Визуализация архитектуры приложений

Дима Zero Bias (Aviasales)

Обо мне

Создатель проекта effector  — стейт менеджер для фронтенда

Пользователи:

Avito, МТС, docsvision, okoo, aviasales, HEALTH SAMURAI, Raffeisen BANK, VK

space307, RED MAD ROBOT, travelpayouts affiliate network, JUNTO MOBILE SOLUTIONS, Yandex, unicef, JOOM, СБЕР

Последние пять лет посвятил разработке нового подхода к проектированию и визуализации архитектуры приложений

Компании добавляют себя в список здесь: github.com/effector/effector/issues/278

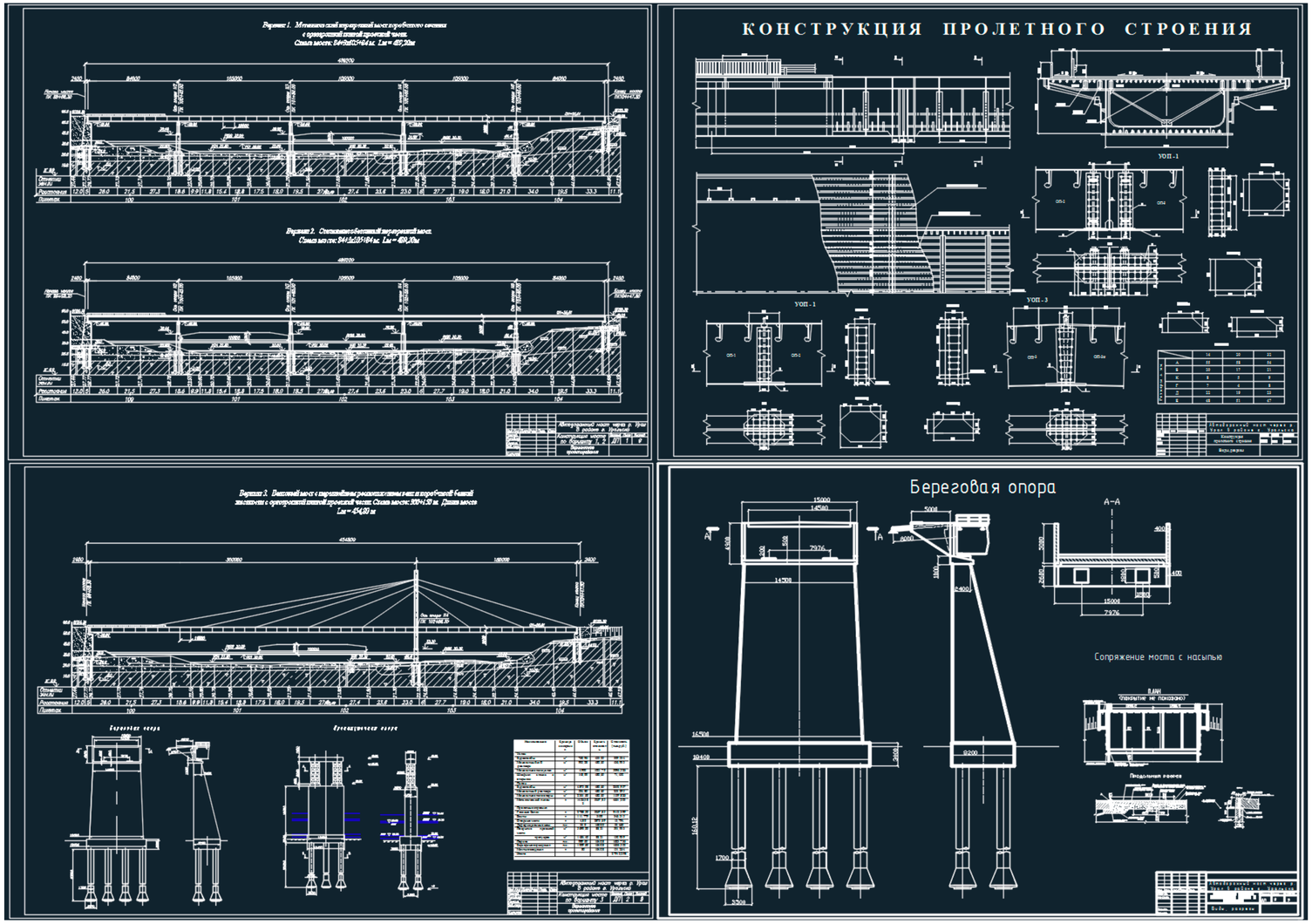
Моя цель

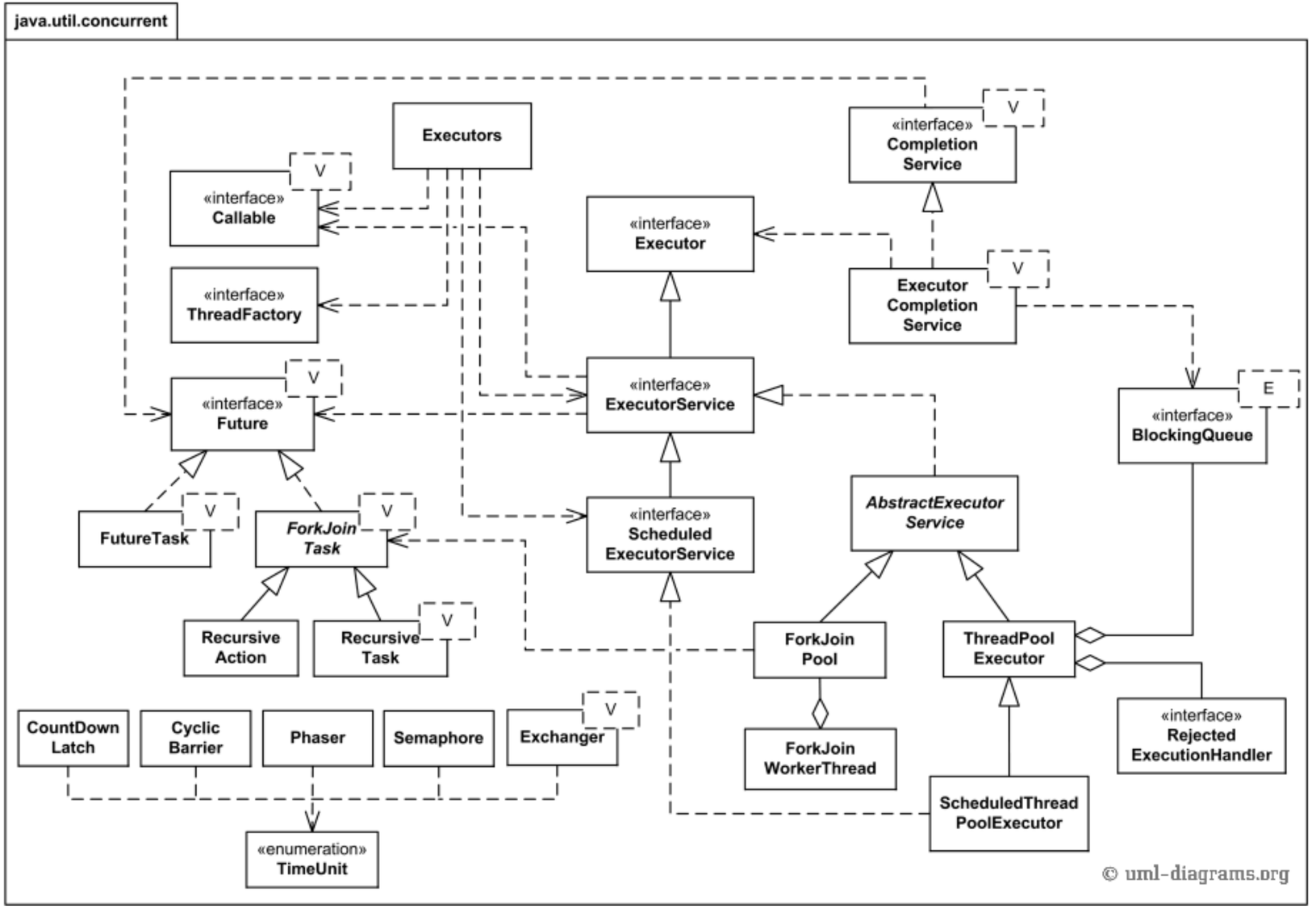
Упрощение разработки больших приложений с множеством команд

Я уверен, что инструмент, дающий возможность качественно проектировать и визуализировать логику приложений открывает для разработчиков новые возможности

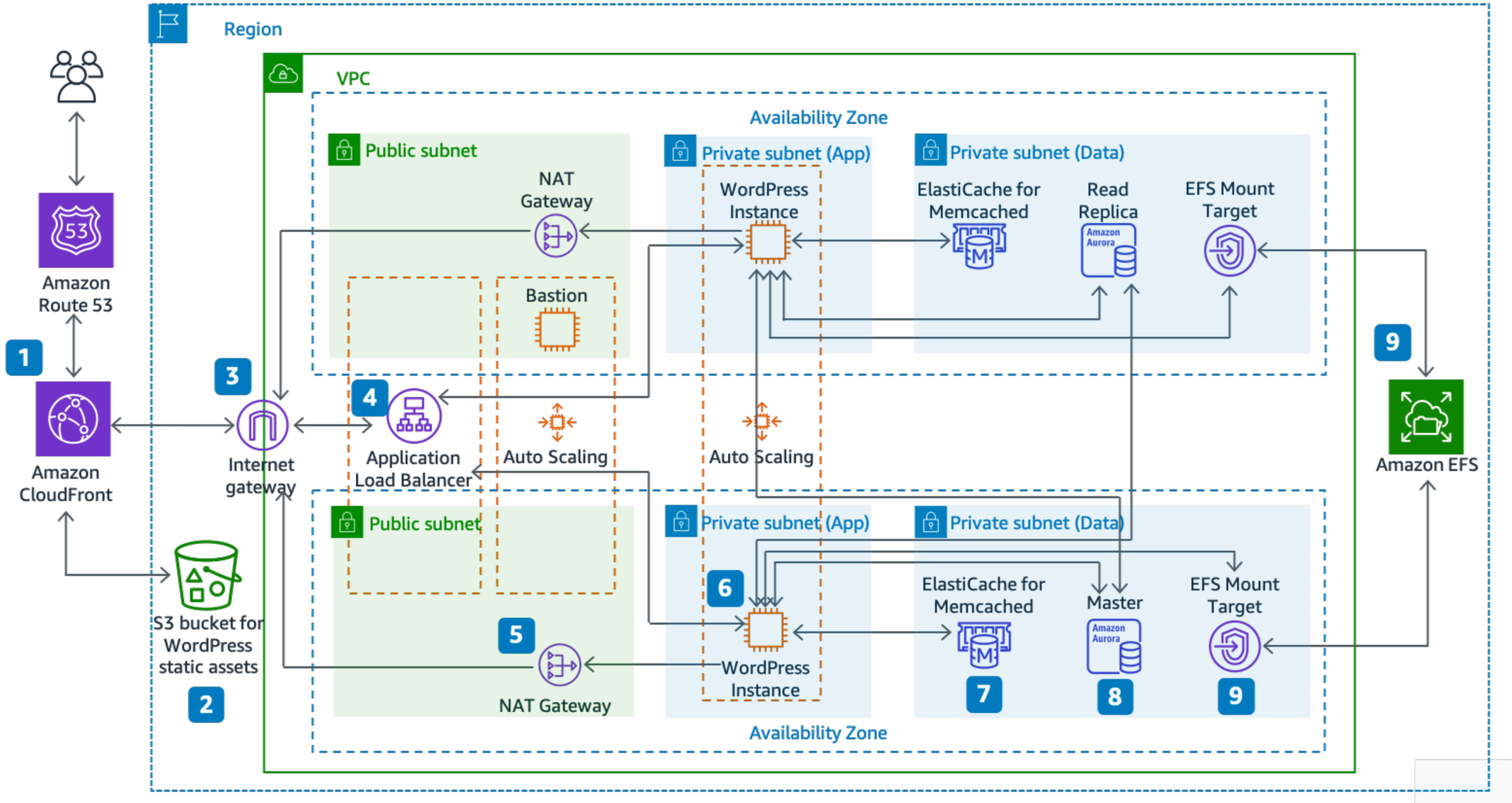
Поэтому я написал effector и теперь строю визуализацию бизнес-логики для его пользователей

