СТУДЕНТ: Анненков Алексей Николаевич
Группа: 238
Дата загрузки работы: 12.01.2026

Подтверждаю, что данная работа является оригинальной, выполненной мною самостоятельно и в установленные сроки.

Код с реализацией экспериментов находится в репозитории
https://github.com/alekseiannenkov/Fuzzy-logic.

Самостоятельная работа №2

Задание №1

Студентам НИУ ВШЭ в рамках выполнения домашней работы по дисциплине *«Интеллектуальный анализ данных»* было необходимо обучить свою нейросеть и сравнить её с baseline моделями. За час до дедлайна студенты осознали, что, увлёкшись выбиванием метрик на собственной модели (и получив на ней значение $Score = 0.93$), они совсем забыли обучить baseline сети и уже никак не успеют это сделать. Помогите студентам, используя модель нечёткого вывода Мамдани, оценить метрики baseline-моделей и на основе этого сделать существенные выводы о применимости своей модели. Для этого рассмотрите систему с тремя входами и общим числом правил ≥ 9.

Решение

Выходом модели будет служить метрика Score (коэффициент качество модели). Формализуем её термы при помощи нечётких интервалов:
Score:

- Low(0, 0, 50, 60);
- Medium(50, 60, 70, 80);
- High(70, 80, 90, 100);
- Excellent(80, 90, 100, 100).

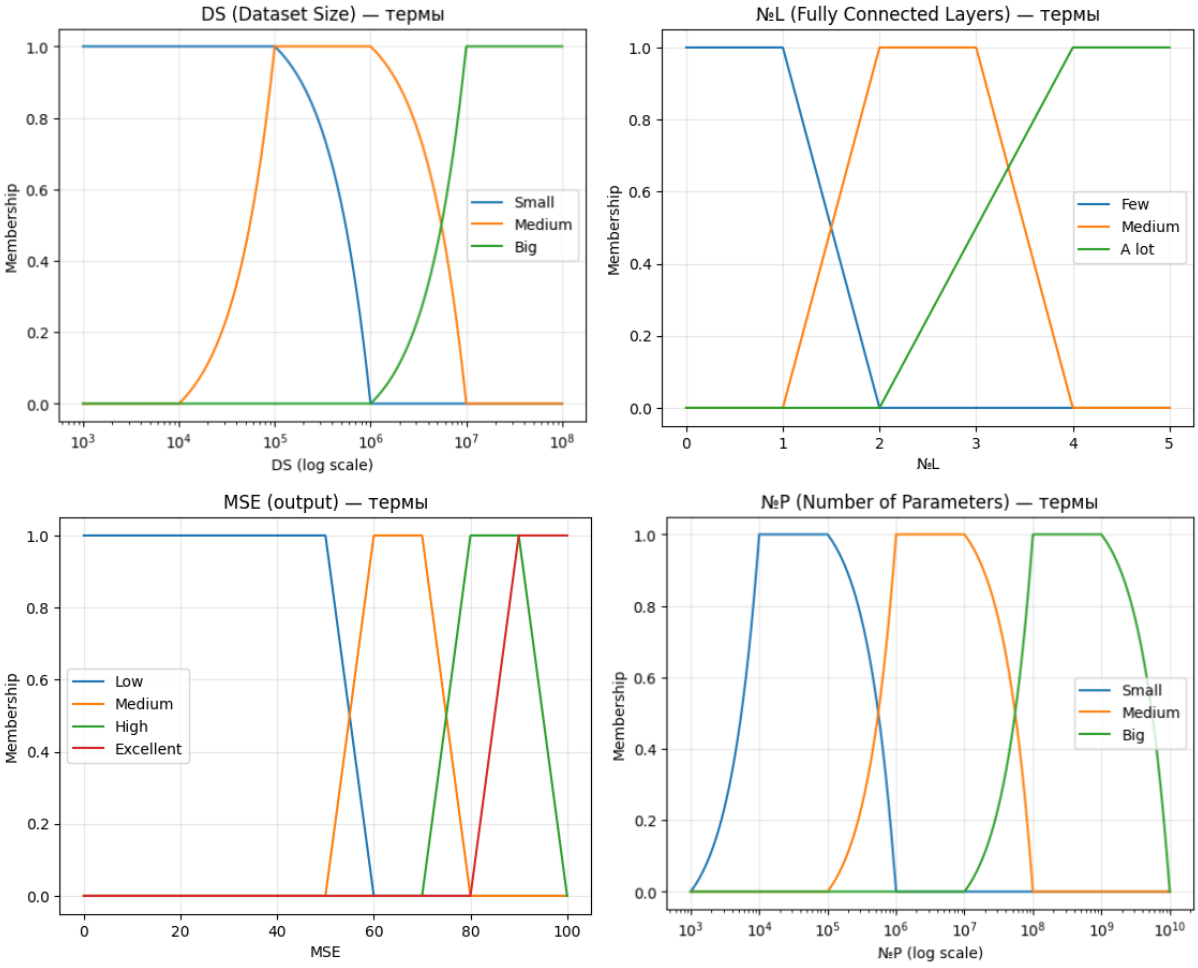
Рассмотрим следующие факторы, влияющие на метрику (входные переменные):

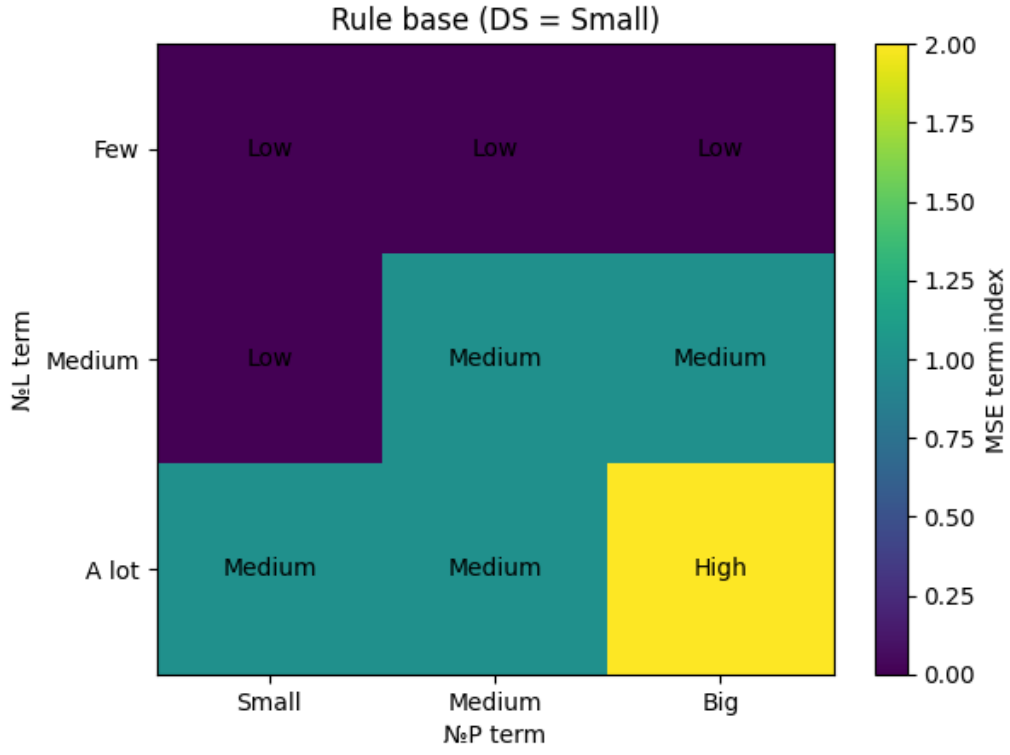
1. **DS (Dataset Size):**
   - Small (10^3, 10^3, 10^5, 10^6);
   - Medium (10^4, 10^5, 10^6, 10^7);
   - Big (10^6, 10^7, 10^8, 10^8).
2. **№L (Number of Fully Connected Layers):**
   - Few (0, 0, 1, 2);
   - Medium (1, 2, 3, 4);
   - A lot (2, 4, 5, 5).
3. **№P (Number of Parameters):**
   - Small (10^3, 10^4, 10^5, 10^6);
   - Medium (10^5, 10^6, 10^7, 10^8);
   - Big (10^7, 10^8, 10^9, 10^{10}).

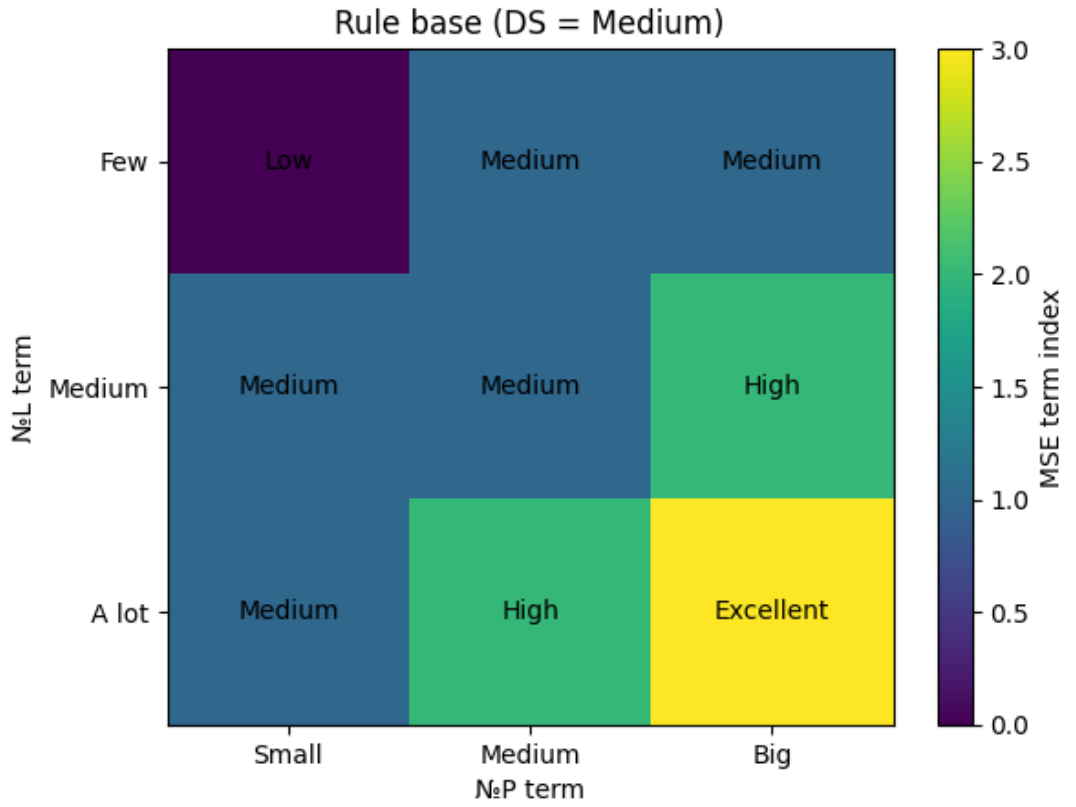
   Ниже — таблица с правилами Мамдани:
   IF DS is _ AND №L is _ AND №P is _ THEN Score is _.

