

$$N_{ij} = f_{\text{norm}}(M_{ij}),$$

где $N_{ij} \in [0, 1]$.

Каждая нормализованная метрика имеет вес w_{ij} .

3.1. Формула доступности системы

$$A_i = \sum_{j=1}^k w_{ij} \cdot N_{ij}.$$

Итоговое значение A_i выражается либо в $[0, 1]$, либо в процентах.

3.2. Формула доступности контура

Пусть контур состоит из n систем. Тогда:

$$A_{\text{contour}} = \sum_{i=1}^n W_i \cdot A_i,$$

где W_i — вес системы в контуре, $\sum_i W_i = 1$.

4. Нормализация метрик

В модели Unified Availability Model (UAM) каждая метрика M_{ij} нормализуется к значению $N_{ij} \in [0, 1]$. Нормализация основана на четырёх фундаментальных концепциях:

1. стабильный baseline,
2. пороги SLA,
3. известные физические границы,
4. допустимые отклонения.

Далее приведены формальные определения и правила нормализации, определяющие преобразование исходных метрик в нормализованную форму.

4.1. Стабильный baseline

Определение. Стабильный baseline метрики M — статистически устойчивая характеристика типичного поведения системы при отсутствии аномалий.

Baseline отражает «нормальный» режим работы и используется как точка отсчёта для выявления деградаций. Он определяется по историческим данным.

Правила нормализации.

Пусть B — baseline, вычисленный одним из статистических методов:

- медиана: $B = \text{median}(M_{\text{hist}})$,
- квантиль $p75$: $B = \text{quantile}_{0.75}$,

- экспоненциальное сглаживание: $B = \text{EWMA}(M_{\text{hist}})$,
- сезонные baseline (по часам, дням недели).

Пусть k — коэффициент допуска (обычно 1.5–2). Тогда:

$$N_{\text{baseline}}(M) = \begin{cases} 1, & M \leq kB, \\ \frac{kB}{M}, & M > kB. \end{cases} \quad (1)$$

4.2. Пороги SLA

Определение. SLA — нормативное значение метрики, определяющее границу между «допустимым» и «недопустимым» качеством работы сервиса.

Правила нормализации.

Формально SLA задаётся условиями:

$$M \leq SLA \quad (\text{для latency}), \quad M \geq SLA \quad (\text{для success rate}).$$

$$N_{\text{SLA}}(M) = \begin{cases} 1, & M \text{ не нарушает SLA,} \\ 1 - \alpha(M - SLA), & M \text{ нарушает SLA,} \end{cases} \quad (2)$$

где α — штрафной коэффициент.

4.3. Известные физические границы

Определение. Физические границы — ограничения аппаратного и архитектурного характера, не зависящие от SLA.

Типичные примеры:

- пропускная способность канала,
- максимальное число соединений в пуле,
- лимит CPU или IO,
- архитектурный предел RPS.

Правила нормализации.

Пусть диапазон физических границ:

$$M_{\min}^{\text{phys}} \leq M \leq M_{\max}^{\text{phys}}.$$

Тогда:

$$N_{\text{phys}}(M) = \frac{M_{\text{max}}^{\text{phys}} - M}{M_{\text{max}}^{\text{phys}} - M_{\text{min}}^{\text{phys}}}. \quad (3)$$

4.4. Допустимые отклонения

Определение. Допустимые отклонения — диапазон значений метрики, в пределах которого небольшие колебания не считаются деградацией.

$$D_{\text{min}} \leq M \leq D_{\text{max}}.$$

Правила нормализации.

$$N_{\text{dev}}(M) = \begin{cases} 1, & D_{\text{min}} \leq M \leq D_{\text{max}}, \\ 1 - \beta(M - D_{\text{max}}), & M > D_{\text{max}}, \\ 1 - \beta(D_{\text{min}} - M), & M < D_{\text{min}}, \end{cases} \quad (4)$$

где β — коэффициент штрафа.

4.5. Итоговая нормализация

Для каждой метрики итоговая нормализация определяется композицией подходов:

$$N(M) = w_b N_{\text{baseline}}(M) + w_s N_{\text{SLA}}(M) + w_p N_{\text{phys}}(M) + w_d N_{\text{dev}}(M), \quad (5)$$

где $w_b + w_s + w_p + w_d = 1$.