Чтобы построить мост нужен чертеж

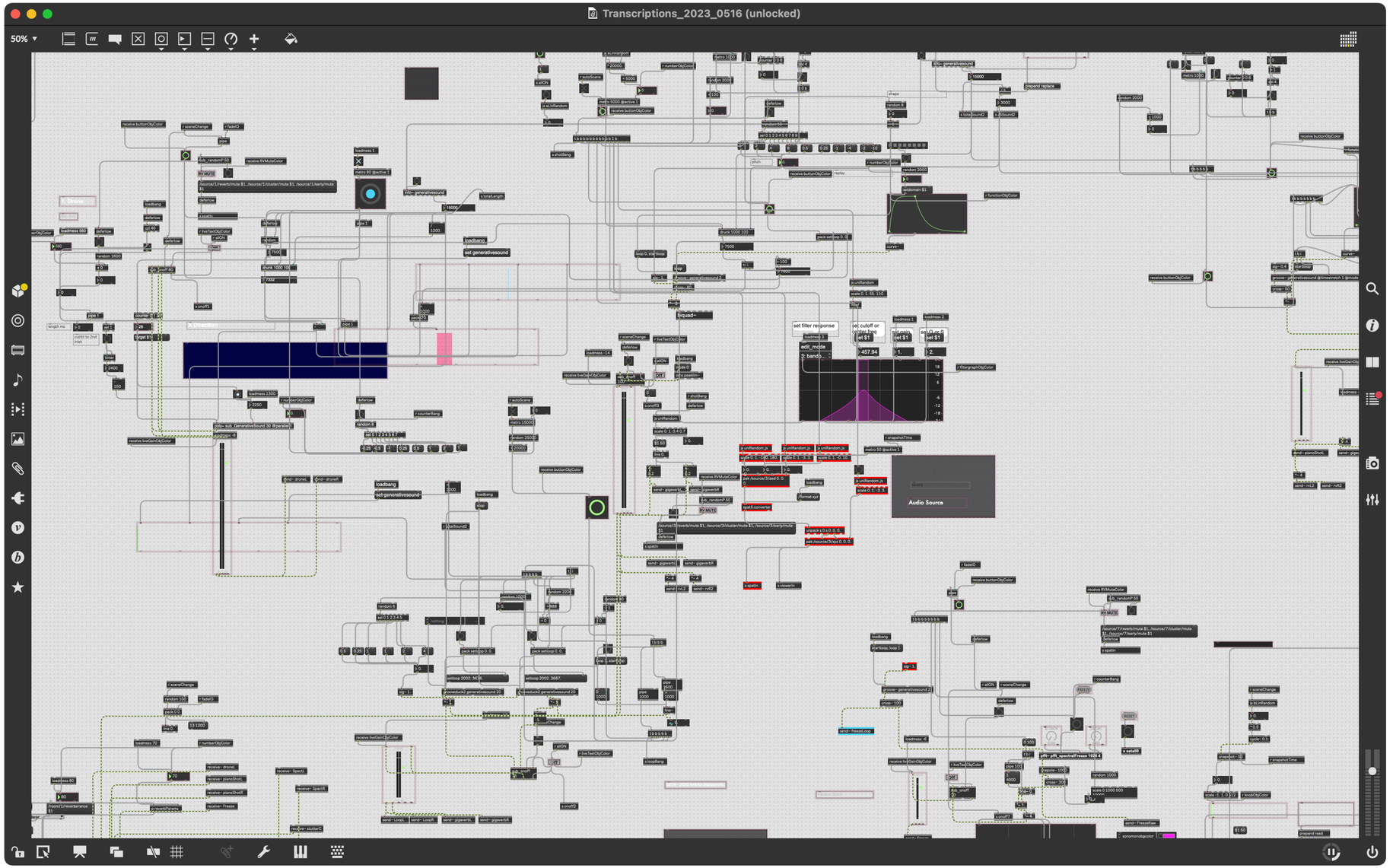




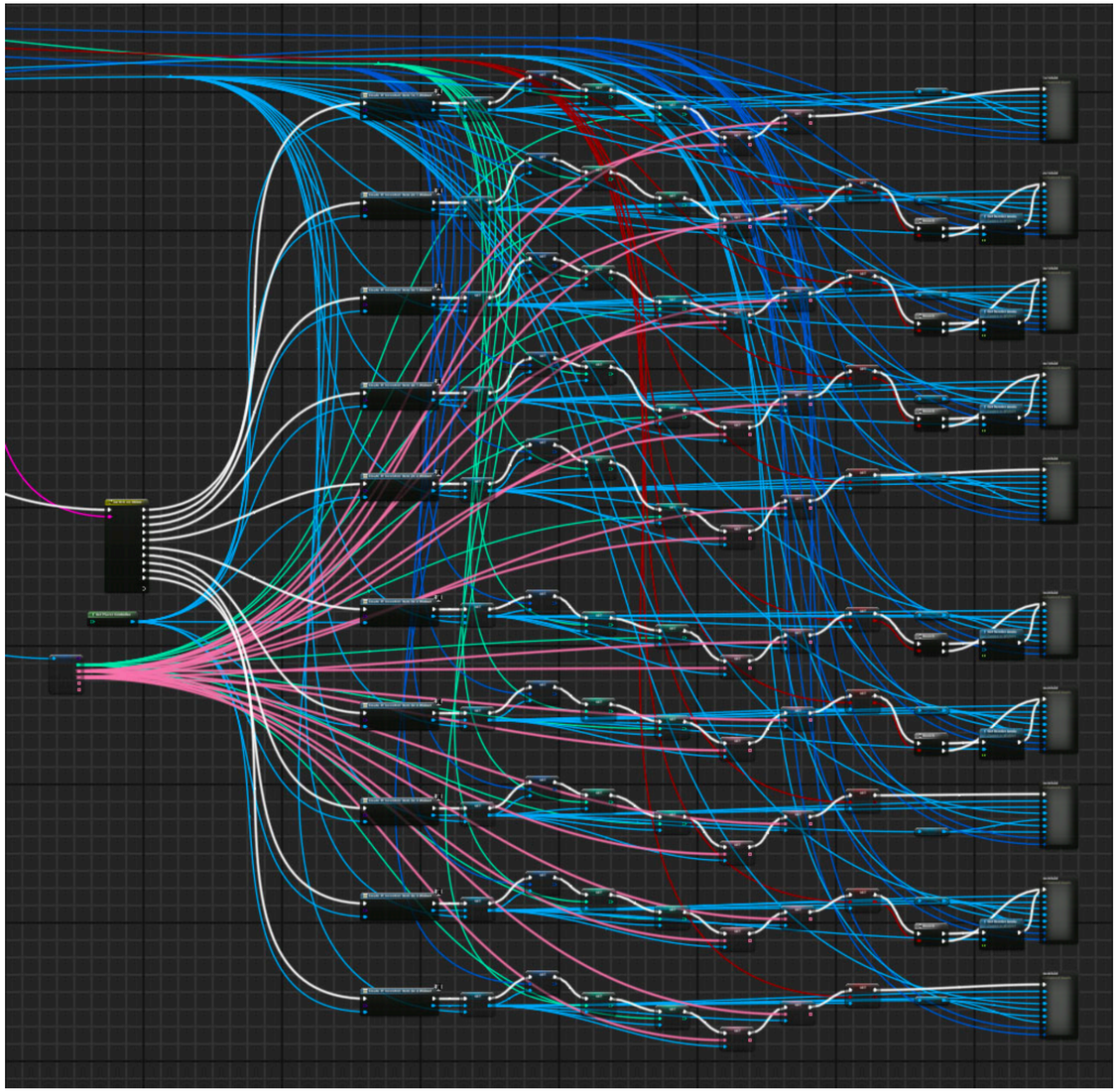
UML diagrams



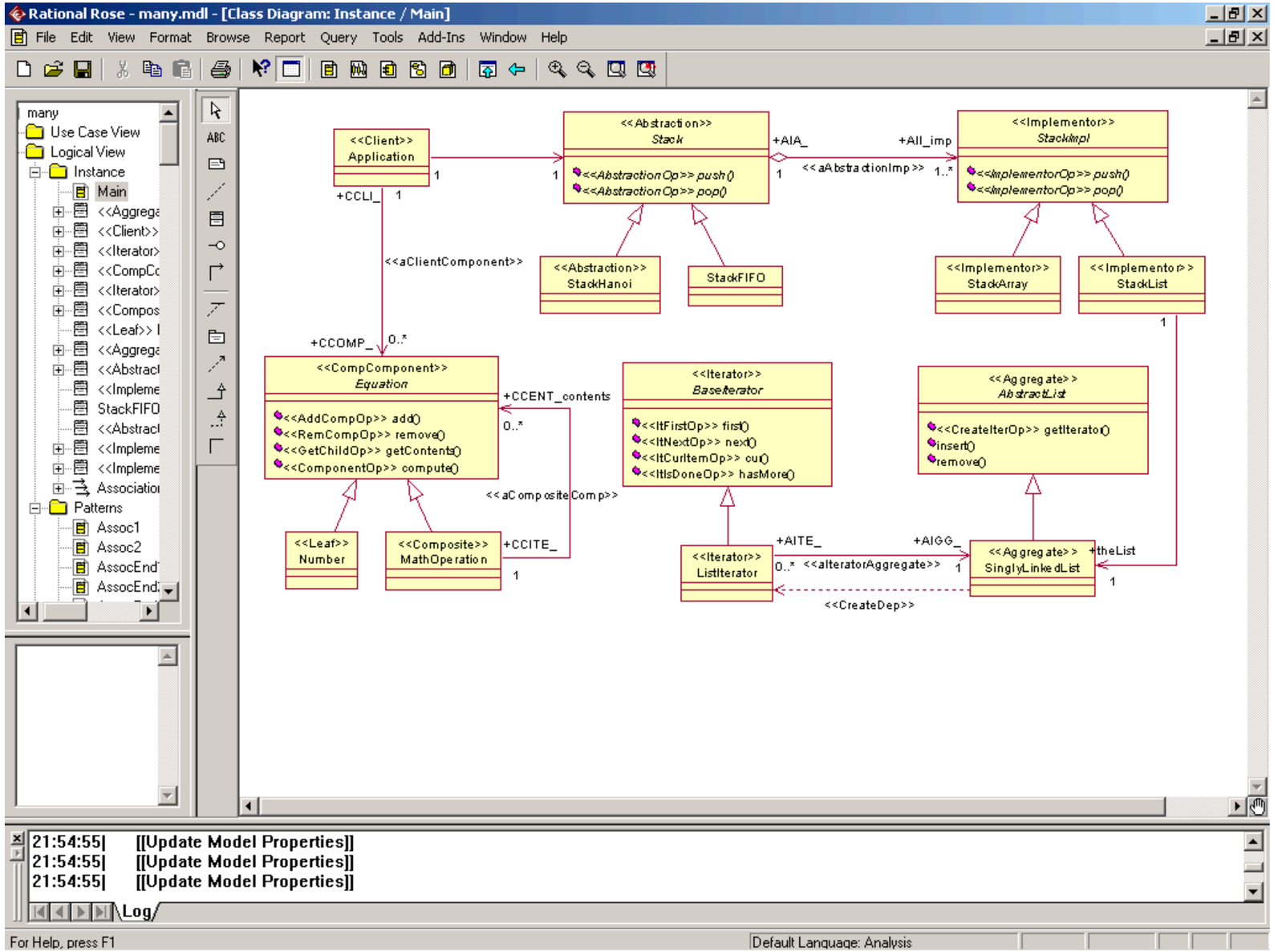
AWS best practices for Wordpress



MAX/MSP (программа для написания музыки)