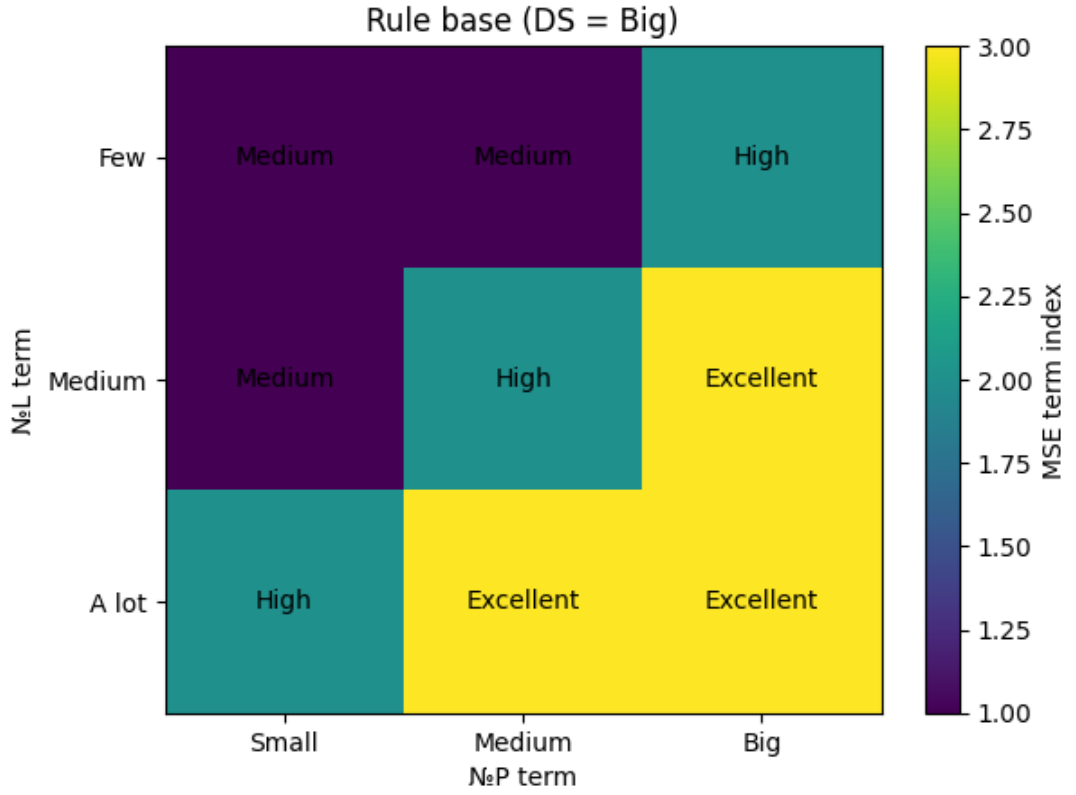
DS	№L	№P	Score (output term)
Small	Few	Small	Low
Small	Few	Medium	Low
Small	Few	Big	Low
Small	Medium	Small	Low
Small	Medium	Medium	Medium
Small	Medium	Big	Medium
Small	A lot	Small	Medium
Small	A lot	Medium	Medium
Small	A lot	Big	High
Medium	Few	Small	Low
Medium	Few	Medium	Medium
Medium	Few	Big	Medium
Medium	Medium	Small	Medium
Medium	Medium	Medium	Medium
Medium	Medium	Big	High
Medium	A lot	Small	Medium
Medium	A lot	Medium	High
Medium	A lot	Big	Excellent
Big	Few	Small	Medium
Big	Few	Medium	Medium
Big	Few	Big	High
Big	Medium	Small	Medium
Big	Medium	Medium	High
Big	Medium	Big	Excellent
Big	A lot	Small	High
Big	A lot	Medium	Excellent
Big	A lot	Big	Excellent

Визуализируем термы и тепловые карты для описанных правил.






Rule base (DS = Medium)
Few
Medium
A lot
№L term
Low
Medium
Medium
Medium
Medium
High
Medium
High
Excellent
Small
Medium
Big
№P term
3.0
2.5
2.0
1.5
1.0
0.5
0.0
MSE term index


Rule base (DS = Big)
Few
Medium
A lot
№L term
Medium
Medium
High
Medium
High
Excellent
High
Excellent
Excellent
Small
Medium
Big
№P term
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
MSE term index

В качестве baseline-решений рассмотрим следующие популярные архитектуры нейросетей, параметры которых укладываются в заранее заданные рамки по числу полносвязных слоёв и общему количеству параметров:

1. **LeNet-5** — компактная сверточная сеть с тремя полносвязными слоями и приблизительно $6 \cdot 10^4$ параметрами;
2. **ResNet-18** — современная архитектура с остаточными связями, содержащая один полносвязный слой и около $1.17 \cdot 10^7$ параметров;
3. **AlexNet** — классическая сверточная сеть с тремя полносвязными слоями и примерно $6 \cdot 10^7$ параметрами;
4. **VGG-16** — глубокая сверточная архитектура с тремя полносвязными слоями и порядка $1.38 \cdot 10^8$ параметров.

Для каждой baseline-модели оценим значение метрики Score, используя модель нечёткого вывода Мамдани и аддитивную агрегацию Барта Коско.

Для этого выполним следующие шаги:

1. Рассмотрим несколько характерных размеров обучающего датасета:

$$DS \in \{10^4,\ 10^5,\ 5 \cdot 10^5,\ 10^6,\ 5 \cdot 10^6,\ 10^7,\ 5 \cdot 10^7\}.$$

   Для каждого значения DS вычислим степени принадлежности соответствующим нечётким термам входной переменной *Dataset Size* и определим все выстреливающие правила Мамдани.

2. Для каждого выстреливающего правила вычислим степень выстреливания (firing degree). В соответствии с аддитивной агрегацией Барта Коско степень выстреливания правила при входе $x = (DS = x_1,\ L = x_2,\ P = x_3)$ определяется как

$$w = \min\big(\mu_{DS}(x_1),\ \mu_L(x_2),\ \mu_P(x_3)\big),$$

   где μ_{DS}, μ_L и μ_P — функции принадлежности соответствующих нечётких термов.

3. Для каждого выстреливающего правила масштабируем соответствующий терм выходной переменной Score, умножая его функцию принадлежности на вычисленный вес w. Далее агрегируем полученные фигуры, беря максимум значений функций принадлежности в каждой точке выходного универсума.

4. Получив итоговую агрегированную нечёткую фигуру для выходной переменной Score, выполняем дефаззификацию методом центра тяжести и тем самым получаем численную оценку Score для фиксированного размера датасета.

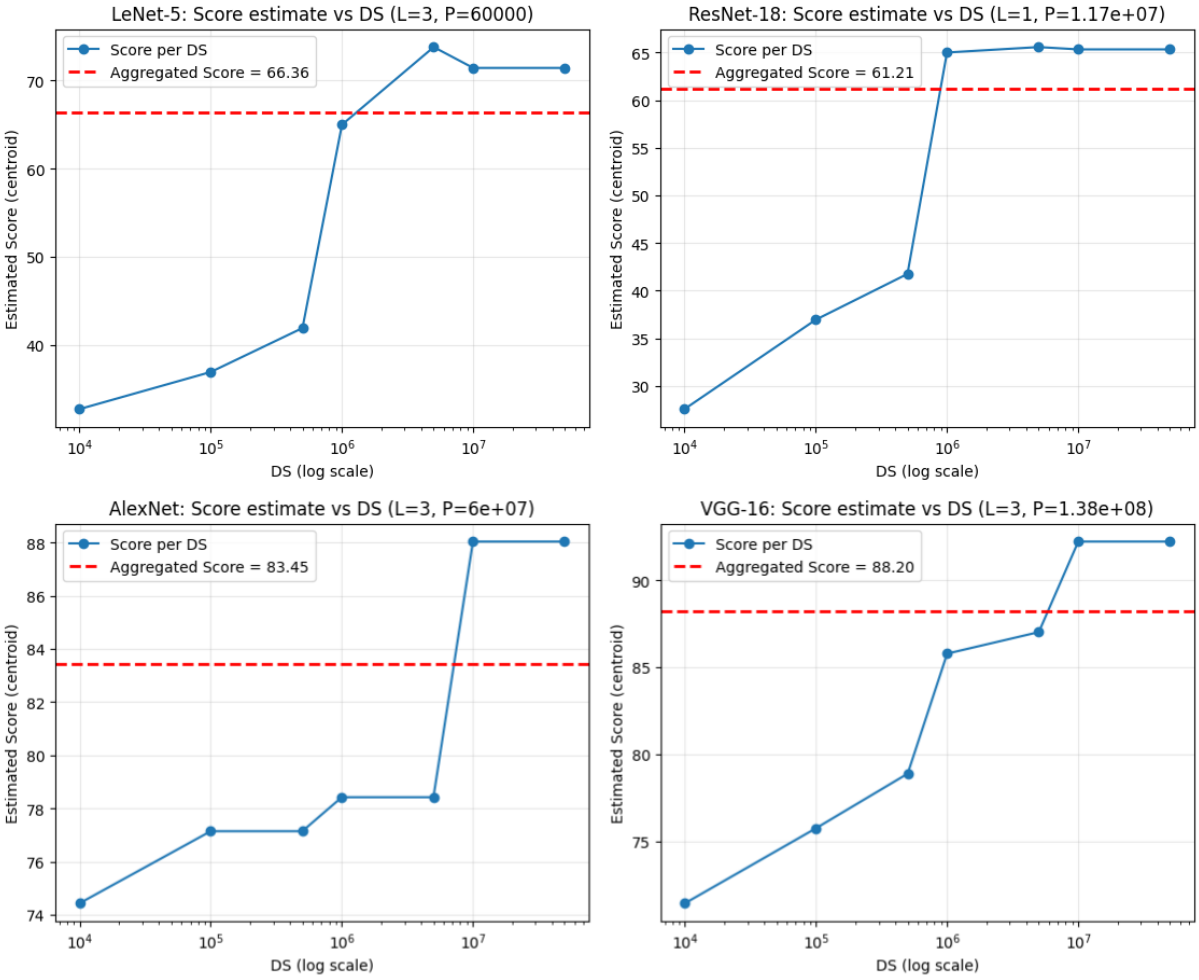
5. Поскольку для каждого значения DS получается собственная оценка Score, агрегируем их в одну финальную оценку качества baseline-модели. Для этого используется взвешенная сумма:

$$\text{Score}_{\text{final}} = \sum_i \alpha_i \cdot \text{Score}(DS_i),$$

где коэффициенты α_i заданы следующим образом:

$$\begin{aligned}
DS &= 10^4 && \to \alpha = 0.025,\\
DS &= 10^5 && \to \alpha = 0.05,\\
DS &= 5 \cdot 10^5 && \to \alpha = 0.075,\\
DS &= 10^6 && \to \alpha = 0.1,\\
DS &= 5 \cdot 10^6 && \to \alpha = 0.2,\\
DS &= 10^7 && \to \alpha = 0.25,\\
DS &= 5 \cdot 10^7 && \to \alpha = 0.3.
\end{aligned}$$

В ходе проведения эксперимента были получены следующие результаты:



Выпишем полученные оценки и собственный результат студентов

1. **LeNet-5**: агрегированная оценка $\text{Score}_{\text{final}} \approx 66.36$.
2. **ResNet-18**: агрегированная оценка $\text{Score}_{\text{final}} \approx 61.21$.
3. **AlexNet**: агрегированная оценка $\text{Score}_{\text{final}} \approx 83.45$.
4. **VGG-16**: агрегированная оценка $\text{Score}_{\text{final}} \approx 88.20$.
5. **Свёрточная сеть студентов**: точное значение Score $= 0.93$.

С учётом того, что приведённые значения являются лишь оценками, студенты могут написать в своей домашней работе следующий вывод:«Из-за особенностей архитектуры и размеров датасетов две baseline модели (**LeNet-5** и **ResNet-18**) показали существенно худший результат, чем две оставшиеся (**AlexNet** и **VGG-16**) – Score ≈ 65 против ≈ 85 соответственно. Полученное качество нашей модели – 0.93, что позволяет считать её вполне применимой при условиях, схожих с условиями проведения домашнего задания.»

Задание №1, часть 2

Составив описанный выше отчёт, студентам казалось, что вывод является неполным с точки зрения оценки применимости модели. Перечитав собственные выкладки, они вспомнили, что не провели анализ длительности обучения моделей. Действительно, если модель обучается на несколько порядков дольше своих аналогов, может возникнуть вопрос о целесообразности её применимости в практических задачах. Со слов студентов, их модель обучалась 1 день (24 часа). Помогите студентам оценить время работы рассмотренных baseline моделей. В качестве инструмента используйте модель Сугено с двумя входами. В конце решения сделайте сравнительный анализ модели и её аналогов.