5. Типовые стратегии нормализации

5.1. Линейная нормализация

$$N = 1 - \frac{M - M_{\text{min}}}{M_{\text{max}} - M_{\text{min}}}.$$

5.2. Нормализация по SLA

$$N = \begin{cases} 1, & M \leq SLA, \\ 1 - k(M - SLA), & M > SLA. \end{cases}$$

5.3. Нормализация с saturating-функцией

$$N = e^{-\alpha M}.$$

5.4. Пример нормализации latency

В качестве примера рассмотрим нормализацию метрики задержки.

$$N_{\text{latency}} = \begin{cases} 1, & L \leq L_{\text{baseline}}, \\ \frac{L_{\text{max}} - L}{L_{\text{max}} - L_{\text{baseline}}}, & L > L_{\text{baseline}}. \end{cases}$$

6. Примеры применения нормализации для различных типов систем

Ниже приведены примеры операционализации нормализации метрик для четырёх критически различных классов систем: API-сервисы, криптосервисы, batch-процессы и ERP/Oracle EBS. Эти примеры демонстрируют, каким образом понятия baseline, SLA, физических границ и допустимых отклонений применяются к реальным метрикам наблюдаемости.

6.1. API-сервисы и платёжные шлюзы

Для API-сервисов основными метриками являются:

- M_1 : latency p_{99} ;
- M_2 : success rate;
- M_3 : доля 5xx ошибок;
- M_4 : глубина очередей или backlog.

Пример baseline:

$$B_{\text{latency}} = \text{median}(L_{\text{hist}}) = 85 \text{ ms}.$$

Пример SLA:

$$\text{latency}_{p_{99}} \leq 300 \text{ ms}.$$

Пример физического предела:

$$L_{\text{max}}^{\text{phys}} = 2000 \text{ ms} \quad (\text{лимит таймаута ядра Nginx}).$$

Допустимое отклонение:

$$D_{\text{max}} = 2.0 \cdot B_{\text{latency}}.$$

Тогда нормализация latency:

$$N_{\text{API-lat}} = \begin{cases} 1, & L \leq 2B_{\text{latency}}, \\ \frac{2B_{\text{latency}}}{L}, & L > 2B_{\text{latency}}. \end{cases}$$

6.2. Криптографические сервисы (HSM, подпись)

Характерные метрики:

- M_1 : доступность HSM (онлайн/офлайн);
- M_2 : время подписи документа;
- M_3 : доля отказов подписи;
- M_4 : доступность слотов ключей.

Пример baseline времени подписи:

$$B_{\text{sign}} = \text{median}(T_{\text{sign}}) = 42 \text{ ms.}$$

SLA:

$$T_{\text{sign}} \leq 150 \text{ ms.}$$

Физические границы:

$$T_{\text{min}}^{\text{phys}} = 15 \text{ ms}, \quad T_{\text{max}}^{\text{phys}} = 500 \text{ ms.}$$

Допустимое отклонение:

$$D_{\text{max}} = 1.5 \cdot B_{\text{sign}}.$$

Нормализация:

$$N_{\text{sign}} = \begin{cases} 1, & T \leq D_{\text{max}}, \\ 1 - 0.01(T - D_{\text{max}}), & T > D_{\text{max}}. \end{cases}$$

6.3. Batch-процессы, пачки и отчёты

Характерные метрики:

- M_1 : длительность выполнения job;
- M_2 : завершение в окно;
- M_3 : ошибки исполнения;
- M_4 : backlog и накопления в очередях.

Пример baseline:

$$B_{\text{job}} = \text{median}(t_{\text{job,hist}}) = 17 \text{ min.}$$

SLA:

$$t_{\text{job}} \leq 30 \text{ min.}$$

Физический предел:

$$t_{\text{max}}^{\text{phys}} = 120 \text{ min} \quad (\text{переполнение batch-окна}).$$

Допустимое отклонение:

$$D_{\text{max}} = 1.2 \cdot B_{\text{job}}.$$

Нормализация длительности job:

$$N_{\text{batch}} = \begin{cases} 1, & t \leq D_{\text{max}}, \\ \frac{D_{\text{max}}}{t}, & t > D_{\text{max}}. \end{cases}$$

6.4. ERP / 1C:ERP

Критические метрики:

- M_1 : среднее время выполнения запросов в кластере 1C;
- M_2 : загрузка рабочих процессов (RPH, количество активных процессов);
- M_3 : очередь запросов на сервере 1C (request queue);
- M_4 : задержки на уровне СУБД (чаще всего PostgreSQL, MS SQL или Oracle).