Blender nodes



IBM Rational Rose

Zero Code это не решение

Первая проблема: что визуализировать?

Стандартная схема импортов не отражает поток данных в приложении

Функции и классы нельзя сопоставить с бизнес-сценариями напрямую

Чтобы визуализировать бизнес-логику, нужен инструмент для её описания

Описание должно быть декларативным, динамические вызовы функций мешают анализу

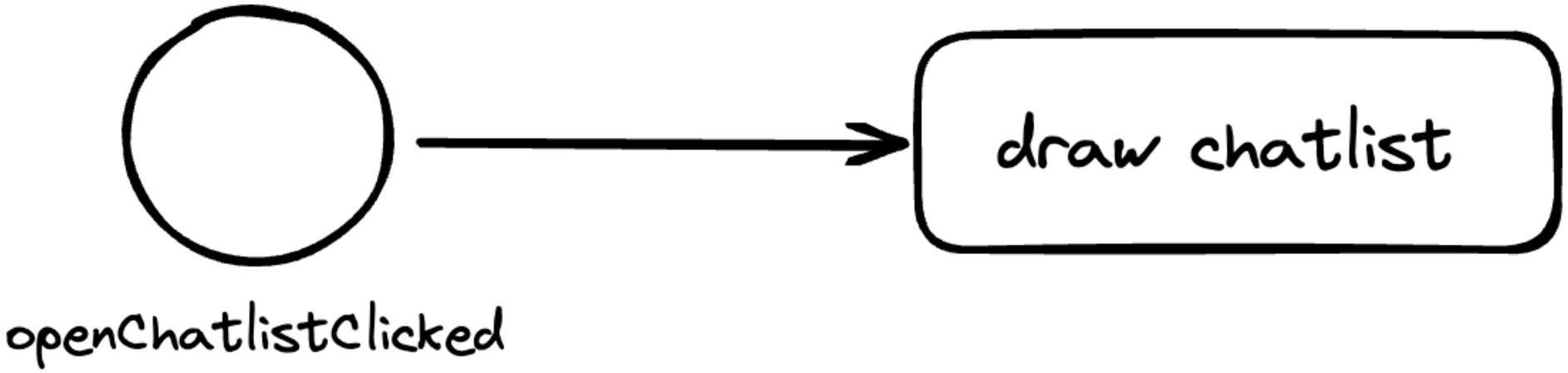
Effector

Инструмент для работы с бизнес логикой

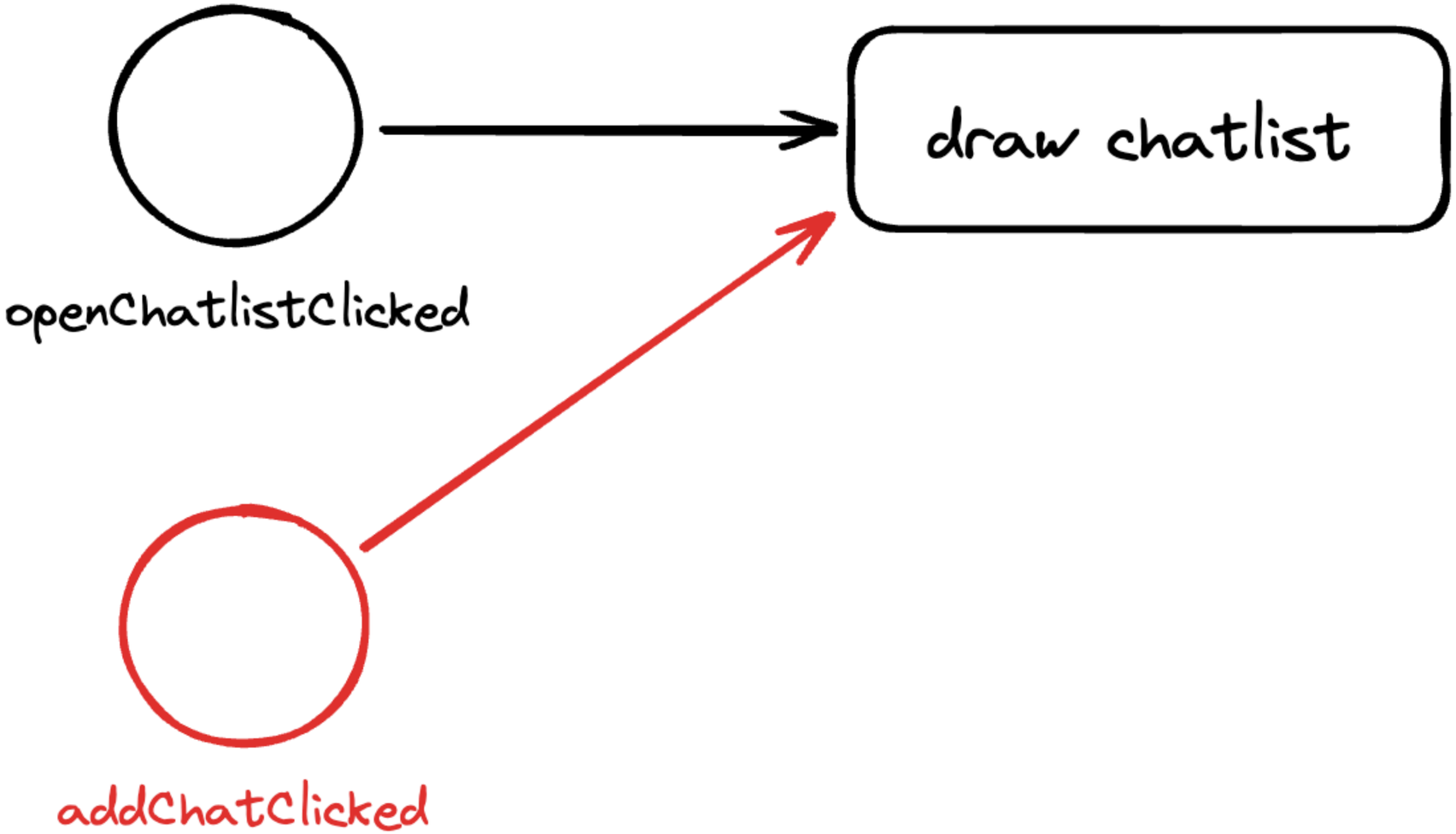
Вводит концепции важные для проектирования приложений
и визуализации