Решение

В качестве входных параметров модели рассмотрим использованное ранее число параметров, а также количество операций в секунду, которое способно выполнять используемое для проведения тензорных вычислений устройство.

1. **№P (Number of Parameters):**
   - Small (10^3, 10^4, 10^5, 10^6);

- Medium (10^5, 10^6, 10^7, 10^8);
- Big (10^7, 10^8, 10^9, 10^{10}).

2. **Perf (Performance – operations per second):**

- Bad (10^{10}, 10^{10}, 10^{12}, 10^{13}) — CPU;
- Acceptable (10^{11}, 10^{12}, 10^{13}, 10^{14}) — GPU;
- Good (10^{13}, 10^{14}, 10^{15}, 10^{15}) — ASIC.

Далее, прикинув порядковый масштаб желаемых результатов, составим следующие правила Сугено (2–ого порядка):

$$d_u = \log_{10}(P) - u_0, \qquad d_v = \log_{10}(\text{Perf}) - v_0$$

$$F = \left(1 + 0.45d_u - 0.55d_v + 0.08d_u^2 + 0.06d_v^2 - 0.04d_u d_v\right)$$

Rule	**№P term**	**Perf term**	(u_0, v_0)	**Sugeno consequent (2nd degree)**
R_{11}	Small	Bad	(5.0, 12.0)	$T = 10800 \cdot F$
R_{12}	Small	Acceptable	(5.0, 13.5)	$T = 1800 \cdot F$
R_{13}	Small	Good	(5.0, 15.0)	$T = 300 \cdot F$
R_{21}	Medium	Bad	(7.0, 12.0)	$T = 172800 \cdot F$
R_{22}	Medium	Acceptable	(7.0, 13.5)	$T = 43200 \cdot F$
R_{23}	Medium	Good	(7.0, 15.0)	$T = 7200 \cdot F$
R_{31}	Big	Bad	(9.0, 12.0)	$T = 864000 \cdot F$
R_{32}	Big	Acceptable	(9.0, 13.5)	$T = 172800 \cdot F$
R_{33}	Big	Good	(9.0, 15.0)	$T = 36000 \cdot F$

На основе приведённых правил опишем процедуру оценки времени обучения:

1. Для фиксированной baseline-модели рассмотрим три характерных класса вычислительных устройств:

$$\text{Perf} \in \{\text{CPU (Bad)},\ \text{GPU (Acceptable)},\ \text{ASIC (Good)}\}.$$

Для каждого значения Perf вычислим степени принадлежности соответствующим нечётким термам входной переменной *Performance* и определим все выстреливающие правила Сугено.

2. Аналогично предыдущей задаче, для каждого выстреливающего правила вычислим степень выстреливания (firing degree):

$$w = \min\left(\mu_P(P),\ \mu_{\text{Perf}}(\text{Perf})\right),$$

где μ_P и μ_{Perf} — функции принадлежности соответствующих нечётких термов.

3. В отличие от модели Мамдани, в модели Сугено заключение каждого правила задаётся аналитически. В данной работе используются правила Сугено второго порядка (с использованием логарифмов для избежания больших значений, возникающих из-за порядков операций в секунду), в которых время обучения аппроксимируется квадратичным полиномом по входным переменным:

$$T_i(x) = f_i(P, \text{Perf}).$$

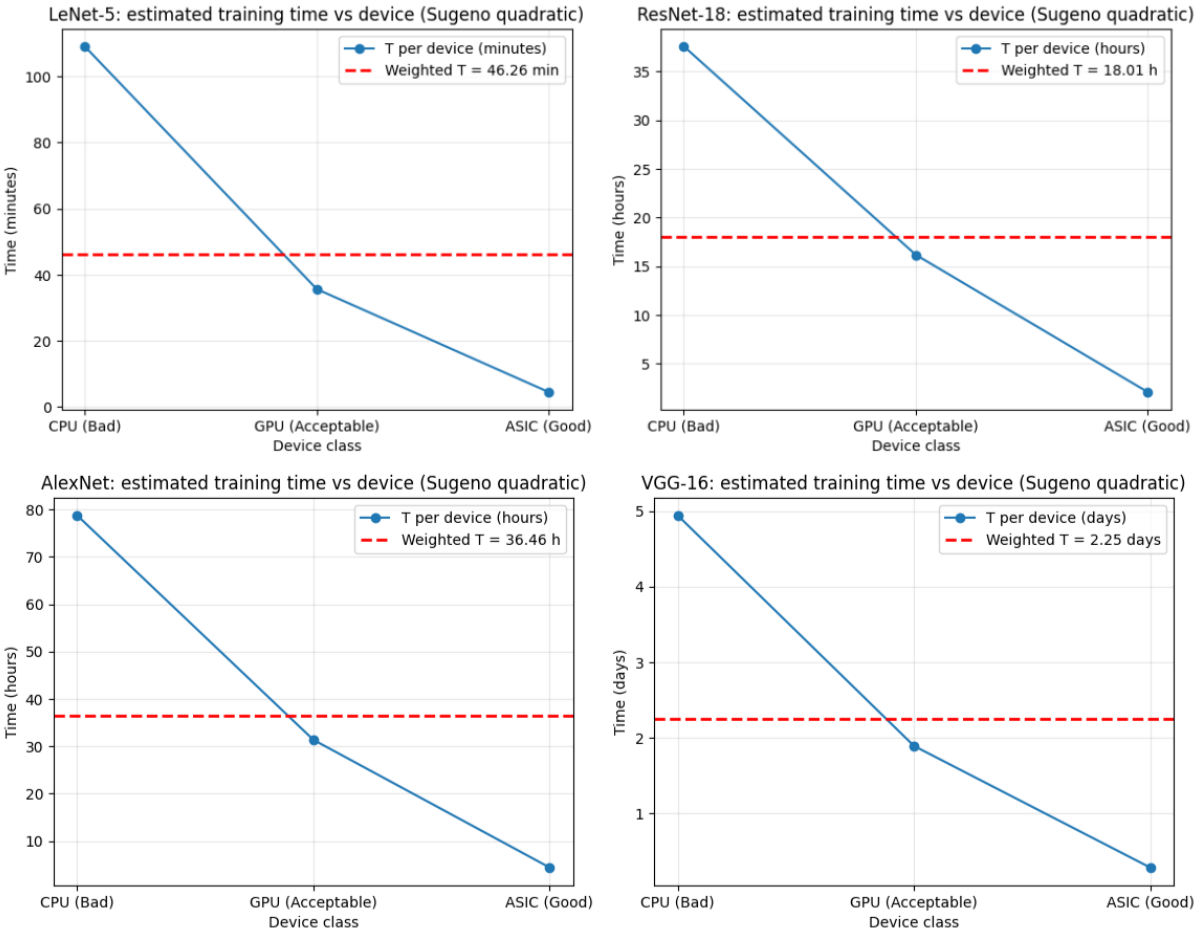
4. Для каждого выстреливающего правила вычислим значение заключения $T_i(x)$, после чего получим итоговую оценку времени обучения как взвешенное среднее:

$$T(P, \text{Perf}) = \frac{\sum_i w_i \cdot T_i(x)}{\sum_i w_i}.$$

5. Поскольку время обучения зависит от используемого вычислительного устройства, для получения финальной оценки применимости модели агрегируем результаты для различных классов устройств с помощью взвешенной суммы:

$$T_{\text{final}} = 0.25 \cdot T_{\text{CPU}} + 0.5 \cdot T_{\text{GPU}} + 0.25 \cdot T_{\text{ASIC}}.$$

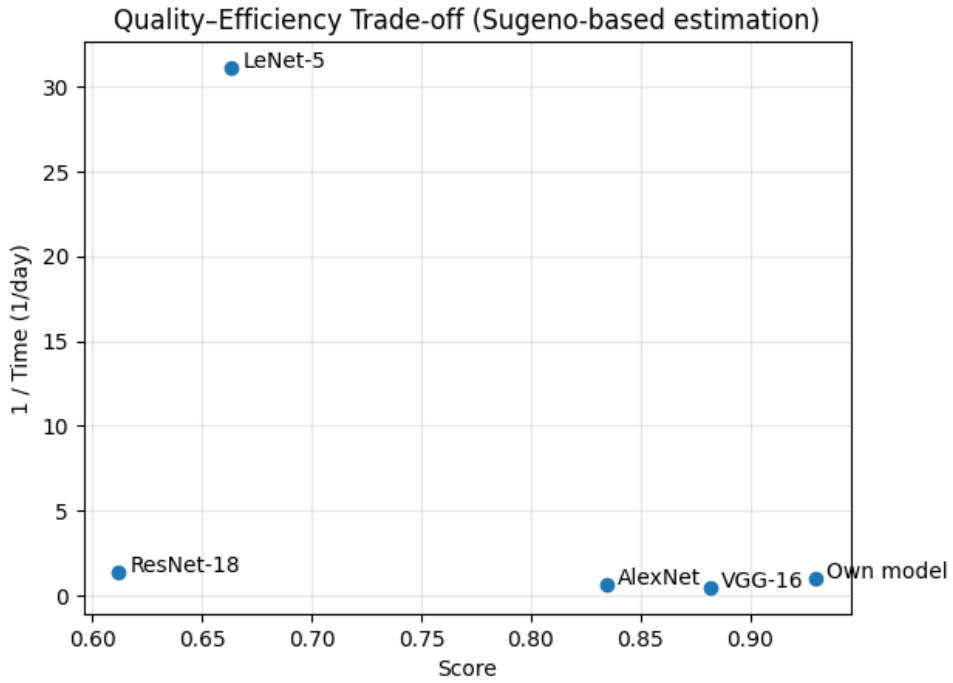
Ниже представлены полученные результаты.

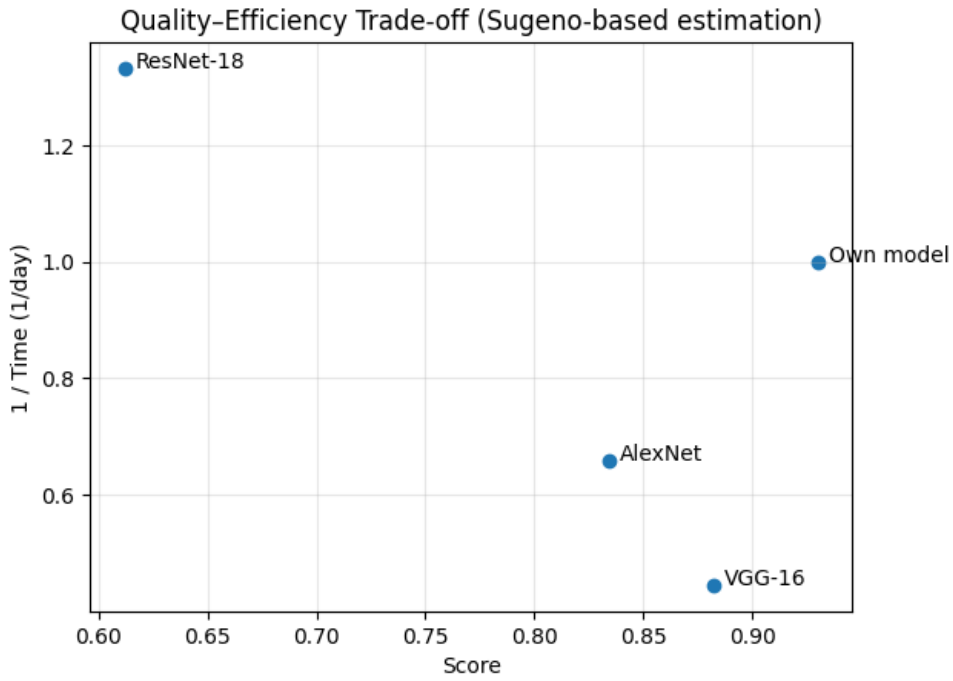


Для наглядности выпишем полученные оценки.