Пример baseline времени выполнения запроса:

$$B_{1C} = \text{median}(T_{\text{request}}) = 35 \text{ ms.}$$

SLA:

$$T_{\text{request}} \leq 150 \text{ ms.}$$

Физические пределы:

$$T_{\text{min}}^{\text{phys}} = 10 \text{ ms}, \quad T_{\text{max}}^{\text{phys}} = 800 \text{ ms.}$$

Допустимое отклонение:

$$D_{\text{max}} = 2.5 \cdot B_{1C}.$$

Нормализация:

$$N_{IC} = \begin{cases} 1, & T \leq D_{\max}, \\ 1 - 0.003 (T - D_{\max}), & T > D_{\max}. \end{cases}$$

6.5. ERP / SAP S/4HANA

Критические метрики:

- M_1 : SAP Dialog Response Time (включая DB Time, CPU Time, Wait Time);
- M_2 : Work Process Utilization (DIA, BTC, UPD, SPO);
- M_3 : Queue Length (SM50/SM66);
- M_4 : HANA DB latency и блокировки (Lock Wait Time).

Пример baseline времени отклика диалоговой операции:

$$B_{SAP} = \text{median}(T_{\text{dialog}}) = 280 \text{ ms.}$$

SLA (стандарт SAP):

$$T_{\text{dialog}} \leq 1000 \text{ ms.}$$

Физические пределы (архитектурные):

$$T_{\min}^{\text{phys}} = 50 \text{ ms}, \quad T_{\max}^{\text{phys}} = 5000 \text{ ms.}$$

Допустимое отклонение:

$$D_{\max} = 2.0 \cdot B_{SAP}.$$

Нормализация:

$$N_{SAP} = \begin{cases} 1, & T \leq D_{\max}, \\ 1 - 0.0005 (T - D_{\max}), & T > D_{\max}. \end{cases}$$

6.6. ERP / Oracle EBS

Критические метрики:

- M_1 : ожидания в БД (TX, TM, enq:...);
- M_2 : активные Concurrent Managers;
- M_3 : глубина workflow-lag;

- M_4 : загрузка session pool.

Пример baseline:

$$B_{TX} = \text{median}(\text{wait_TX}) = 8 \text{ ms.}$$

SLA:

$$\text{wait_TX} \leq 40 \text{ ms.}$$

Физический предел:

$$\text{wait_TX}^{\max} = 500 \text{ ms.}$$

Допустимое отклонение:

$$D_{\max} = 3 \cdot B_{TX}.$$

Нормализация:

$$N_{EBS} = \begin{cases} 1, & W_{TX} \leq D_{\max}, \\ 1 - 0.005(W_{TX} - D_{\max}), & W_{TX} > D_{\max}. \end{cases}$$

6.7. SAP IDoc / EDI интеграции

Критические метрики:

- M_1 : время обработки IDoc (end-to-end latency);
- M_2 : доля IDoc в статусах ошибки (status 51/68);
- M_3 : очередь IDoc-обработки (Inbound Queue / Outbound Queue);
- M_4 : блокировки объектов (Lock Conflicts) при обработке IDoc.

Пример baseline времени обработки:

$$B_{IDoc} = \text{median}(T_{idoc}) = 1.8 \text{ s.}$$

SLA (типичное для B2B/EDI потоков):

$$T_{idoc} \leq 5 \text{ s.}$$

Физические пределы (архитектурные ограничения):

$$T_{\min}^{\text{phys}} = 0.5 \text{ s}, \quad T_{\max}^{\text{phys}} = 30 \text{ s.}$$

Допустимое отклонение:

$$D_{\max} = 2.0 \cdot B_{IDoc}.$$

Нормализация:

$$N_{\text{IDoc}} = \begin{cases} 1, & T \leq D_{\max}, \\ 1 - 0.02(T - D_{\max}), & T > D_{\max}. \end{cases}$$