Пример: проектирование мессенджера

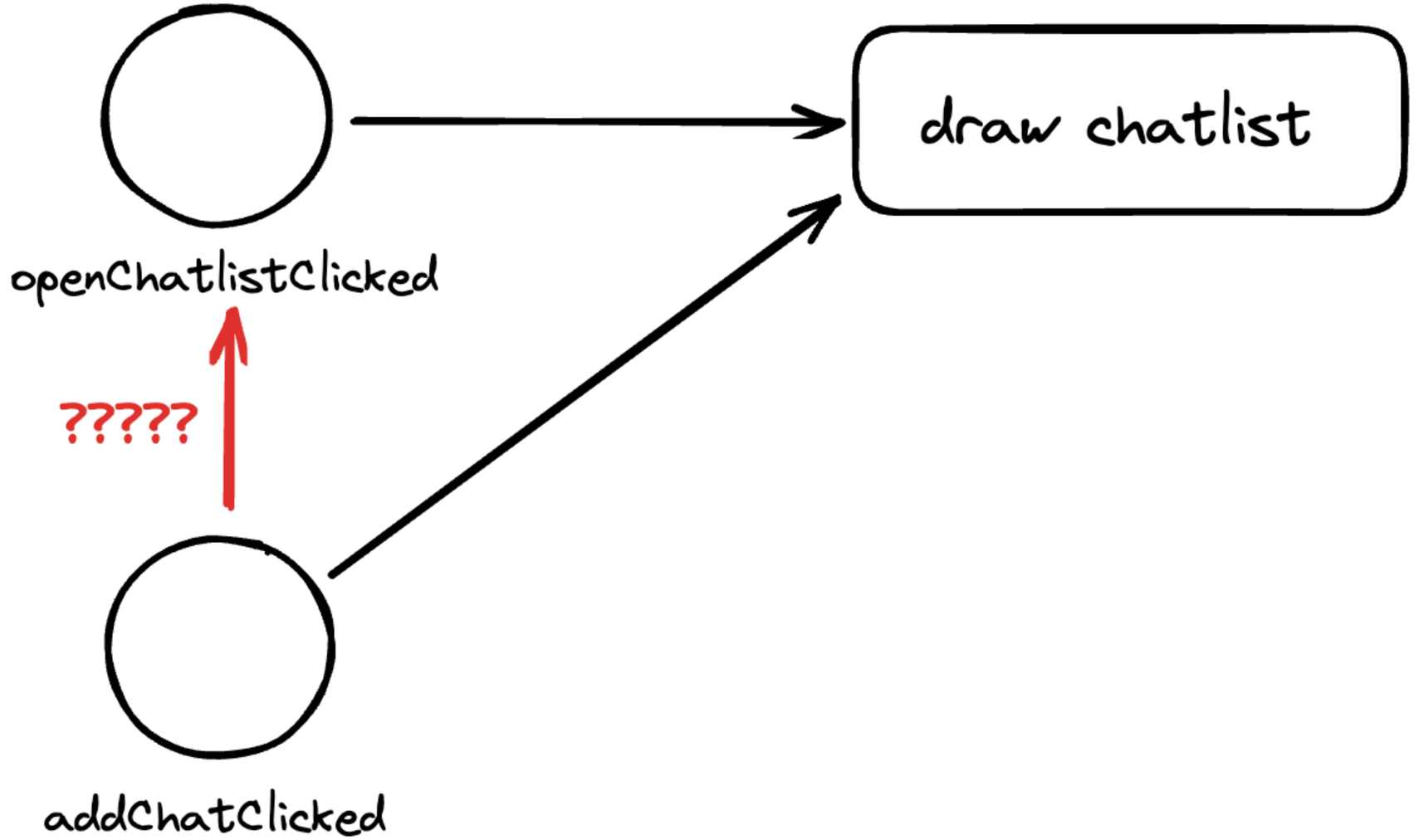
Пример: проектирование мессенджера



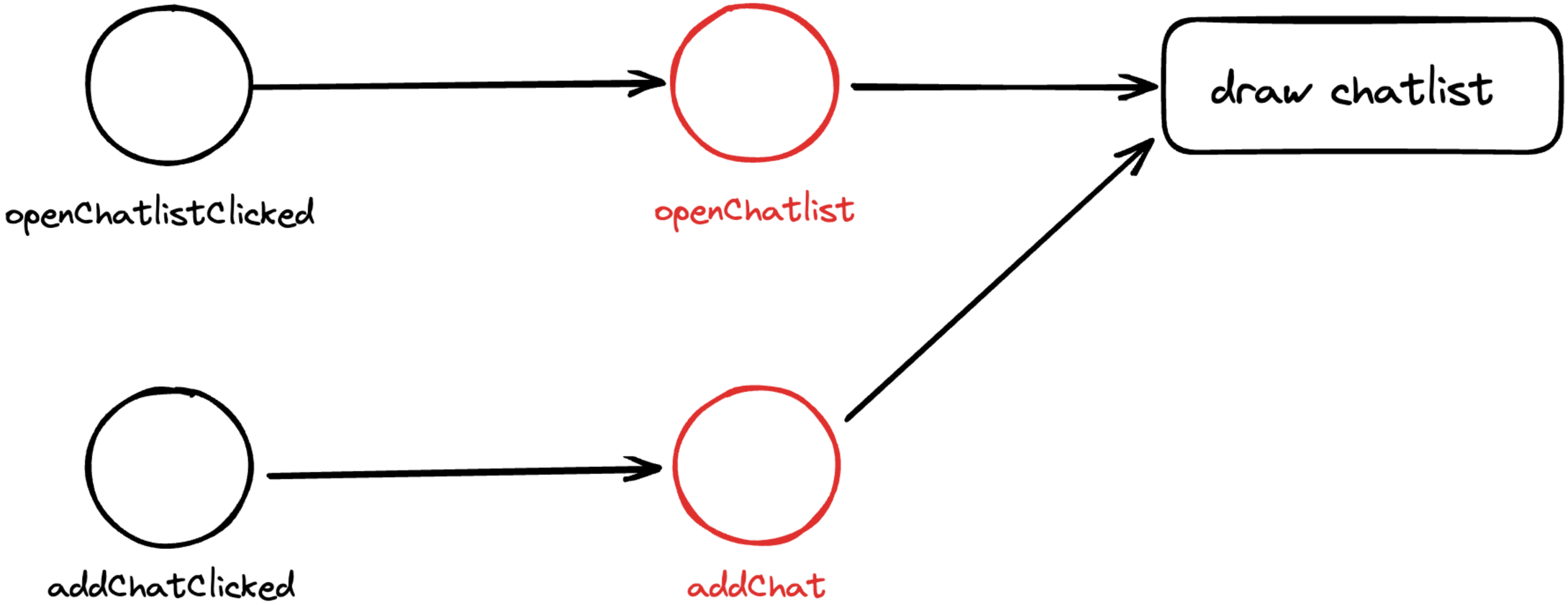
Пример: проектирование мессенджера



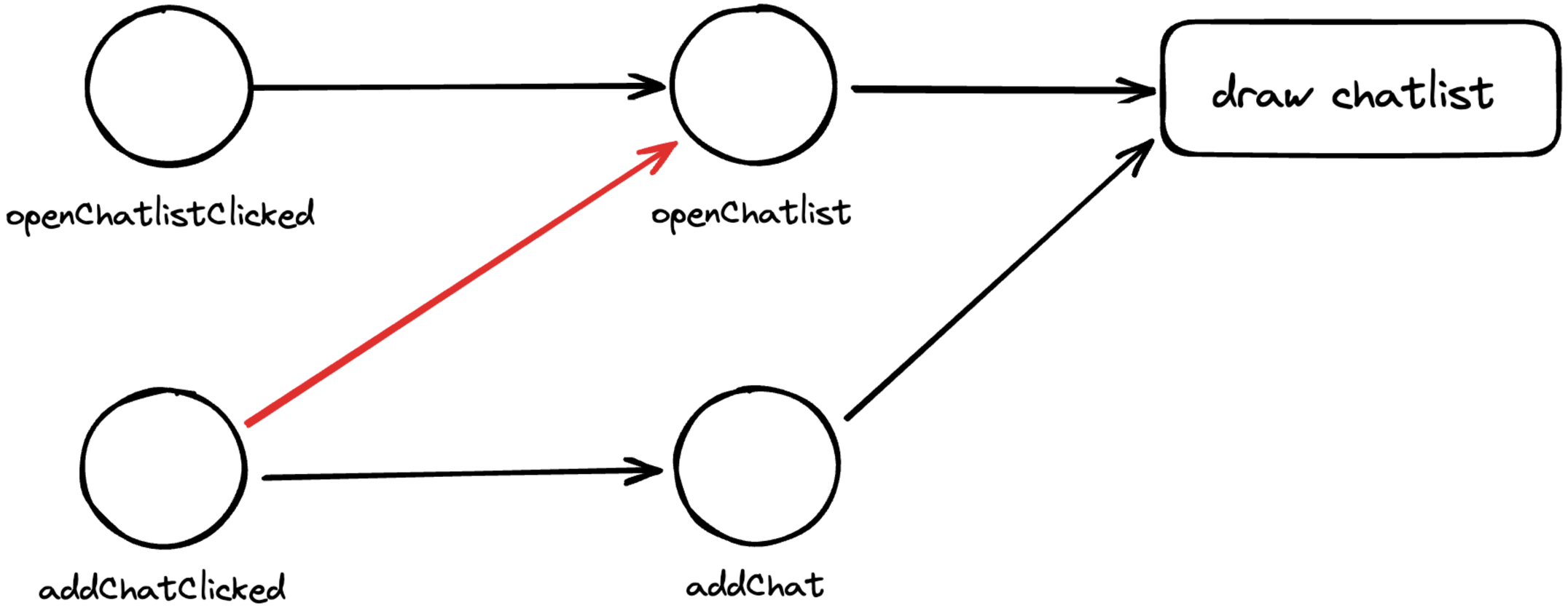
Пример: проектирование мессенджера



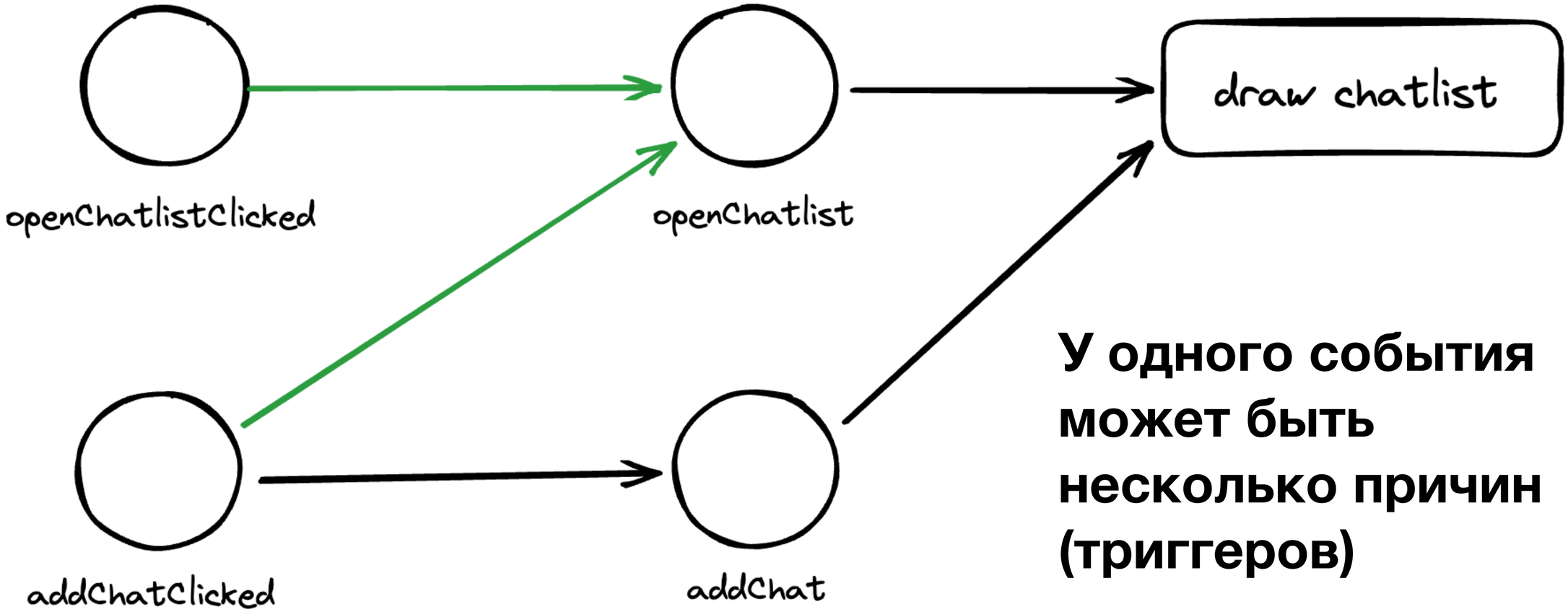
Пример: проектирование мессенджера



Пример: проектирование мессенджера

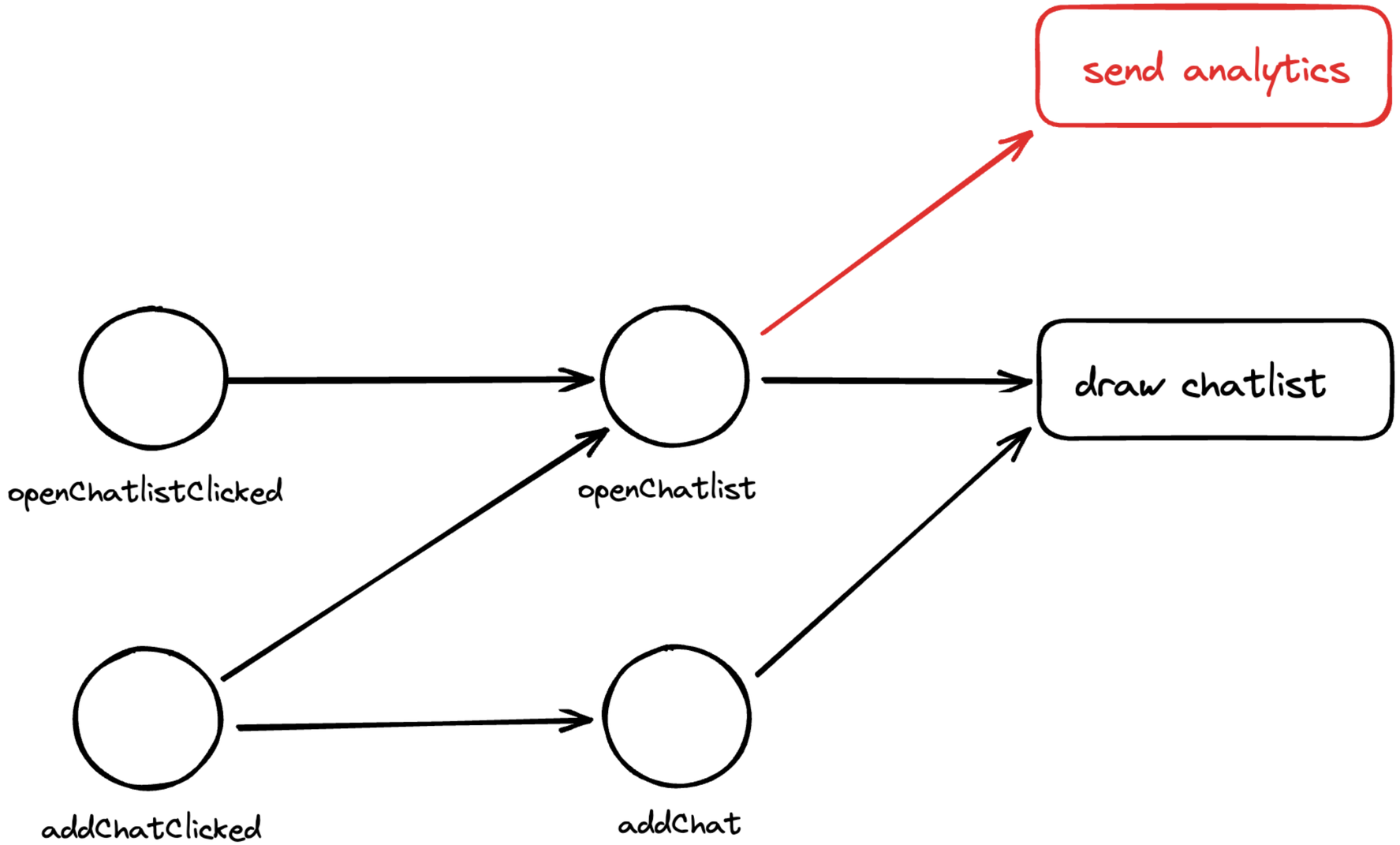


Пример: проектирование мессенджера

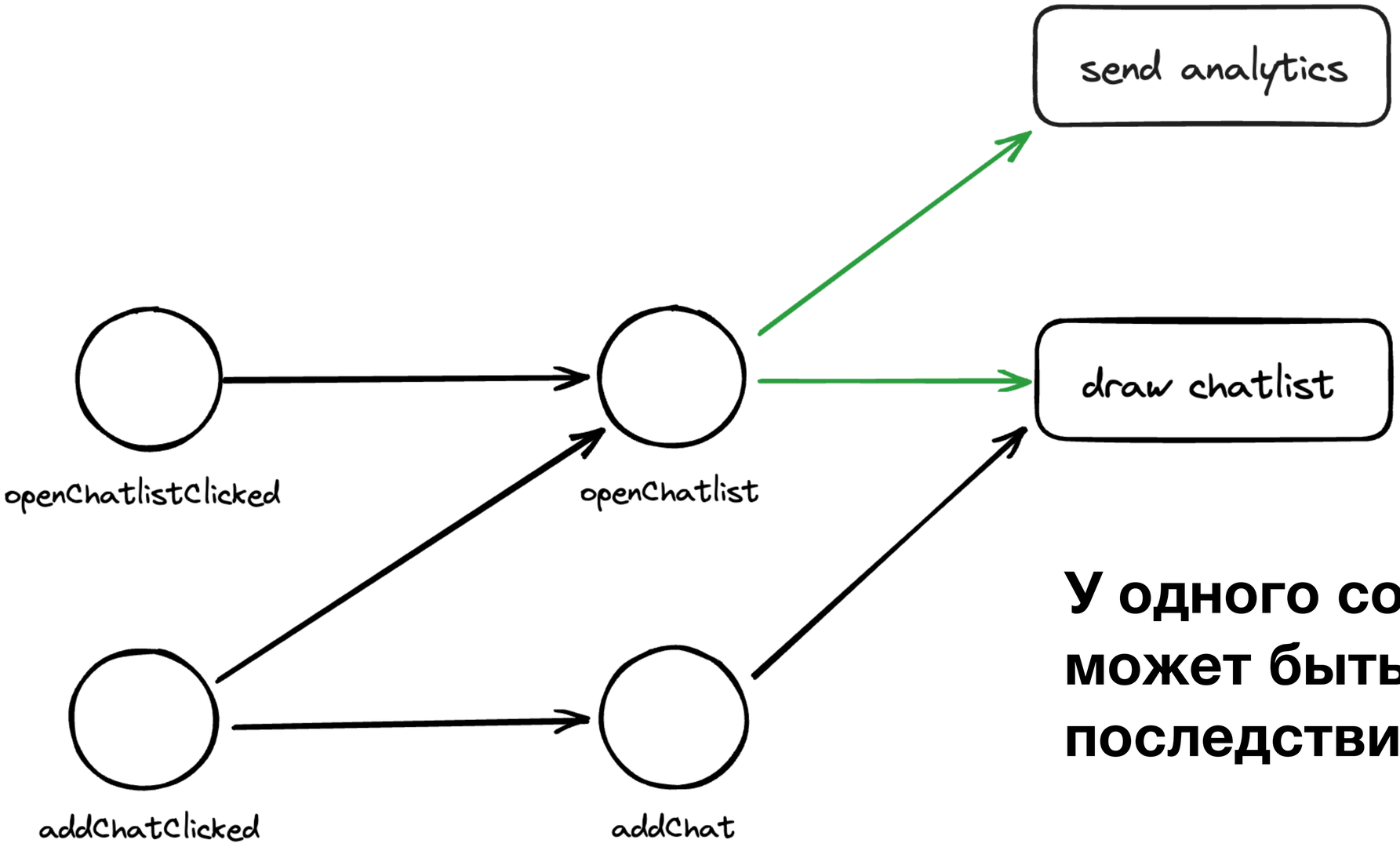


У одного события может быть несколько причин (триггеров)

Пример: проектирование мессенджера



Пример: проектирование мессенджера