1. **LeNet-5**: агрегированная оценка времени обучения $T_{\text{final}} \approx 46.26$ min.
2. **ResNet-18**: агрегированная оценка времени обучения $T_{\text{final}} \approx 18.01$ h.
3. **AlexNet**: агрегированная оценка времени обучения $T_{\text{final}} \approx 36.46$ h.
4. **VGG-16**: агрегированная оценка времени обучения $T_{\text{final}} \approx 2.25$ days.
5. **Свёрточная сеть студентов**: заданная оценка времени обучения $T = 1$ day.

Изобразим графики, на котором сравним пары значений (Score, 1/Learning Time) для всех моделей – на одной графике сравним все модели, на втором уберём **LeNet-5** из-за относительного малого времени обучения для анализа остальных альтернатив.


Quality–Efficiency Trade-off (Sugeno-based estimation)
LeNet-5
ResNet-18
AlexNet
VGG-16
Own model
1 / Time (1/day)
Score
0
5
10
15
20
25
30
0.60
0.65
0.70
0.75
0.80
0.85
0.90


Quality–Efficiency Trade-off (Sugeno-based estimation)
ResNet-18
Own model
AlexNet
VGG-16
1 / Time (1/day)
Score
0.6
0.8
1.0
1.2
0.60
0.65
0.70
0.75
0.80
0.85
0.90

Приведём возможное описание полного вывода для работы студентов:

На графиках выше приведены оценки качества моделей и времени обучения. Как видно, у **LeNet-5** и **ResNet-18** метрики около 0.65, что является небольшим результатом. Однако время обучения **LeNet-5** равно 46 минутам, что на порядок меньше, чем у альтернатив. Другие свёрточные сети – в частности, наша, **AlexNet** и **VGG-16** -– показали схожие результаты как по метрике ($\approx 0.85-0.9$), так и по времени обучения (20-40 часов). Однако по обоим параметрам наша модель превзошла альтернативы из группы «высокой метрики».

С учётом вышесказанного, можно утверждать, что выбор модели должен быть основан в первую очередь на приоритетах пользователя. Если необходимо обучить модель быстро (в течение часа) – хорошо подойдёт **LeNet-5**; если же главным критерием является высокая метрика, то целесообразно использовать нашу модель – она не только имеет лучший среди альтернатив коэффициент качества, но и обучается в течение дня (что также лучше некоторых альтернатив).

Задание №2

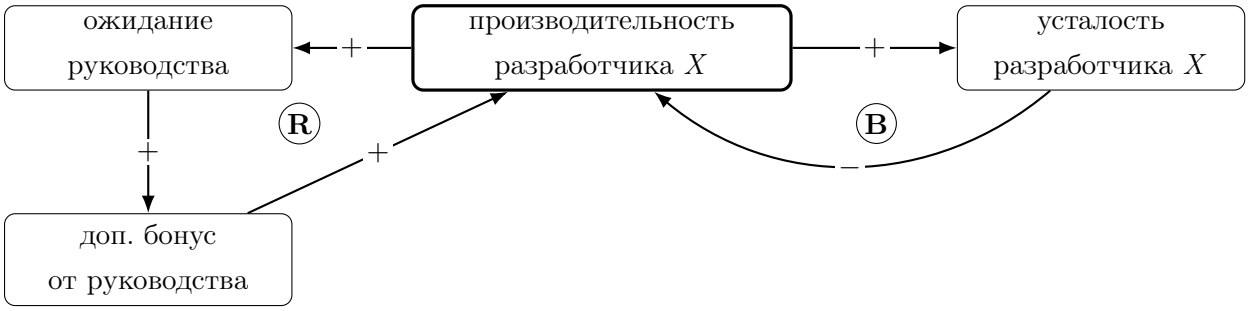
В течение 6 недель в компании наблюдается устойчивое снижение производительности разработчика X. Разработчик является ключевым звеном команды, на нём завязаны критически важные части проекта, поэтому любые отклонения в его работе существенно отражаются на общем прогрессе.

Ситуация усугубляется тем, что менее чем через месяц запланирована сдача проекта заказчику. В этих условиях становится особенно важно в кратчайшие сроки выявить причины снижения производительности и принять управленческие решения, направленные на её восстановление и стабилизацию.

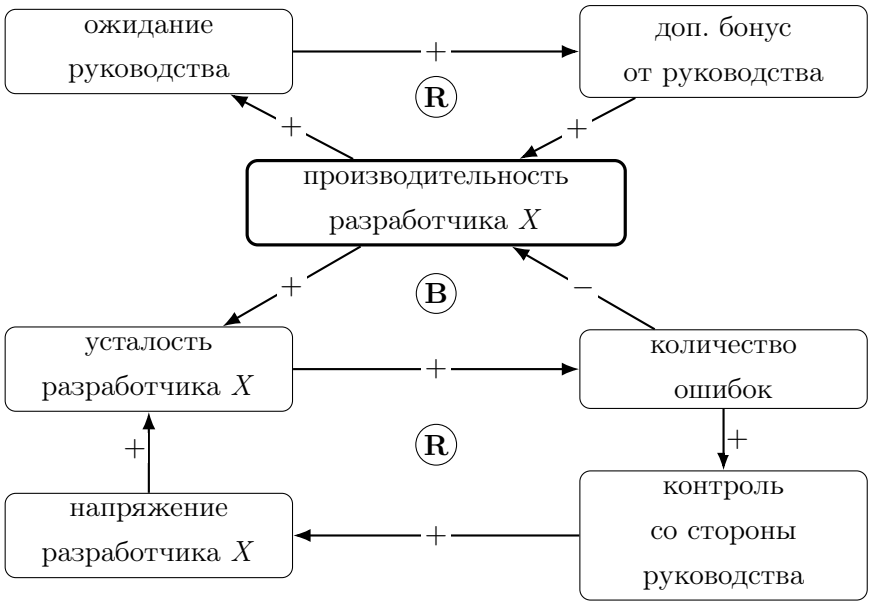
Что может быть предложено для преодоления этой ситуации? Рассмотреть Causal Loop Diagram с тремя циклами. Понять, какой цикл является доминирующим в момент рассмотрения ситуации. В качестве переменных взять интервальные нечёткие множества второго типа. Для всех переменных построить график Behaviour over time. Предложить альтернативы улучшающего вмешательства.

Решение

На семинаре была рассмотрена следующая диаграмма:



Было предложено дополнить диаграмму ещё несколькими факторами. Ниже приведена обновлённая диаграмма.


ожидание руководства
доп. бонус от руководства
производительность разработчика X
усталость разработчика X
количество ошибок
напряжение разработчика X
контроль со стороны руководства
+
+
+
+
+
+
+
−
R
B
R

Проанализируем полученную диаграмму:

1. **Главная переменная.** Центральной переменной в рассматриваемой диаграмме является *производительность разработчика X*. Именно она отражает наблюдаемую проблему и выступает итоговым показателем, на который оказывают влияние все остальные факторы.

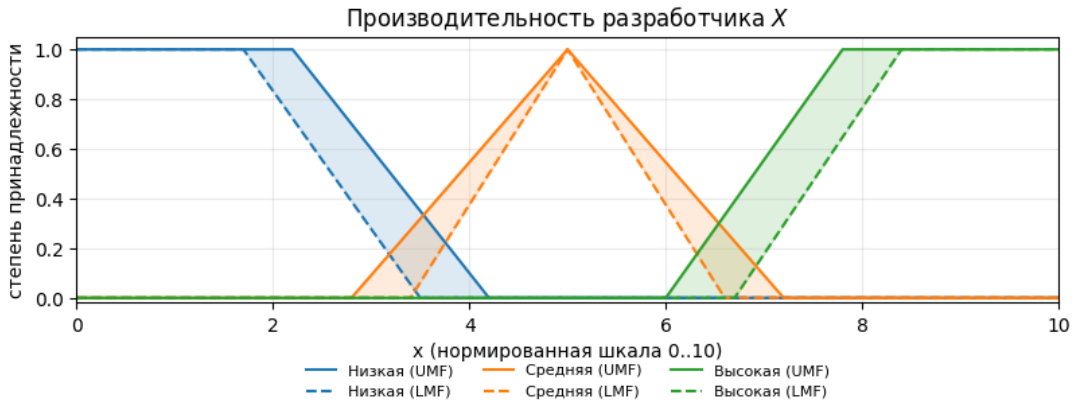
2. **Усиливающий R-цикл «ожидания–бонус–производительность».** В данном цикле рост производительности разработчика приводит к росту ожиданий со стороны руководства. Повышенные ожидания, в свою очередь, выражаются в дополнительном бонусе, который положительно влияет на мотивацию разработчика и способствует дальнейшему росту производительности. Этот цикл является усиливающим: при благоприятных условиях он поддерживает рост производительности.

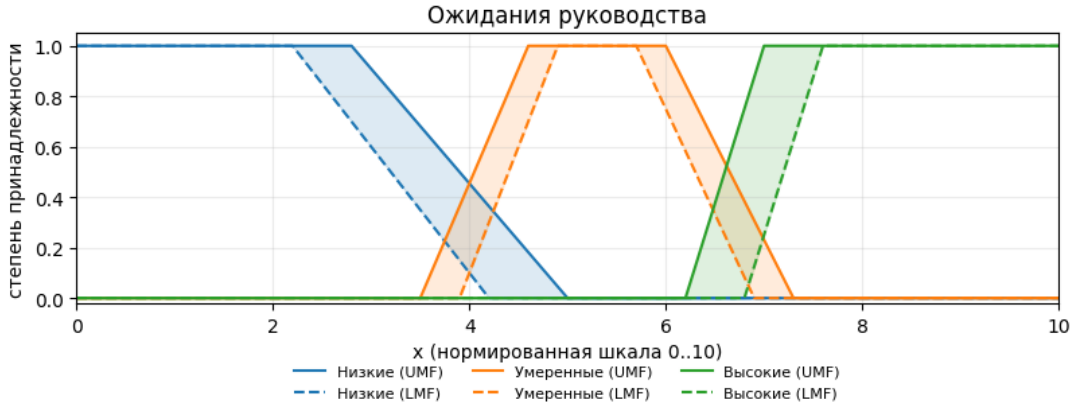
3. **Балансирующий B-цикл «производительность–усталость–ошибки».** Увеличение производительности разработчика приводит к росту усталости. Повышенная усталость увеличивает количество ошибок, а рост числа ошибок негативно сказывается на производительности. Данный цикл является балансирующим и отражает естественные ограничения человека: при чрезмерной нагрузке система сама стремится снизить уровень производительности.

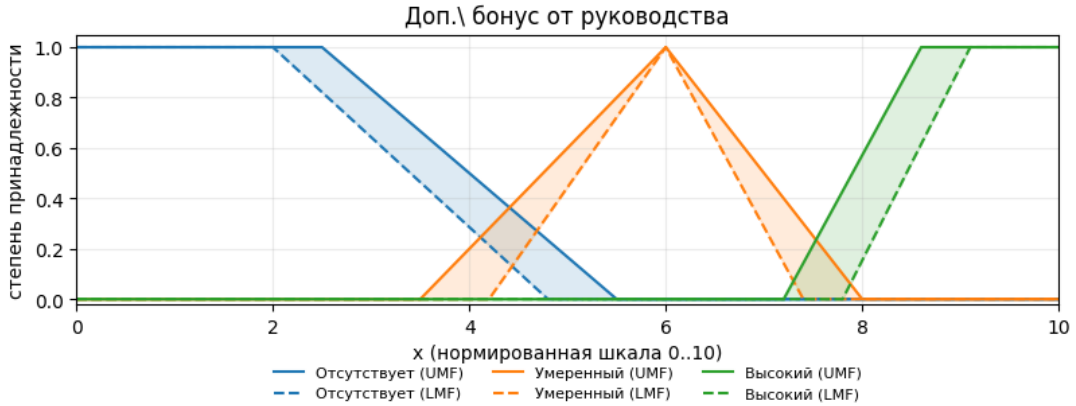
4. **Усиливающий R-цикл «выгорания».** В этом цикле усталость разработчика приводит к увеличению количества ошибок. Рост ошибок вызывает усиление контроля со стороны руководства. Повышенный контроль создаёт дополнительное напряжение для разработчика, что, в свою очередь, ещё больше увеличивает усталость. Таким образом формируется усиливающий цикл, который может приводить к состоянию выгорания и ускорять снижение производительности.