6.8. Сравнительная таблица нормализационных подходов

Параметр	Baseline	SLA	Физические границы / Допустимые отклонения
Источник значения	Исторические данные (эмпирика)	Норматив / обязательство	Архитектурные/аппаратные ограничения
Тип контроля	Выявление деградаций	Бинарная проверка “нормы”	Контроль устойчивости и пределов ресурса
Ориентация	Поведение системы	Обязательства перед клиентом	Физические и операционные пределы
Пример для API	p50/p75 latency = baseline	latency _{p99} ≤ 300 ms	Nginx timeout = 2000 ms
Пример для batch	median job = baseline	job ≤ SLA window	max job = 120 min
Пример для EBS	median TX wait	TX wait ≤ SLA	session pool = [0, 300]
Роль в UAM	Основа нормализации деградаций	Жёсткие штрафы за нарушение	Плавная штрафная зона перед отказом

Таблица 1: Сравнение подходов нормализации в UAM

7. Типы систем и рекомендуемые метрики

Ниже приведены рекомендуемые метрики и веса для различных категорий систем. Значения являются шаблоном и адаптируются под конкретный ландшафт.

7.1. API-сервисы и платёжные шлюзы

Метрика	Обозначение	Вес w_j
Success Rate	N_{succ}	0.40
Latency p_{99}	N_{lat}	0.30
5xx Ratio	N_{err}	0.20
Очереди / backlog	N_{queue}	0.10

$$A_{\text{API}} = 0.4N_{\text{succ}} + 0.3N_{\text{lat}} + 0.2N_{\text{err}} + 0.1N_{\text{queue}}.$$

7.2. Банковские API

Метрика	Обозначение	Вес
Success Rate	N_{succ}	0.50
Timeout Ratio	N_{timeout}	0.20
Connection Failures	N_{conn}	0.20
Backlog / Queue	N_{queue}	0.10

7.3. Криптографические сервисы

Метрика	Обозначение	Вес
HSM Online	N_{hsm}	0.40
Success Rate	N_{succ}	0.30
Signing Time	N_{sign}	0.20
Slot Availability	N_{slot}	0.10

7.4. Batch-процессы

Метрика	Обозначение	Вес
Завершено в окно	N_{deadline}	0.60
Ошибки job	N_{error}	0.20
Backlog	N_{queue}	0.20

7.5. ERP / 1C:ERP

Метрика	Обозначение	Вес
Время выполнения запросов	N_{req}	0.40
Загрузка рабочих процессов (RPH)	N_{rph}	0.25
Очередь запросов	N_{queue}	0.20
DB Latency / Blocking	N_{db}	0.15

7.6. ERP / SAP S/4HANA

Метрика	Обозначение	Вес
Dialog Response Time	N_{dialog}	0.40
Work Process Utilization	N_{wp}	0.25
Queue Length (SM50/SM66)	N_{queue}	0.20
HANA DB Latency / Locks	N_{hana}	0.15

7.7. ERP / Oracle EBS

Метрика	Обозначение	Вес
Active Concurrent Managers	N_{cm}	0.35
DB Wait Events	N_{dbwait}	0.25
Session Pool Usage	N_{sess}	0.20
Workflow Lag	N_{wf}	0.20

7.8. SAP IDoc / EDI Интеграции

Метрика	Обозначение	Вес
Время обработки IDoc (latency)	N_{idoc}	0.45
Ошибка обработки (status 51/68)	N_{err}	0.30
Размер очереди IDoc	N_{queue}	0.15
Lock Conflicts / DB Blocking	N_{lock}	0.10

8. Доступность контура

8.1. Определение контура

Под **контуром** в модели UAM понимается связанная совокупность информационных систем, сервисов и процессов, образующая *единый бизнес-поток*, в рамках которого пользователь или внешний контрагент получает конечный результат.

Контур обладает следующими свойствами:

- **функциональная цельность** — все его части обеспечивают одну бизнес-функцию;
- **логическая последовательность** — элементы контура вызываются или используются в определённом порядке;
- **техническая взаимозависимость** — отказ одной системы приводит к снижению доступности всего контура;
- **измеримость** — доступность каждого компонента можно выразить числом $A_i \in [0, 1]$.

Таким образом, контур — это не инфраструктура и не архитектура как таковая, а **бизнес-ориентированная цепочка зависимостей**.

8.2. Состав контура

Каждый контур включает три слоя:

1. Front-системы (front-line):

- API, веб-методы, интеграционные шлюзы,
- мобильные или офисные интерфейсы,
- внешние REST/gRPC сервисы.