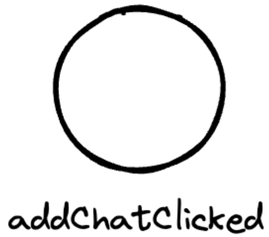


У одного события может быть несколько последствий

Концепция: событие (event)

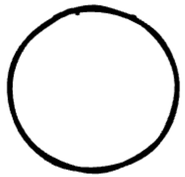
Концепция: событие (event)

Действие пользователя

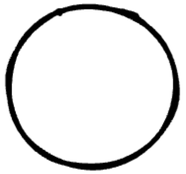


Концепция: событие (event)

Действие пользователя или команда на выполнение



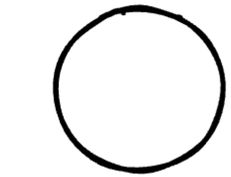
addChatClicked



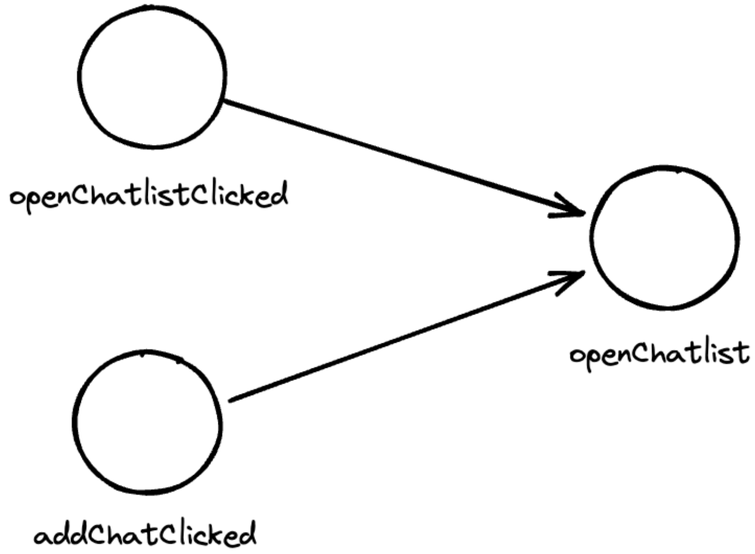
openChatlist

Концепция: событие (event)

Действие пользователя или команда на выполнение

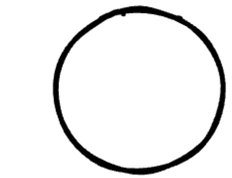


Может иметь несколько триггеров

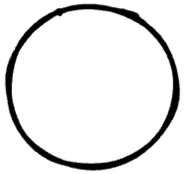


Концепция: событие (event)