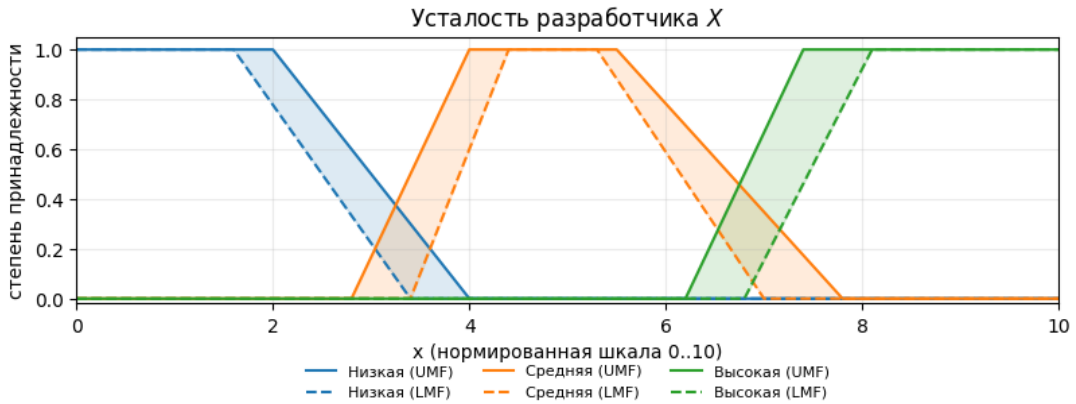
5. **Совместное влияние циклов.** При длительном снижении производительности (что соответствует рассматриваемой проблеме) доминирующим становится балансирующим цикл с R-подциклом выгорания, который усиливает негативные тенденции.

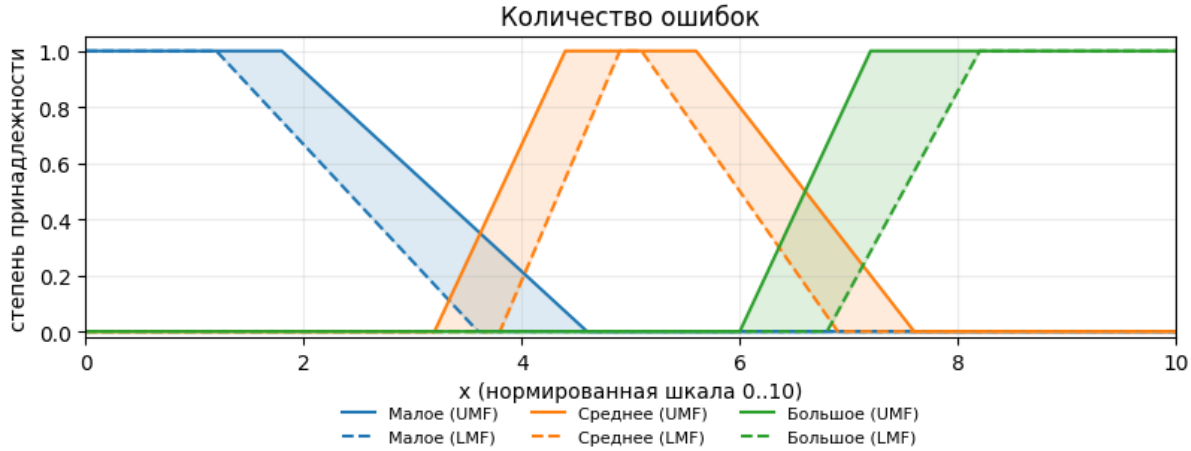
Для установления возможных зависимостей определим термы всех переменных и построим графики Behaviour over time.

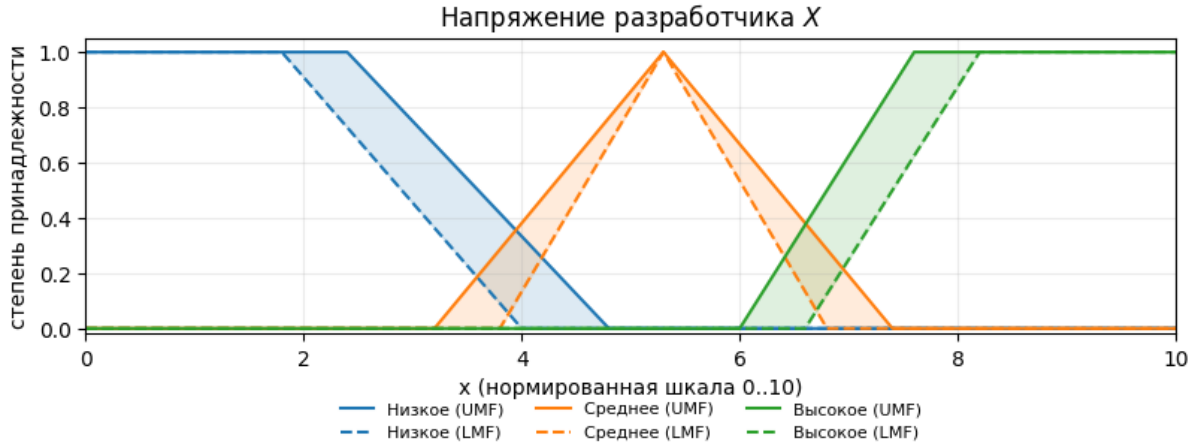

Производительность разработчика X
степень принадлежности
x (нормированная шкала 0..10)
Низкая (UMF)
Низкая (LMF)
Средняя (UMF)
Средняя (LMF)
Высокая (UMF)
Высокая (LMF)

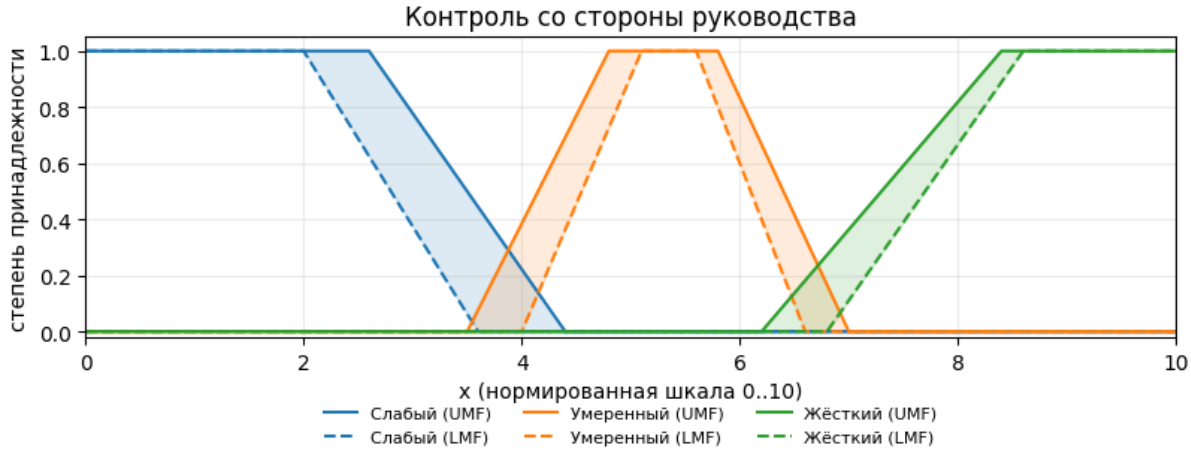

Ожидания руководства
степень принадлежности
x (нормированная шкала 0..10)
Низкие (UMF)
Низкие (LMF)
Умеренные (UMF)
Умеренные (LMF)
Высокие (UMF)
Высокие (LMF)


Доп.\ бонус от руководства
степень принадлежности
x (нормированная шкала 0..10)
Отсутствует (UMF)
Отсутствует (LMF)
Умеренный (UMF)
Умеренный (LMF)
Высокий (UMF)
Высокий (LMF)


Усталость разработчика X
степень принадлежности
x (нормированная шкала 0..10)
Низкая (UMF)
Низкая (LMF)
Средняя (UMF)
Средняя (LMF)
Высокая (UMF)
Высокая (LMF)







Для всех связей в диаграмме положим следующие задержки:

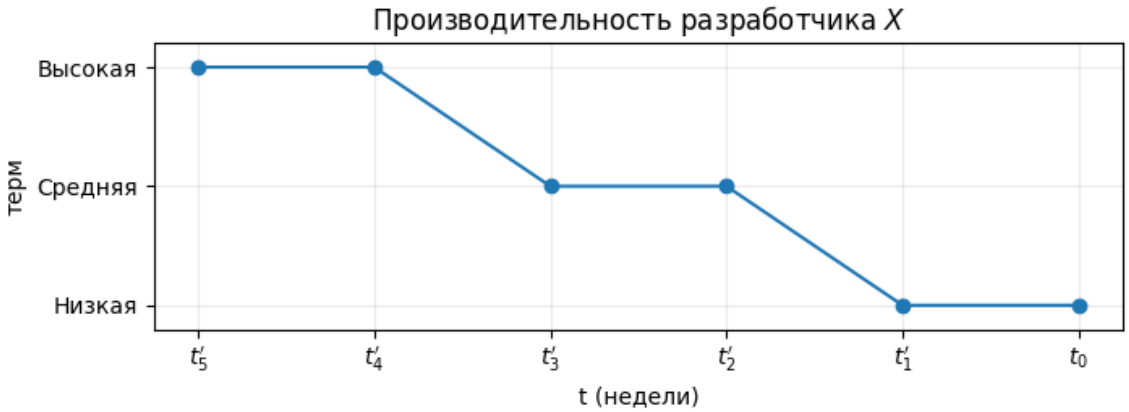
1. **Цикл мотивации (R):**
   - $P \to E$ (производительность → ожидания): задержка 1 неделя;
   - $E \to B$ (ожидания → доп. бонус): задержка 1 неделя;
   - $B \to P$ (бонус → производительность): задержка 2 недели.
2. **Цикл ограничений через ошибки (B):**

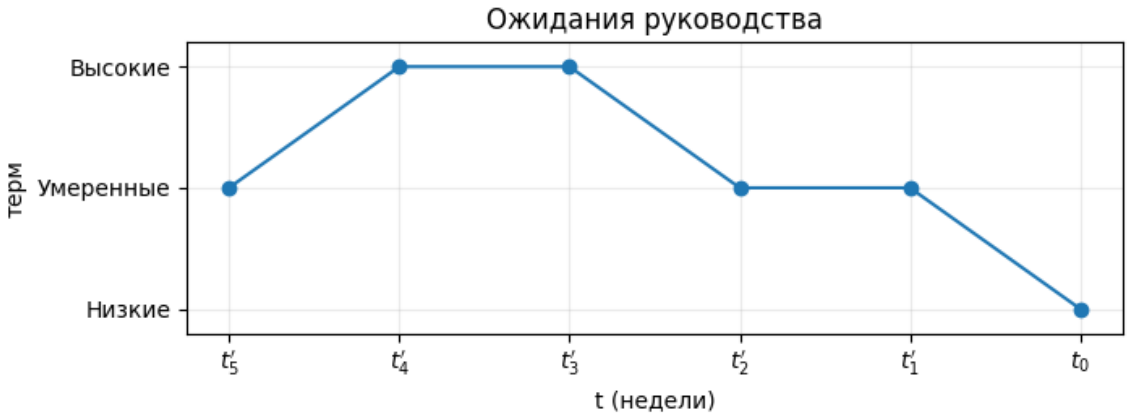
- $P \to F$ (производительность → усталость): задержка 1 неделя;
- $F \to Err$ (усталость → ошибки): задержка 1 неделя;
- $Err \to P$ (ошибки → производительность): задержка 1 неделя.