2. Логические сервисы (mid-layer):

- CRM/ERP-модули,
- банковские API, платёжники, биллинговые узлы,
- криптографические сервисы (подпись, шифрование),
- workflow и очереди сообщений.

3. Фоновые процессы (back-office):

- batch-процессы,

- отчётность,
- периодические расчёты,
- системные транзакции ERP.

Каждый компонент контура имеет собственную доступность A_i , вычисляемую в соответствии с правилами UAM.

8.3. Весовой коэффициент системы в контуре

Каждая система имеет вес W_i — коэффициент её влияния на итоговую работу контура. Он определяется на основании:

- **Критичности** компонента (может ли контур функционировать без него);
- **Временной зависимости** (участвует ли система в онлайн-части или в итоговых расчётах);
- **Частоты использования** (сколько запросов или транзакций проходит через систему);
- **Глубины в цепочке** (расположена ближе к клиенту или глубже в ядре);
- **Степени связности** (сколько других подсистем завязано на неё).

Формально:

$$\sum_{i=1}^n W_i = 1, \quad W_i \geq 0.$$

Это обеспечивает корректность интерпретации A_{contour} как доступности целого бизнес-процесса.

8.4. Формула доступности контура

$$A_{\text{contour}} = \sum_{i=1}^n W_i A_i,$$

где:

- A_i — доступность подсистемы (нормализованная в UAM),
- W_i — вес подсистемы в данном контуре.

8.5. Пример структуры контура

Рассмотрим контур проведения платежа:

- Платёжный шлюз — 0.25
- Банковское API — 0.25
- Подписание документов (HSM) — 0.15
- Batch/отчёты (формирование выписок) — 0.20
- ERP (Oracle EBS / SAP / 1C) — 0.15

Тогда доступность контура:

$$A_{\text{pay}} = 0.25A_{\text{gateway}} + 0.25A_{\text{bankAPI}} + 0.15A_{\text{sign}} + 0.20A_{\text{batch}} + 0.15A_{\text{ERP}}.$$

8.6. Почему доступность контура важнее доступности отдельных систем

В реальных бизнес-процессах:

- клиенту всё равно, что ERP доступно на 99.99%, если банковское API работает на 80%;
- отказ даже одной неключевой системы (batch) влияет на entire chain;
- наличие весов позволяет реалистично учитывать вклад каждого компонента;
- A_{contour} даёт руководству показатель, связанный с бизнес-ценностью, а не технической.

Таким образом, контур является **основным уровнем агрегирования доступности** в Unified Availability Model.

9. Сквозной пример расчёта доступности контура

В данном разделе представлен синтетический, но реалистичный пример применения Unified Availability Model (UAM) к бизнес-процессу, состоящему из трёх гетерогенных систем:

- Web-сайт (frontend + backend),
- Банк-клиент API (REST шлюз банка),

- ERP–система 1С (учёт и обмен документами).

Такой контур типичен для сценариев, где пользователь формирует документ или платёж на сайте, затем веб-сервис взаимодействует с банковским API, а итоговые данные поступают в 1С для последующей обработки.

9.1. Метрики подсистем

9.1.1. Web–сайт

Метрика	Обозначение	Вес
Success Rate	N_{succ}	0.40
Latency p_{95}	N_{lat}	0.30
5xx Errors	N_{err}	0.20
DB/Cache backlog	N_{backlog}	0.10

Таблица 2: Метрики Web–сайта

Фактические значения:

$$\text{succ} = 98.2\%, \quad p_{95} = 420 \text{ ms}, \quad 5xx = 1.8\%, \quad \text{backlog} = 250.$$

Baseline / SLA:

$$B_{\text{lat}} = 210 \text{ ms}, \quad SLA = 350 \text{ ms}, \quad L_{\text{max}} = 2000 \text{ ms}, \quad D_{\text{max}} = 1.8 B_{\text{lat}} = 380.$$

Нормализация:

$$N_{\text{succ}} = 0.82, \quad N_{\text{lat}} = \frac{380}{420} = 0.90,$$

$$N_{\text{err}} \approx 0.60, \quad N_{\text{backlog}} = 0.75.$$

Итоговая доступность Web:

$$A_{\text{web}} = 0.40 \cdot 0.82 + 0.30 \cdot 0.90 + 0.20 \cdot 0.60 + 0.10 \cdot 0.75 = 0.79.$$