Действие пользователя или команда на выполнение

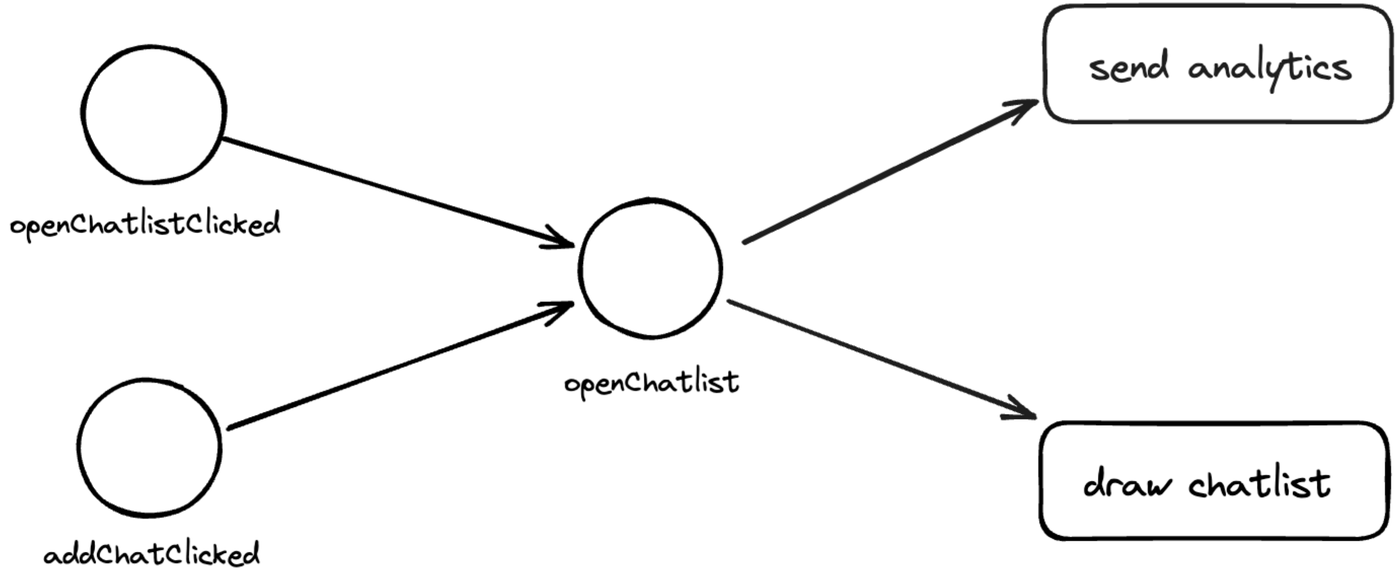


addChatClicked

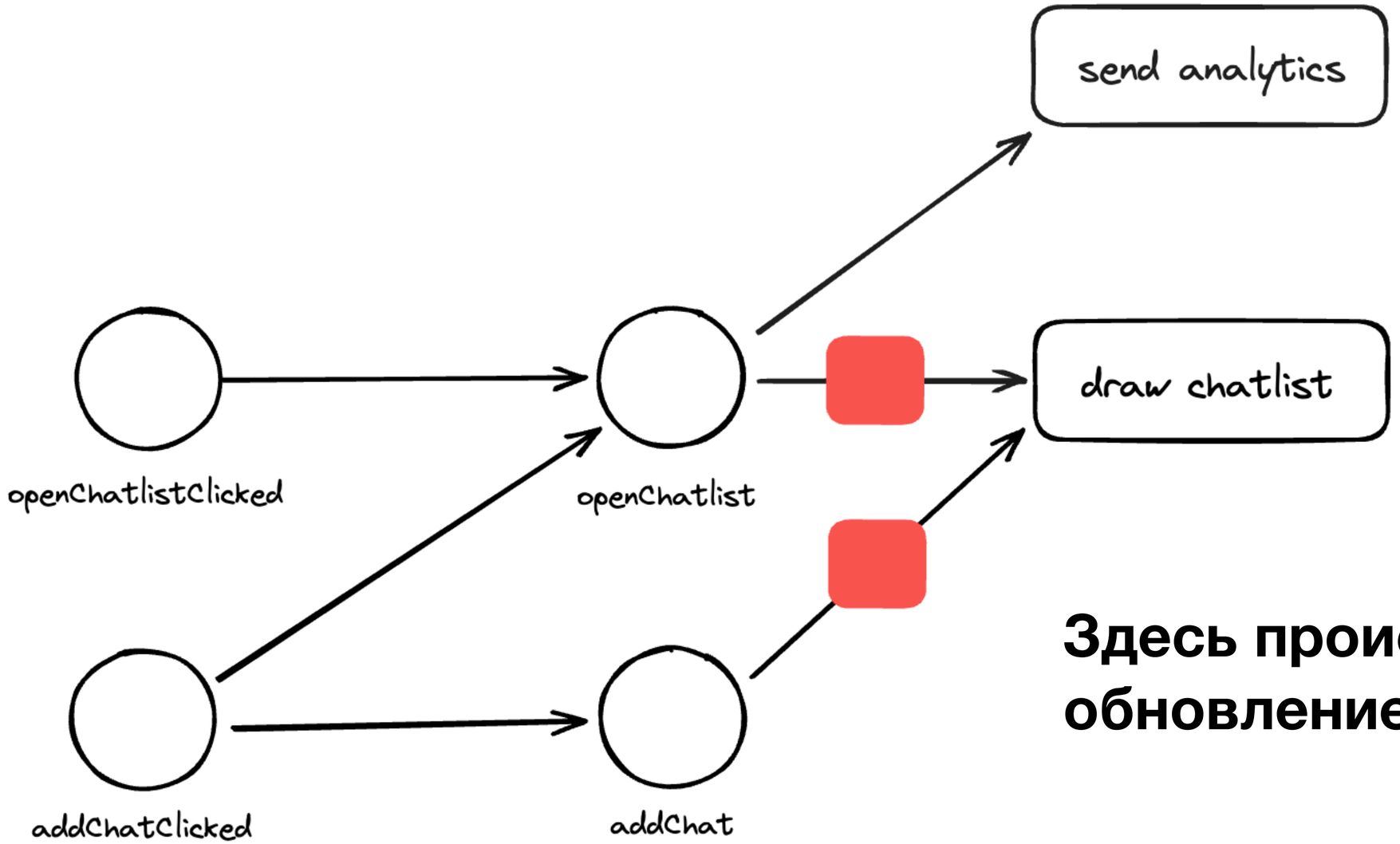


openChatlist

Может иметь несколько триггеров и последствий

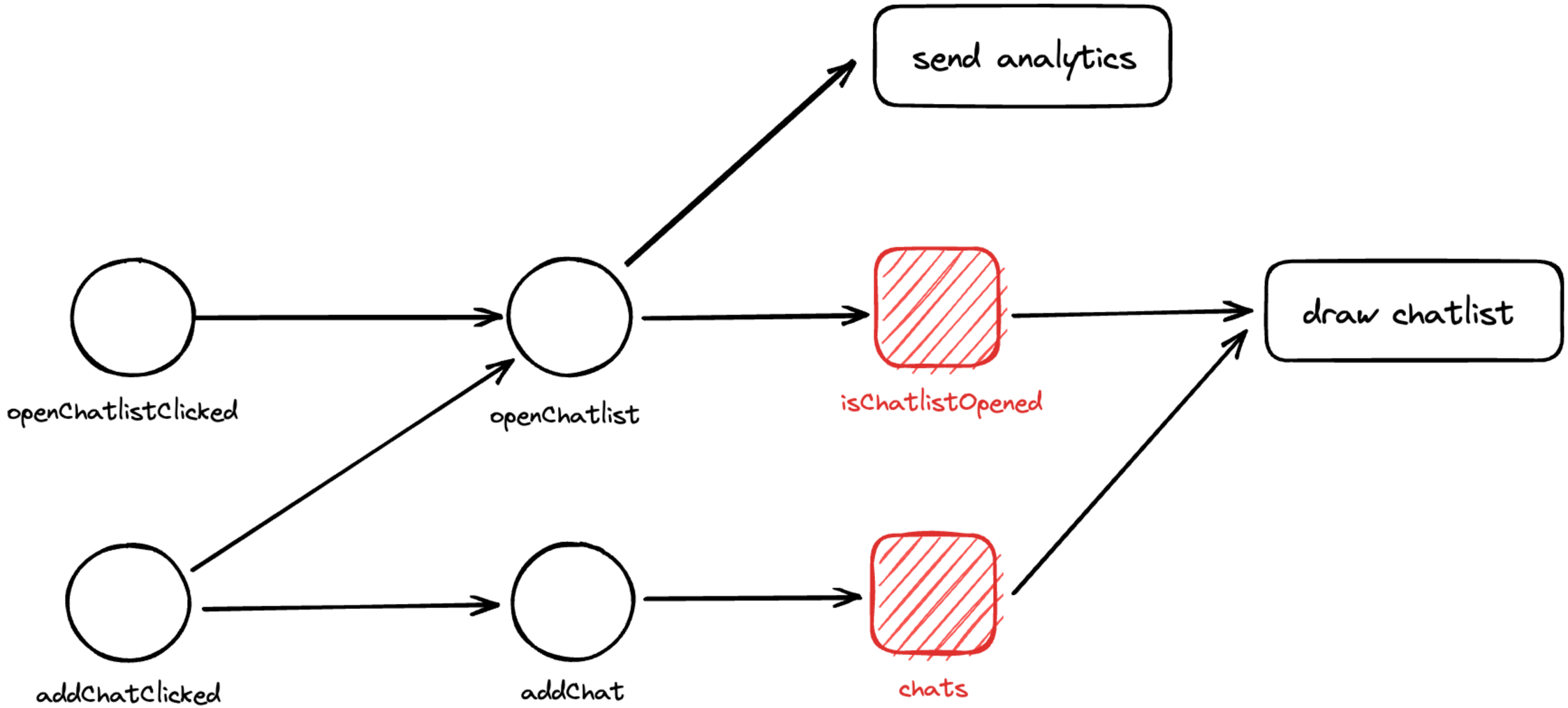


Пример: проектирование мессенджера

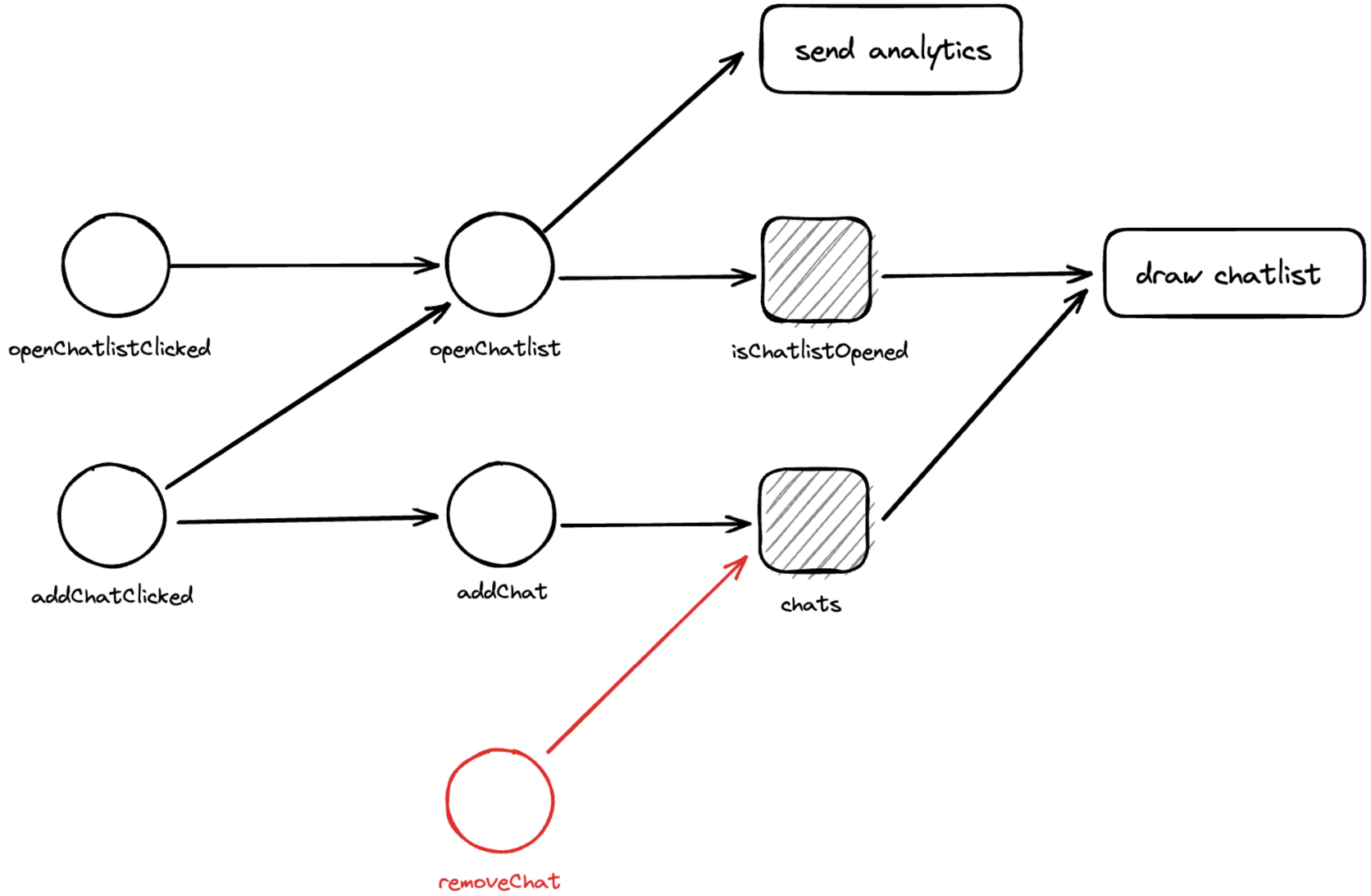


Здесь происходит обновление состояний

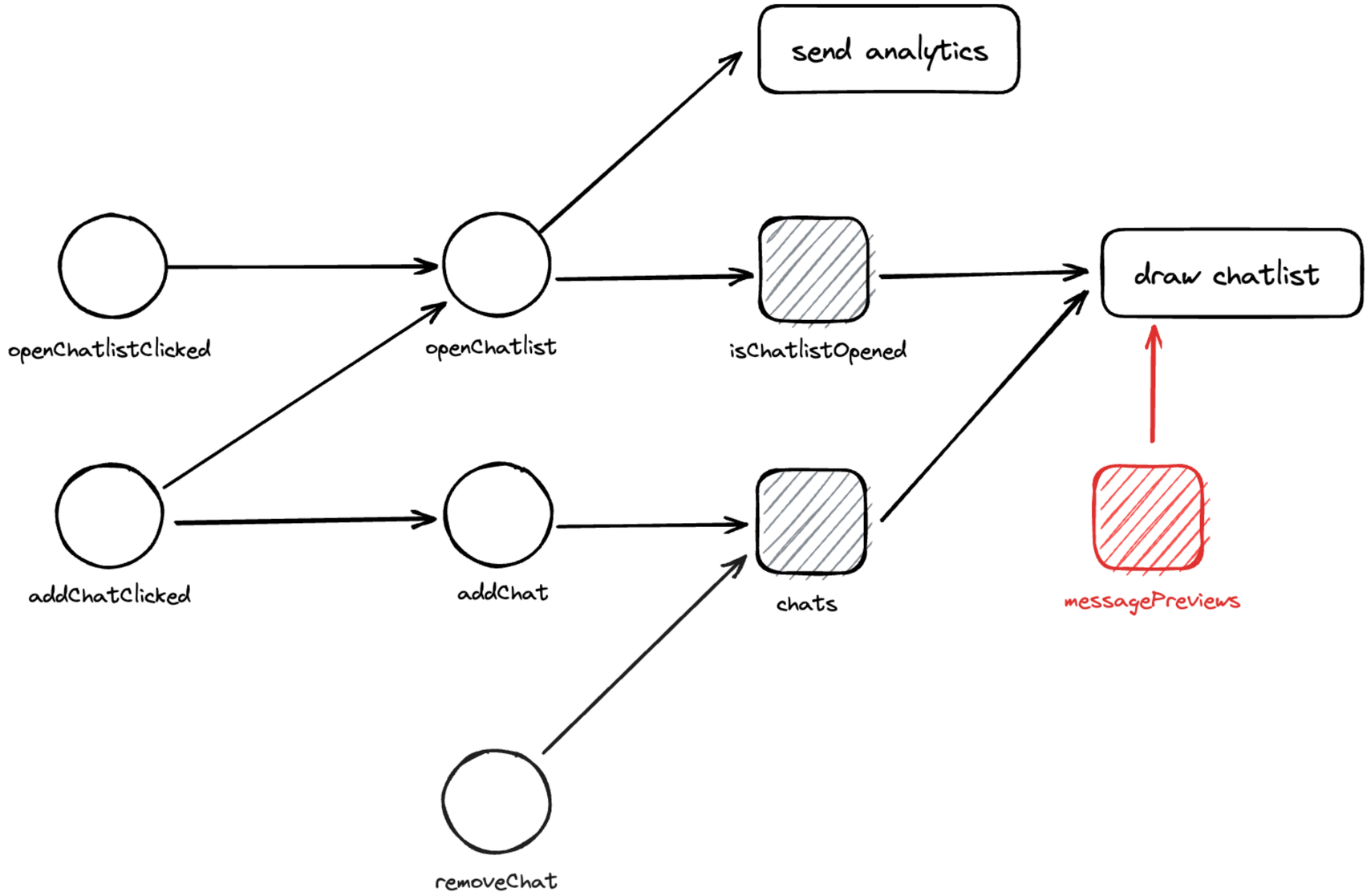