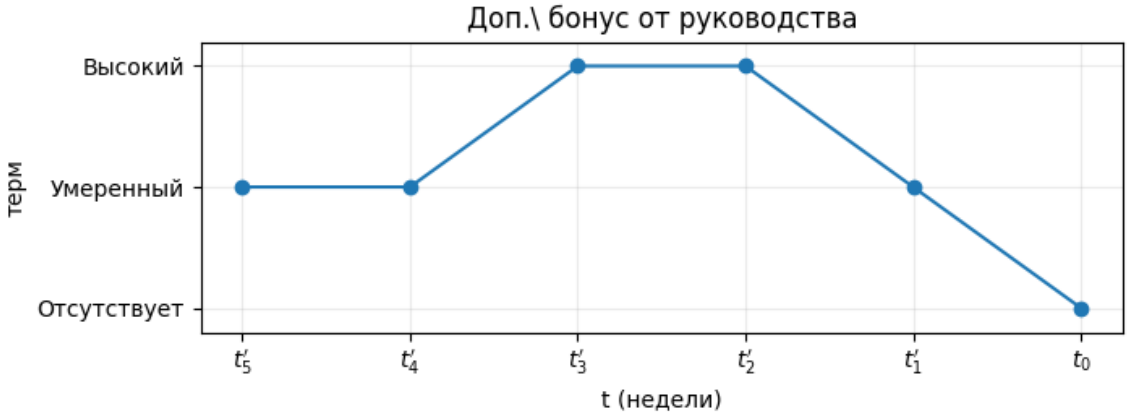
3. **Цикл выгорания (R):**

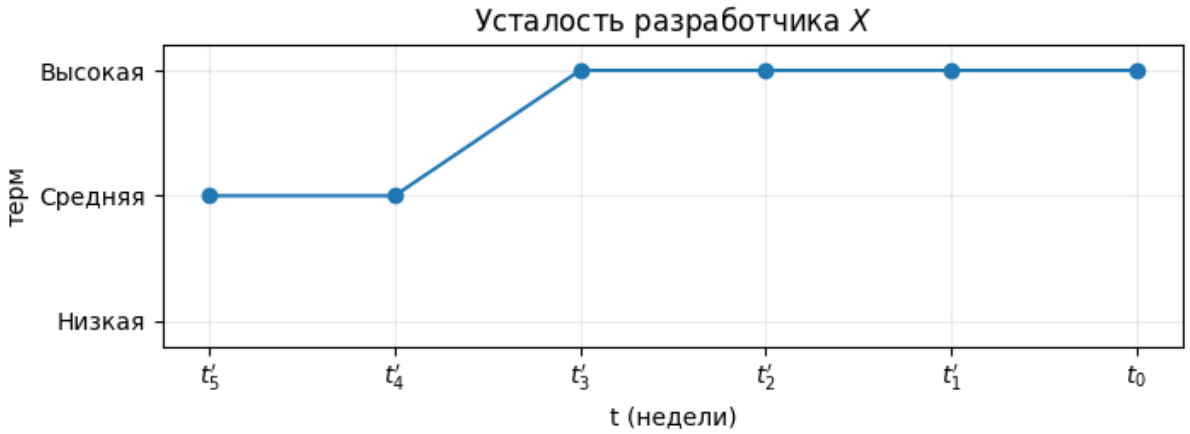
- $Err \to Ctrl$ (ошибки → контроль): задержка 1 неделя;
- $Ctrl \to Tens$ (контроль → напряжение): задержка 1 неделя;
- $Tens \to F$ (напряжение → усталость): задержка 1 неделя.

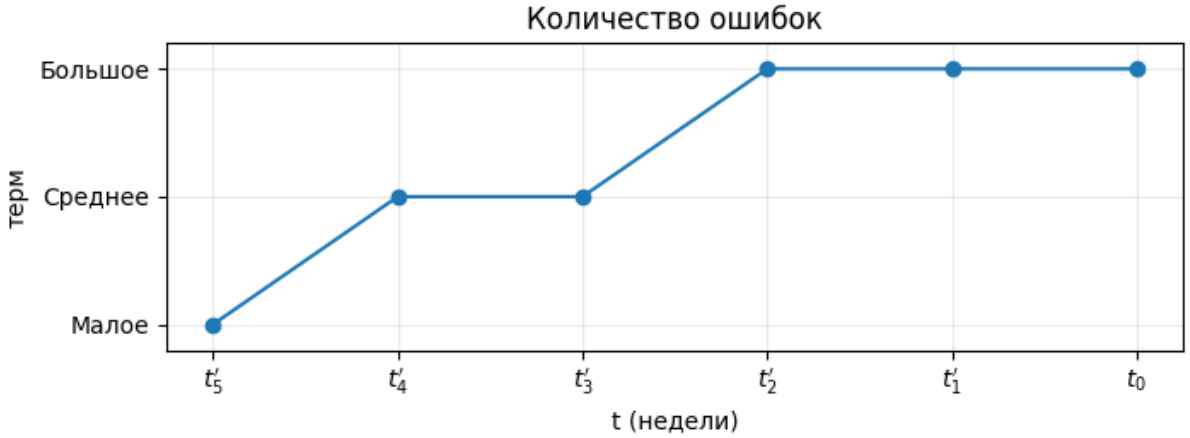
С учётом задержек построим оценочную динамику переменных в период последних 6 недель. Одно деление оси абсцисс соответствует одной неделе.

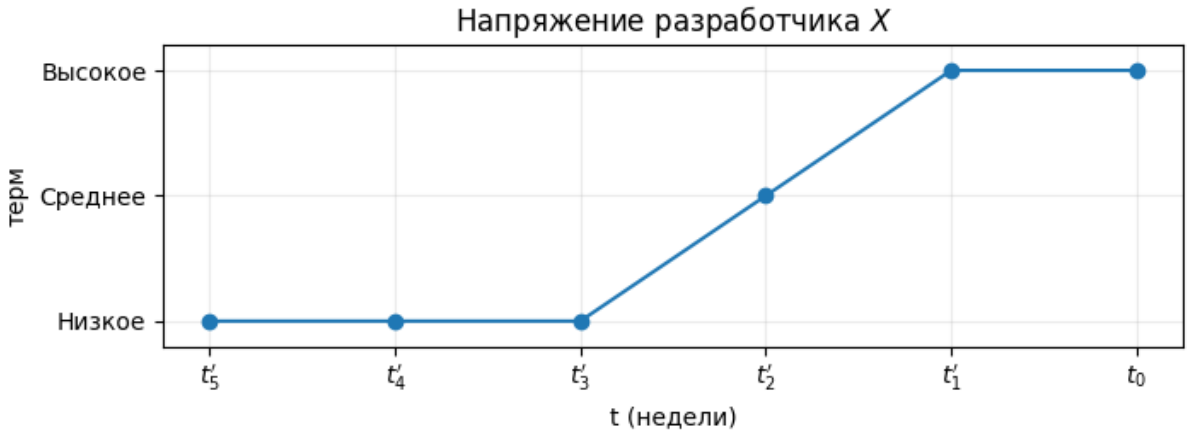


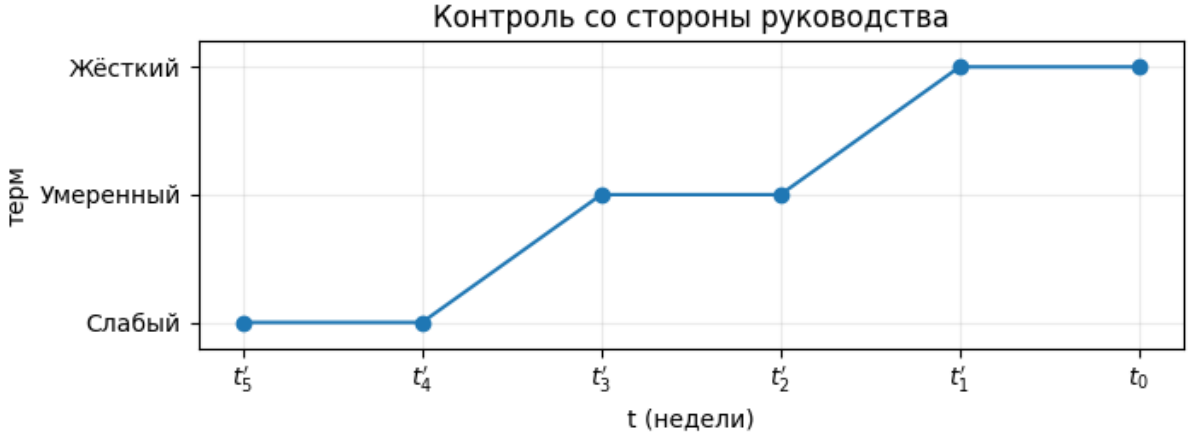





Усталость разработчика X
Высокая
Средняя
Низкая
терм
t'_5
t'_4
t'_3
t'_2
t'_1
t_0
t (недели)


Количество ошибок
Большое
Среднее
Малое
терм
t'_5
t'_4
t'_3
t'_2
t'_1
t_0
t (недели)


Напряжение разработчика X
Высокое
Среднее
Низкое
терм
t'_5
t'_4
t'_3
t'_2
t'_1
t_0
t (недели)


Контроль со стороны руководства
Жёсткий
Умеренный
Слабый
терм
t'_5
t'_4
t'_3
t'_2
t'_1
t_0
t (недели)

Проанализируем сложившуюся ситуацию

1. **Возможные причины снижения производительности.** Наблюдаемое снижение производительности разработчика X может быть связано с совокупным действием нескольких факторов. Во-первых, длительное поддержание высокого темпа работы приводит к накоплению усталости, что негативно сказывается на концентрации и качестве выполняемых задач. Во-вторых, рост усталости сопровождается увеличением количества ошибок, требующих дополнительного времени на исправление. В-третьих, на фоне ошибок руководство может усиливать контроль, что повышает психологическое напряжение разработчика и способствует формированию состояния выгорания. В результате возникает замкнутый негативный контур, поддерживающий дальнейший спад производительности.

2. **Факторы, доступные для управленческого воздействия.** Рассматривая ситуацию со стороны руководства, можно выделить три фактора, на которые можно относительно быстро воздействовать с целью повышения производительности:

   - *Ожидания руководства* — объём задач, жёсткость сроков и требования к результатам, которые могут быть скорректированы без существенных организационных издержек;
   - *Контроль со стороны руководства* — частота проверок и формат взаимодействия с разработчиком;
   - *Дополнительный бонус* — материальные стимулы, используемые для поддержки мотивации.

С учётом того, что разработчик X является ключевым участником команды, а проект должен быть завершён в сжатые сроки, пассивное ожидание естественного восстановления производительности может ещё больше усугубить проблему. Требуется целенаправленное управленческое вмешательство, направленное на разрыв негативных усиливающих циклов и создание условий для стабилизации и последующего роста производительности. Для этого рассмотрим две альтернативы:

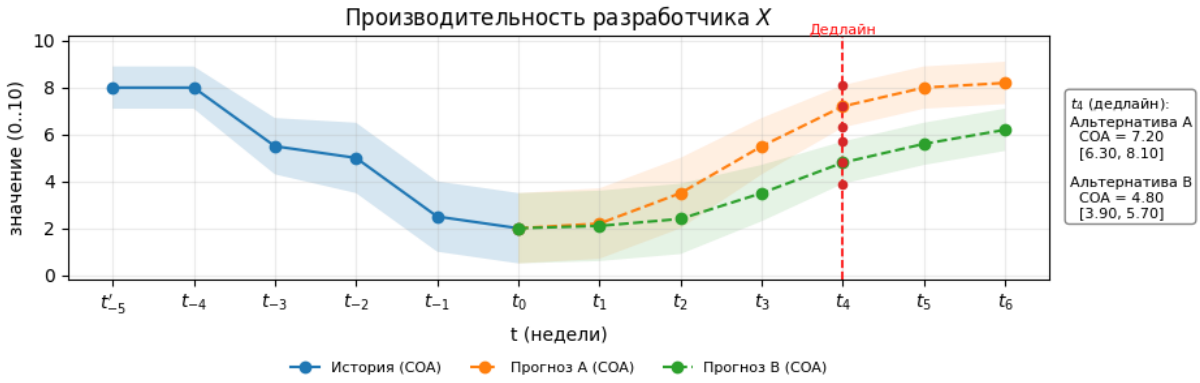
1. **Альтернатива A: снижение давления и усиление поддержки.** В рамках данной стратегии руководство целенаправленно снижает уровень ожиданий и ослабляет контроль за работой разработчика X. Параллельно вводится повышенный дополнительный бонус как сигнал поддержки и признания вклада разработчика. Снижение контроля приводит к уменьшению напряжения практически сразу, тогда как усталость и количество ошибок начинают снижаться с

задержкой в одну–две недели. Эффект от бонуса проявляется не мгновенно, однако спустя несколько недель наблюдается рост производительности, который позволяет вывести проект на приемлемую траекторию выполнения.