9.1.2. Банк–клиент API

Метрика	Обозначение	Вес
Success Rate	N_{succ}	0.50
Timeout Ratio	N_{timeout}	0.20
Connection Failures	N_{conn}	0.20
Queue / Backpressure	N_{queue}	0.10

Таблица 3: Метрики банк–клиент API

Фактические значения:

$$\text{succ} = 99.1\%, \quad \text{timeout} = 0.7\%, \quad \text{connfail} = 0.3\%, \quad \text{queue} = 120.$$

Baseline / SLA:

$$B_{\text{succ}} = 99.6\%, \quad SLA_{\text{succ}} = 99\%, \quad SLA_{\text{timeout}} = 1.0\%.$$

Нормализация (укороченная):

$$N_{\text{succ}} = 0.93, \quad N_{\text{timeout}} = 1, \quad N_{\text{conn}} = 0.92, \quad N_{\text{queue}} = 0.85.$$

Доступность API:

$$A_{\text{api}} = 0.5 \cdot 0.93 + 0.2 \cdot 1 + 0.2 \cdot 0.92 + 0.1 \cdot 0.85 = 0.93.$$

9.1.3. 1C ERP

Метрика	Обозначение	Вес
Blocking Transactions	N_{blk}	0.35
DB Wait Events	N_{wait}	0.25
Job Lag	N_{lag}	0.20
Exchange Queue Lag	N_{ex}	0.20

Таблица 4: Метрики 1C ERP

Фактические значения:

$$\text{blk} = 12, \quad \text{wait} = 45 \text{ ms}, \quad \text{lag} = 6 \text{ min}, \quad \text{queue} = 240.$$

Baseline / SLA:

$$B_{\text{wait}} = 14 \text{ ms}, \quad SLA_{\text{wait}} = 50 \text{ ms}, \quad W_{\text{max}} = 300 \text{ ms}.$$

Упрощённая нормализация:

$$N_{\text{blk}} = 0.80, \quad N_{\text{wait}} = 0.92, \quad N_{\text{lag}} = 0.88, \quad N_{\text{ex}} = 0.75.$$

Доступность 1С:

$$A_{1c} = 0.35 \cdot 0.80 + 0.25 \cdot 0.92 + 0.20 \cdot 0.88 + 0.20 \cdot 0.75 = 0.84.$$

9.2. Расчёт доступности контура

Веса систем:

$$W_{\text{web}} = 0.30, \quad W_{\text{api}} = 0.40, \quad W_{1c} = 0.30.$$

Итоговая доступность:

$$A_{\text{contour}} = 0.30 \cdot 0.79 + 0.40 \cdot 0.93 + 0.30 \cdot 0.84 = 0.857.$$

9.3. Интерпретация

Полученное значение

$$A_{\text{contour}} = 0.857$$

указывает на наличие деградации контура, обусловленной главным образом Web-сервисом. Банк-клиент API является наиболее стабильной частью контура, 1С ERP находится в умеренной зоне риска из-за увеличенных очередей обмена.

Данный пример демонстрирует практическую применимость UAM для систем, которые в традиционных моделях не могут быть объединены в единый SLA или общую шкалу доступности.

10. Реализация в мониторинге

10.1. Prometheus

- recording rules для нормализации,
- функции `clamp`, `rate`, `scalar`,
- итоговые правила для A_i и A_{contour} .

10.2. Zabbix

- dependent items для нормализации,
- формулы пользовательских элементов,
- триггеры на пороги $A_i < 0.8$, $A_i < 0.5$.

10.3. Grafana

- панели gauge,
- композитные панели SLA,
- цветовая логика 4 уровней: Green / Yellow / Orange / Red.

11. Иерархическая модель когерентности уровней (расширение к Unified Availability Model)

11.1. Введение

В сложных ИТ-ландшафтах доступность бизнес-процесса зависит не только от состояния отдельных подсистем, но и от **иерархии уровней зависимости**: инфраструктура ограничивает прикладные сервисы, а прикладные сервисы — бизнес-логику.

Даже идеально работающий верхний уровень не способен компенсировать деградацию нижнего. Это формирует фундаментальный принцип:

Доступность контура ограничена минимальной согласованностью его базовых уровней.

Для формализации этого принципа вводится **Иерархическая Модель Когерентности Уровней (ИКУ)**, являющаяся расширением UAM.