Пример: проектирование мессенджера



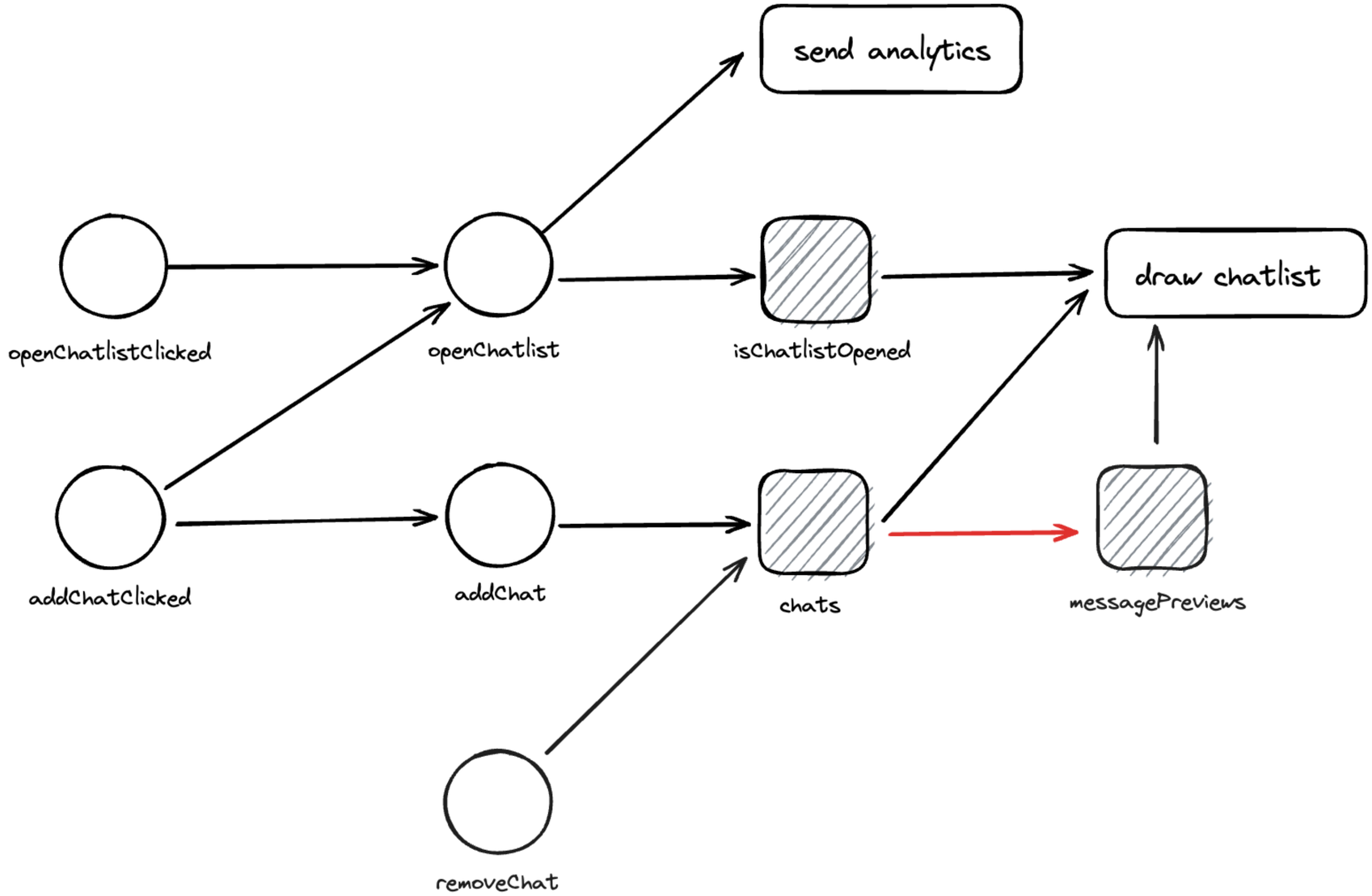
Пример: проектирование мессенджера



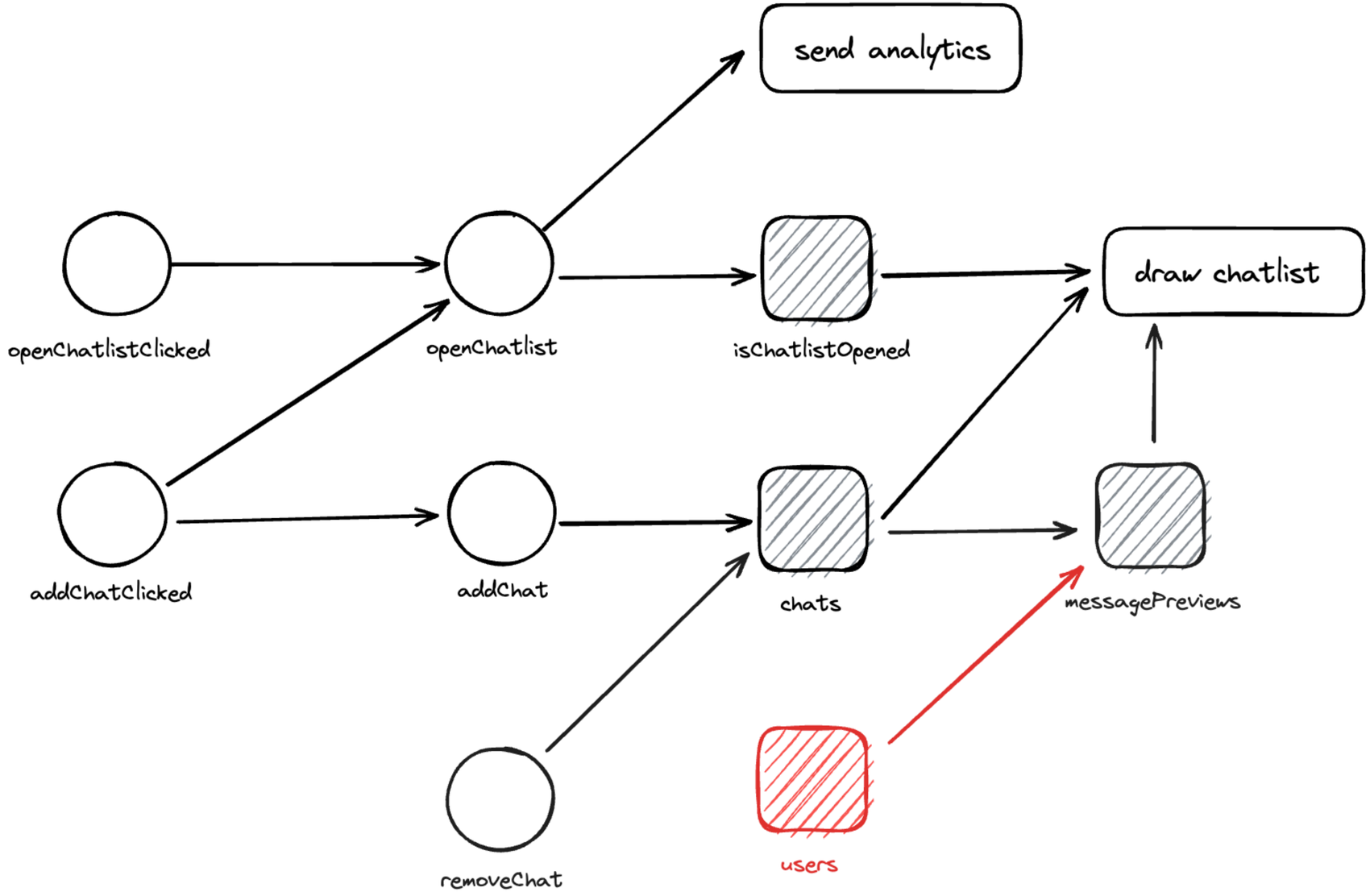
Пример: проектирование мессенджера



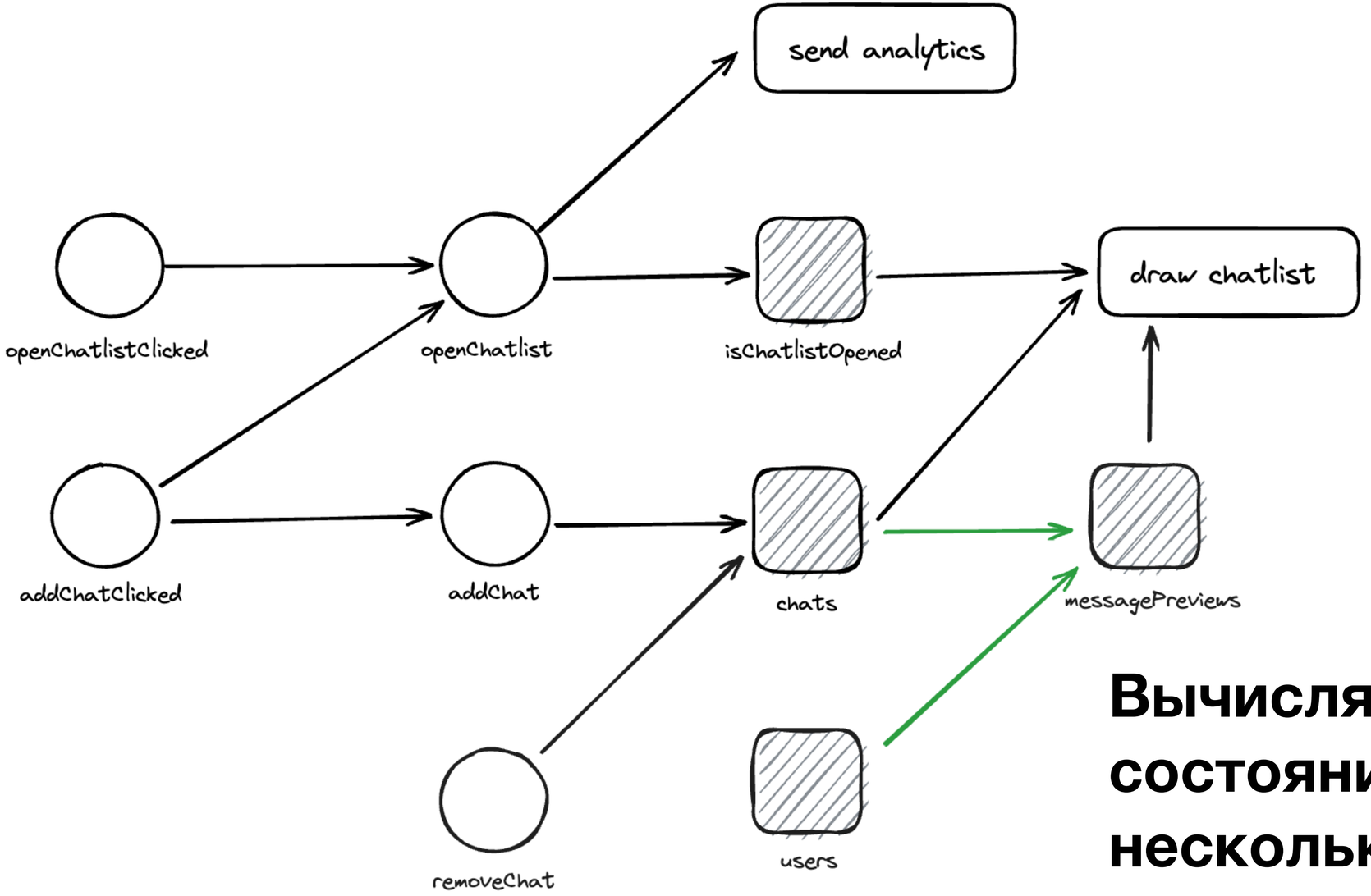
Пример: проектирование мессенджера



Пример: проектирование мессенджера



Пример: проектирование мессенджера



Вычисляемое состояние зависит от нескольких других