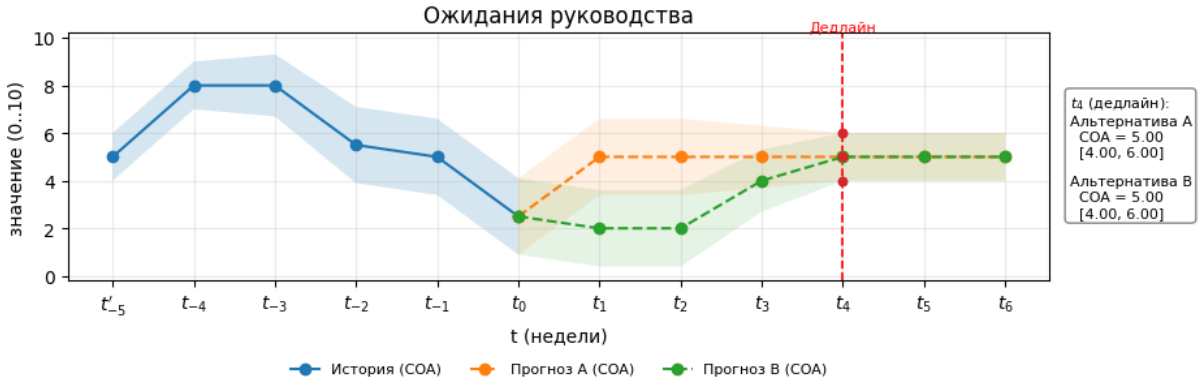
2. **Альтернатива B: мягкая стабилизация с сохранением управляемости.** В данной стратегии руководство временно снижает ожидания до низкого уровня, позволяя разработчику восстановить рабочий ритм, при этом контроль не отменяется полностью, а переводится в умеренный режим. Дополнительный бонус устанавливается на среднем уровне и носит поддерживающий характер. Такое вмешательство приводит к более медленному, но устойчивому снижению напряжения и усталости. Количество ошибок уменьшается постепенно, а производительность сначала стабилизируется, после чего начинает расти, не вызывая повторного усиления негативных эффектов, связанных с перегрузкой.

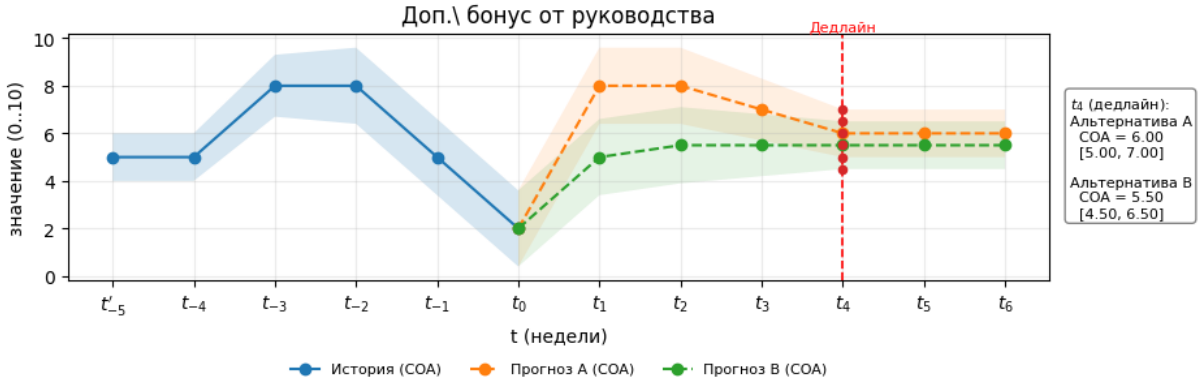
Абстрагируясь от воздействия внешних (не рассмотренных) факторов, смоделируем динамику изменения переменных для рассмотренных альтернатив улучшающего вмешательства. Будем изменять количественное значение переменных и смотреть, как от этого меняются термы.

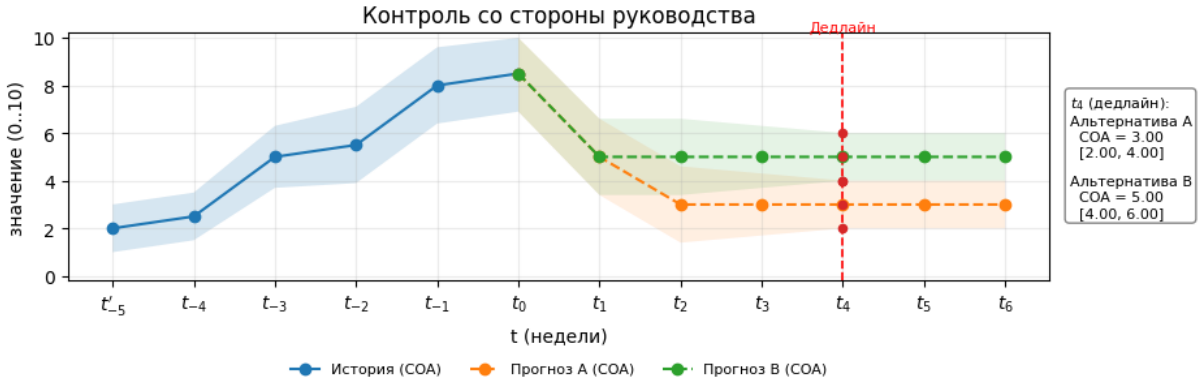
Главная переменная:



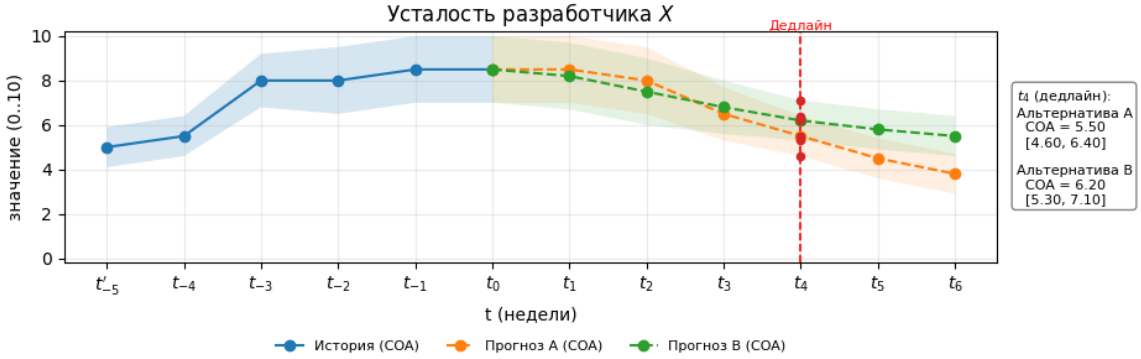
Контролируемые факторы (ожидание руководства, доп. бонус, контроль со стороны руководства):

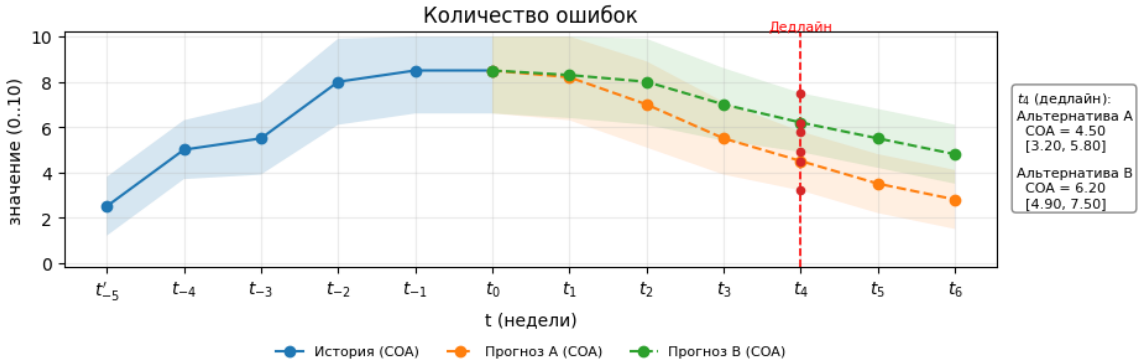


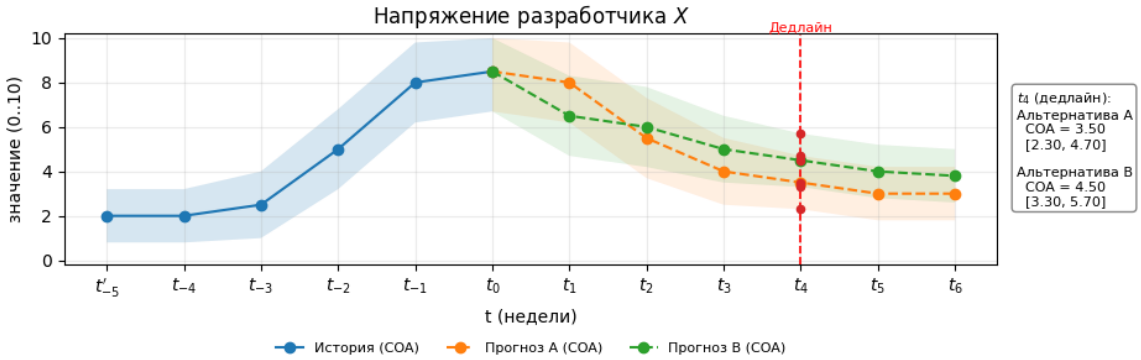




Не контролируемые напрямую факторы (усталость, количество ошибок, напряжение разработчика):







Проведём анализ полученных результатов

1. **Главная переменная — производительность разработчика *X*.** К моменту дедлайна t_4 при реализации **альтернативы A** ожидаемая производительность (COA) достигает высокого уровня и имеет сравнительно узкий интервал неопределённости, что указывает на высокую вероятность успешной стабилизации ситуации. В случае **альтернативы B** производительность к дедлайну повышается лишь до среднего уровня, а интервал неопределённости остаётся шире, что делает исход менее надёжным с точки зрения сроков сдачи проекта.

2. **Контролируемые управленческие факторы (ожидания, бонус, контроль).** Динамика ожиданий руководства, дополнительного бонуса и уровня контроля демонстрирует их ключевую роль в немедленном влиянии на производительность разработчика. Исторически рост ожиданий и усиление контроля сопровождаются увеличением напряжения и усталости. В **альтернативе A** руководство снижает уровень контроля и стабилизирует ожидания, одновременно значительно повышая дополнительный бонус. Такое сочетание факторов позволяет быстро снизить давление на разработчика и создать условия для ускоренного восстановления производительности уже через две недели. В **альтернативе B** управленческое воздействие носит более мягкий характер: ожидания временно снижаются, контроль удерживается на умеренном уровне, а бонус используется как поддерживающий инструмент. Это обеспечивает большую устойчивость системы, однако приводит к более медленному улучшению ключевых показателей.