11.2. Архитектура уровней

Мы выделяем три уровня наблюдаемости:

1. **Инфраструктурный уровень (Infra)** сеть, TCP latency, ping, CPU, RAM, диски, storage, кластеры.
2. **Прикладной уровень (App)** HTTP 5xx, TNS, RPS, thread-pool saturation, application backlog.

3. **Бизнес-уровень (Biz)** скорость обработки транзакций, SLA бизнес-операций, задержка бизнес-цепочек, глубина workflow.

Каждый уровень имеет собственную доступность:

$$H_{\text{infra}}, H_{\text{app}}, H_{\text{biz}} \in [0, 1].$$

11.3. Проблема фиксированных весов и динамическая перенормировка

В практической эксплуатации споры между инженерами почти всегда сводятся к вопросам вида:

- «Почему ping важнее latency?»
- «Почему success-rate должен весить больше, чем backlog?»
- «Почему бизнес-показатели не доминируют над техническими?»

Эти конфликты возникают не из-за самих метрик, а из-за **фиксированных весов**, которые не отражают реальное состояние уровней.

Решение: в рамках уровня веса становятся *динамическими*— они автоматически увеличиваются для деградирующих метрик и уменьшаются для стабильных.

Это устраняет «споры о важности» и отражает естественное поведение системы.

11.4. Мягкая иерархия уровней (Multiplicative Coherence Model)

Плавное влияние уровней друг на друга описывается формулой:

$$H_{\text{final}} = H_{\text{infra}} \cdot (\alpha H_{\text{app}} + (1 - \alpha) H_{\text{biz}}), \quad (6)$$

где:

- H_{infra} — фундамент системы,
- H_{app} — реакция приложения на состояние инфраструктуры,
- H_{biz} — качество выполнения бизнес-операций,
- $\alpha \in [0, 1]$ — чувствительность сервисного уровня.

Смысл: если инфраструктура просела до 0.5, то даже идеальные App и Biz не могут дать более $0.5 \times 1 = 0.5$.

11.5. Жёсткая иерархия (Foundational Limit Model)

Для финансовых систем, банковских API, электронных платежей и ERP-операций действует жёсткое правило:

если фундаментальная инфраструктура недоступна — весь контур недоступен.

Это формализуется предельным оператором:

$$H_{\text{total}} = \min (H_{\text{infra}}, H_{\text{weighted-app-biz}}) . \quad (7)$$

Примеры:

- если сеть падает — ERP не спасёт контур;
- если DNS недоступен — API полностью недоступно;
- если storage деградирует — batch-процессы не запускаются.

11.6. Динамическая логика весов внутри уровней

Для уровня L веса метрик пересчитываются по правилу:

$$w_j^{\text{new}} = \frac{w_j^{\text{base}}}{1 + \gamma \cdot \Delta_j} , \quad (8)$$

где:

- w_j^{base} — базовый вес метрики,
- Δ_j — степень её отклонения от нормы,
- γ — коэффициент чувствительности уровня.

Таким образом, проблемные метрики автоматически становятся весомее, и необходимость вручную «торговаться о важности ping» исчезает.

12. Заключение

Unified Availability Model (UAM) предоставляет формальный и расширяемый подход к расчёту доступности разнородных систем. Методика позволяет:

- реалистично сравнивать разные классы приложений;
- строить единый показатель доступности контура;

- интегрировать Prometheus, Zabbix, Grafana;
- адаптировать модель под любые новые системы.

Данная методика предназначена для практического использования в высоконагруженных и критичных для бизнеса ИТ-ландшафтах.

Аннотация к Приложению А

В Приложении А рассматриваются альтернативные подходы к агрегации метрик в разнородных ИТ-системах. Показано, почему нечеткая логика применима **только внутри** отдельных уровней (Infra, App, Biz), где границы нормальности размыты и опираются на экспертные оценки, и почему её использование **между уровнями** недопустимо из-за строгой иерархической зависимости: верхний уровень не может быть более доступным, чем нижний.

Также обоснована необходимость динамической перенормировки весов, введены мягкая и жёсткая модели иерархической когерентности и кратко разобраны ограничения нейросетевых методов, требующих периодического или полного переобучения при изменении нагрузки.

Приложение А формирует методологическое обоснование архитектуры UAM v2 и формализует инженерную логику построения многослойной модели доступности.

Приложение А. Fuzzy-логика и Нейросети в оценке доступности: область применения и ограничения

А.1. Почему fuzzy допустим *внутри* уровней (Infra / App / Biz)

А.1.1. Причины применения

Низкоуровневые метрики часто имеют размытые или неоднозначные критерии «нормальности». Типичные примеры:

- дневная и ночная латентность могут различаться в 2–3 раза при одинаковой норме;
- допустимый ping по-разному трактуется разными командами;
- backlog не имеет универсального порога, применимого к любым нагрузкам.

Общая проблема: **границы между «хорошо», «нормально» и «плохо» имеют диапозонный характер.**

Fuzzy-логика естественно работает в условиях:

- размытых порогов,
- экспертных оценок,
- неоднозначных режимов работы.

То есть fuzzy решает прежде всего эпистемическую¹ проблему интерпретации метрик, а не математическую задачу.

А.1.2. Что даёт применение fuzzy внутри уровня

- **Снятие спорных трактовок.** Размытые пороги переводят субъективные обсуждения в формальные диапазоны.
- **Плавность реакции.** Метрика не падает скачком, а изменяется непрерывно.
- **Устойчивость к шуму.** Малые колебания latency или CPU не вызывают ложных срабатываний.
- **Лучшая предварительная нормализация.** Fuzzy используется на этапе подготовки данных перед вычислением N_{ij} .
- **Гибкость без потери контроля.** UAM остаётся строгой моделью, а fuzzy выступает как инструмент предобработки сырых метрик.

Таким образом, fuzzy применим только как **внутриуровневая смягчающая оболочка** для неоднозначных показателей Infra, App и Biz.

А.2. Почему fuzzy нельзя использовать между уровнями (Infra → App → Biz)

А.2.1. Причины отказа

Иерархия уровней обладает жёсткой зависимостью:

- деградация инфраструктуры неизбежно приводит к деградации приложений;
- недоступность приложений делает невозможными бизнес-операции;
- верхний уровень не может быть лучше нижнего.

Фундаментальный принцип:

когерентность верхнего уровня ограничена когерентностью нижнего.

Если применить fuzzy на стыке уровней, возникают:

¹ в том числе человеческую

- размытые зависимости,
- противоречивые оценки,
- риск «косметически нормального Biz» при критически проблемном Infra,
- потеря строгой интерпретации,
- рекурсивные колебания пересчётов.

Иначе говоря, fuzzy делает уровни «слишком равными» — что **структурно неверно** и противоречит их иерархической природе.

А.2.2. Что даёт отказ от fuzzy между уровнями

- **Строгая иерархия ограничений.** Верхний уровень не может «обмануть» фундамент ниже.
- **Бизнес-ориентированная прозрачность.** Менеджерам не требуется понимание сложных fuzzy-градиаций.
- **Корректное распространение деградаций.** Если инфраструктура деградирует, App и Biz не могут демонстрировать «условно нормальное» состояние.
- **Реалистичная модель доступности.** Операторы $\min(\cdot)$ и произведение обеспечивают реалистичность оценки.
- **Совместимость с UAM и COE.** Модель согласована с принципом онтологической ограниченности уровней.
- **Отсутствие рекурсивной неопределённости.** Fuzzy между уровнями приводит к неуправляемой перенормировке.

А.3. Почему мы не используем нейросети

Нейросетевые методы прогнозирования и агрегирования также были рассмотрены, но сознательно отвергнуты.

Причины:

- изменение нагрузки требует **периодического или даже полного переобучения**, что неприемлемо для метрики, используемой в бизнес-отчётности;
- нейросети обладают *низкой интерпретируемостью решений* (low interpretability);
- модель доступности должна быть аудируемой и повторяемой, что невозможно при стохастических весах;

- нейросеть оптимизирует предсказание, а не ответственность за SLA.

Таким образом, нейросети непригодны для **руководящей метрики доступности**.

A.4. Вывод

Fuzzy-логика может применяться только для нормализации отдельных метрик внутри уровней (Infra, App, Biz) — и не должна использоваться для агрегации уровней.

Это осознанный инженерный выбор:

- метрики имеют размытые границы → fuzzy подходит;
- уровни имеют жёсткую иерархию → fuzzy противопоказан.

Список литературы

- [1] Алексей Алексеевич Неклюдов, *Философия Дискретного Бытия. Манифест*, Zenodo (2025). [doi:10.5281/zenodo.17572909](https://doi.org/10.5281/zenodo.17572909).