3. **Неконтролируемые напрямую факторы: усталость, ошибки и напряжение.** Для всех трёх переменных характерна инерционная динамика. Даже при управленческом вмешательстве их снижение происходит с задержкой. В альтернативе A эти показатели начинают снижаться раньше и быстрее, что приводит к заметному уменьшению количества ошибок уже к дедлайну. В альтернативе B снижение происходит более плавно, и к моменту дедлайна сохраняется повышенный уровень усталости и ошибок.

4. **Сравнительный вывод по альтернативам.** Альтернатива A ориентирована на быстрый эффект и разрыв негативных усиливающих циклов, связанных с выгоранием и чрезмерным контролем. Она демонстрирует более высокий уровень производительности и меньшую неопределённость в точке дедлайна t_4, что делает её предпочтительной при жёстких сроках сдачи проекта. Альтернатива B использует меньшие ресурсы и обеспечивает более мягкую и устойчивую стабилизацию системы, однако её эффект проявляется медленнее и может оказаться недостаточным в условиях ограниченного времени.

5. **Итоговый вывод.** Анализ графиков Behaviour over time показывает, что при наличии жёсткого дедлайна и ключевой роли разработчика X более применимой является альтернатива A. Альтернатива B может рассматриваться как запасной вариант, уместный в ситуациях, когда руководство не готово предоставить разработчику необходимые для альтернативы A ресурсы.

Задание №3

В предыдущей задаче была рассмотрена динамика снижения производительности разработчика X с использованием причинно-следственной диаграммы и графиков Behaviour over time. На основе анализа системы были предложены две альтернативы управленческого вмешательства, направленные на стабилизацию и последующее повышение производительности разработчика в условиях приближающегося дедлайна проекта.

В данной задаче требуется рассмотреть проблему выбора наиболее предпочтительной альтернативы. Выбор должен осуществляться на основе многокритериальной (≥ 5) оценки, учитывающей как результат проекта, так и устойчивость системы в целом. Каждому из критериев необходимо присвоить вес, пропорциональный его значимости в финальной оценке. Как оценки критериев, так и веса необходимо представить в виде нечётких значений (интервалов). Для агрегирования критериев и получения итогового результата необходимо применить метод Рональда Яджера. Полученные результаты различных альтернатив следует проанализировать и сравнить между собой для выбора наиболее предпочтительного решения.

Решение

В качестве критериев оценки альтернатив могут быть использованы следующие факторы с учётом их важности для решения текущей имеющейся проблемы:

- **Качество продукта** — характеризует практическую ценность получаемого решения и степень соответствия результата требованиям заказчика. Данный критерий является одним из ключевых, так как напрямую влияет на успешность проекта. Вес критерия задаётся нечётким интервалом

$$p_1 \in [0.30;\ 0.40].$$

- **Затраченные дополнительные ресурсы** — описывает объём материальных, временных и управленческих ресурсов, необходимых со стороны руководства для внедрения альтернативы. Несмотря на важность данного критерия, в условиях жёсткого дедлайна он рассматривается как вторичный по отношению к результату. Вес критерия задаётся интервалом

$$p_2 \in [0.10;\ 0.20].$$

- **Надёжность выполнения в срок** — отражает вероятность завершения проекта в установленные временные рамки. В условиях приближающегося дедлайна данный критерий приобретает критическое значение. Вес критерия задаётся

интервалом

$$p_3 \in [0.25;\ 0.35].$$

- **Риски повторного снижения производительности** — характеризует вероятность возврата системы в неблагоприятное состояние после реализации управленческого вмешательства. Данный критерий важен с точки зрения устойчивости решения, однако его влияние проявляется в более долгосрочной перспективе. Вес критерия задаётся интервалом

$$p_4 \in [0.05;\ 0.15].$$

- **Адаптивность остальной команды к изменениям** — описывает способность команды быстро и эффективно подстроиться под изменения, необходимые для внедрения альтернативы, без снижения общей эффективности работы. Вес критерия задаётся интервалом

$$p_5 \in [0.15;\ 0.25].$$

Рассмотрим три альтернативы управленческого воздействия (альтернативы A и B соответствуют альтернативам, применённым к CLD):

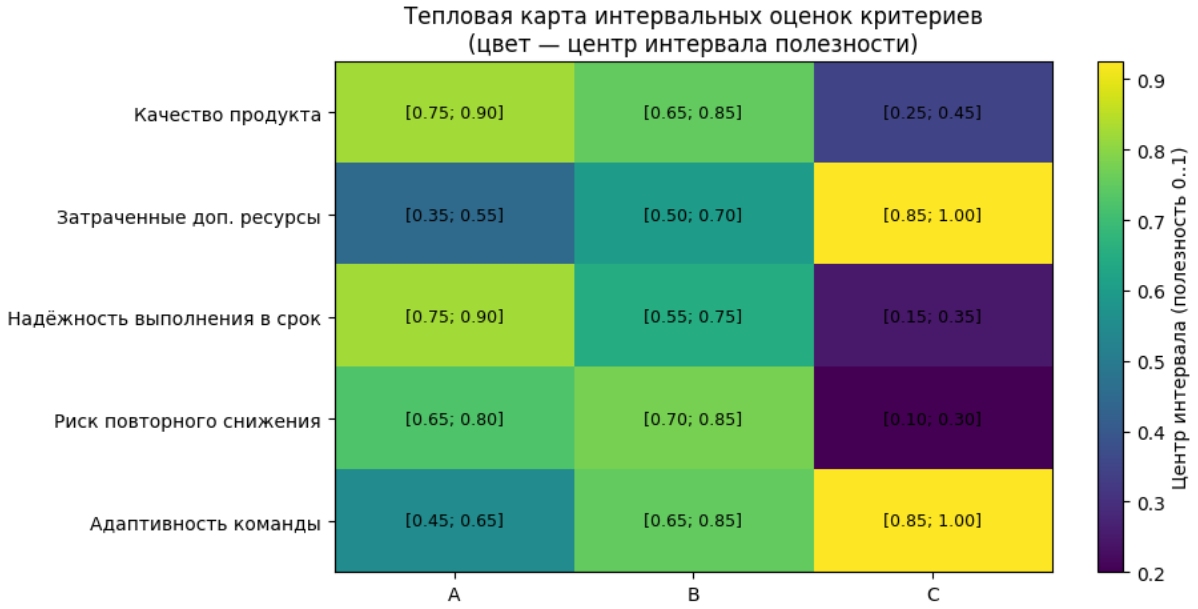
- **Альтернатива A (активное вмешательство).** Данная альтернатива соответствует сценарию быстрого управленческого воздействия. Руководство снижает давление на разработчика X за счёт стабилизации ожиданий, ослабляет контроль до умеренного уровня и вводит дополнительную поддержку (в том числе материальную). Целью альтернативы является максимально быстрое разрыв негативного усиливающего цикла «усталость – ошибки – снижение производительности» и восстановление работоспособности разработчика к моменту дедлайна.

- **Альтернатива B (мягкая стабилизация).** В рамках данной альтернативы управленческое вмешательство носит более осторожный характер. Ожидания руководства временно снижаются, контроль сохраняется на умеренном уровне, а дополнительный бонус используется как поддерживающий, а не стимулирующий инструмент. Основной акцент делается на постепенное восстановление устойчивости системы и снижение рисков, однако ожидаемый эффект проявляется медленнее по сравнению с альтернативой A.

- **Альтернатива C (отсутствие вмешательства).** Данная альтернатива предполагает игнорирование наблюдаемого снижения производительности. Управленческие параметры не изменяются, и система продолжает развиваться по

текущей траектории. В этом случае ожидается сохранение высокой неопределённости, рост рисков срыва сроков и ухудшение качества конечного продукта.

Представим возможные оценки значений критериев для альтернатив в виде тепловой карты. Заметим, что затраченные дополнительные ресурсы и риск повторного снижения производительности выражают обратные зависимости.



Для выбора наиболее предпочтительной альтернативы используем метод Рональда Яджера. Данный подход при небольших модификациях позволяет учитывать неопределённость как в оценках критериев, так и в их весах, что особенно важно в условиях неполной информации.

Итак, пусть веса критериев заданы интервалами

$$p_k \in [\underline{p_k}, \overline{p_k}], \quad k = 1, \ldots, 5,$$

а оценки альтернатив по критериям заданы в виде интервальных значений

$$C_k(a_i) \in [\underline{C_{k,i}}, \overline{C_{k,i}}], \quad a_i \in \{A, B, C\}.$$

В классическом методе Яджера используется следующая decision measure:

$$M(C_k(a_i), p_k) = \overline{p_k} \vee C_k(a_i),$$

где $\overline{p_k} = 1 - p_k$, а операция $\vee$ интерпретируется как максимум.

С учётом интервальной природы весов получаем:

$$\overline{p_k} \in [1 - \overline{p_k},\ 1 - \underline{p_k}].$$

Тогда интервальная оценка decision measure для k-го критерия и альтернативы a_i вычисляется как

$$\underline{M_k(a_i)} = \max(1 - \overline{p_k},\, \underline{C_{k,i}}), \qquad \overline{M_k(a_i)} = \max(1 - \underline{p_k},\, \overline{C_{k,i}}).$$

Итоговая оценка альтернативы определяется пересечением по всем критериям:

$$D(a_i) = \bigcap_{k=1}^{5} M_k(a_i),$$

что в интервальном виде даёт

$$\underline{D(a_i)} = \min_k \underline{M_k(a_i)}, \qquad \overline{D(a_i)} = \min_k \overline{M_k(a_i)}.$$

Для удобства сравнения альтернатив вычислим центры интервалов:

$$D_{\mathrm{COA}}(a_i) = \frac{\underline{D(a_i)} + \overline{D(a_i)}}{2}.$$

Ниже представлены результаты вычислений

Альтернатива A.

$$D(A) \in [0.75;\, 0.85], \qquad D_{\mathrm{COA}}(A) = 0.80.$$

Альтернатива B.

$$D(B) \in [0.65;\, 0.75], \qquad D_{\mathrm{COA}}(B) = 0.70.$$

Альтернатива C (отсутствие вмешательства).

$$D(C) \in [0.60;\, 0.70], \qquad D_{\mathrm{COA}}(C) = 0.65.$$

Полученные результаты позволяют упорядочить альтернативы по степени предпочтительности:

$$D(A) > D(B) > D(C).$$

Альтернатива A демонстрирует наибольшее значение итоговой оценки и наименьшую неопределённость, что делает её наиболее предпочтительной в условиях жёсткого дедлайна и высокой критичности результата. Альтернатива B обеспечивает более устойчивое, но менее быстрое улучшение ситуации и может рассматриваться как компромиссный вариант. Альтернатива C, несмотря на минимальные затраты ресурсов, характеризуется наименьшей итоговой полезностью и наибольшими рис-

ками, вследствие чего является наименее предпочтительной.

Заключение.

Заметим, что полученные результаты не только подтверждают корректность выводов, сделанных в предыдущей задаче, но и хорошо согласуются с интуитивными ожиданиями. Действительно, активное вмешательство со стороны руководства (альтернатива A) приводит к наиболее быстрому восстановлению работоспособности разработчика X, что, в свою очередь, существенно повышает вероятность успешной сдачи проекта заказчику в установленные сроки.

Полное игнорирование ситуации (альтернатива C) обладает единственным преимуществом — отсутствием дополнительных управленческих и ресурсных затрат, однако сопровождается высокими рисками снижения качества продукта и срыва дедлайна, что делает данную альтернативу наименее предпочтительной.

Сбалансированная стабилизация (альтернатива B) может рассматриваться как компромисс между двумя описанными крайностями, поскольку она снижает давление на разработчика и ограничивает затраты ресурсов. Тем не менее, в условиях приближающегося дедлайна приоритетом является своевременная поставка качественного продукта, а по совокупности оцениваемых рисков данная альтернатива уступает активному вмешательству, что и обусловливает её второе место в итоговом ранжировании.