Концепция: Состояние (store)

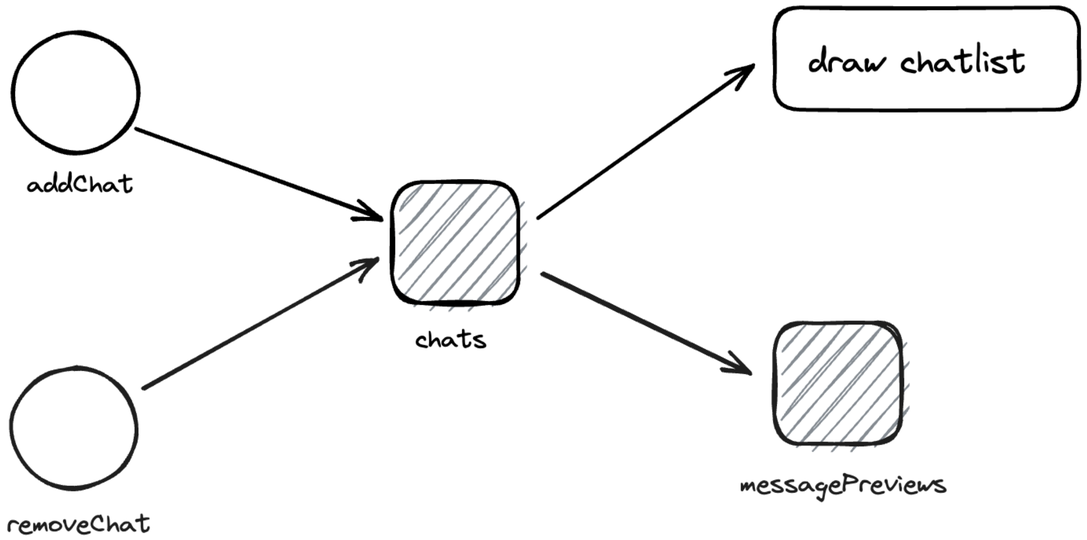
Концепция: Состояние (store)

Хранит данные приложения

Концепция: Состояние (store)

Хранит данные приложения

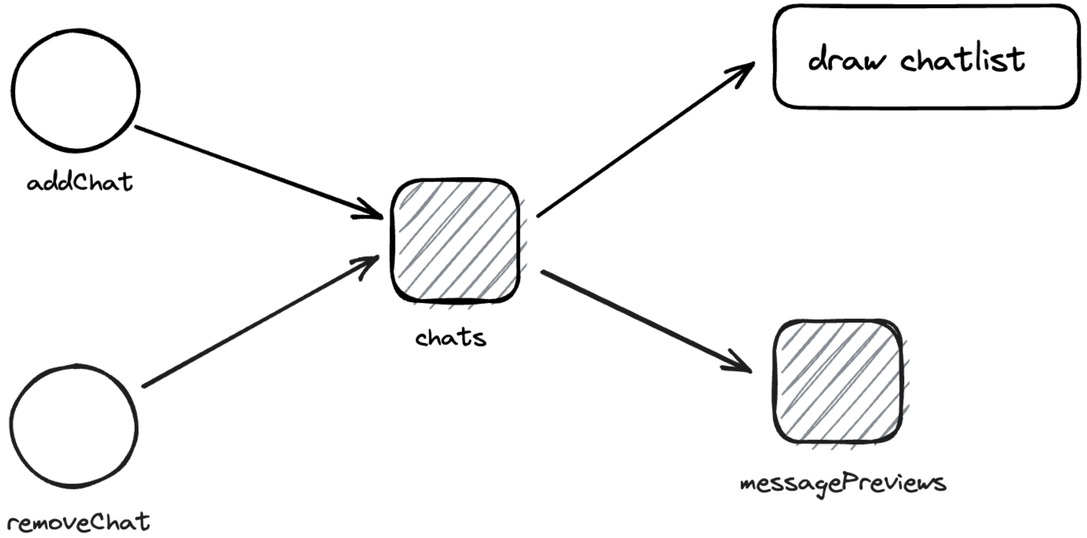
Может иметь несколько триггеров
и последствий



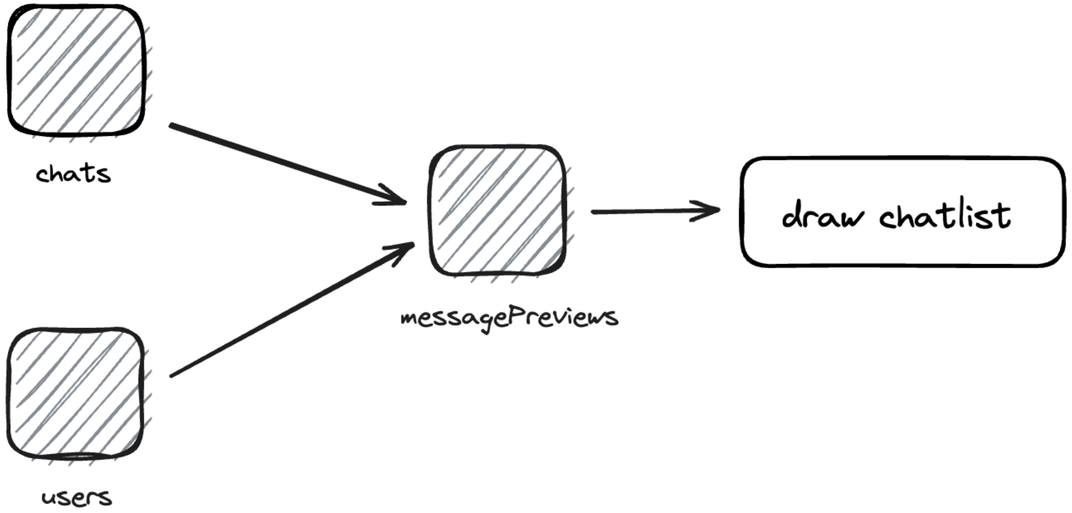
Концепция: Состояние (store)

Хранит данные приложения

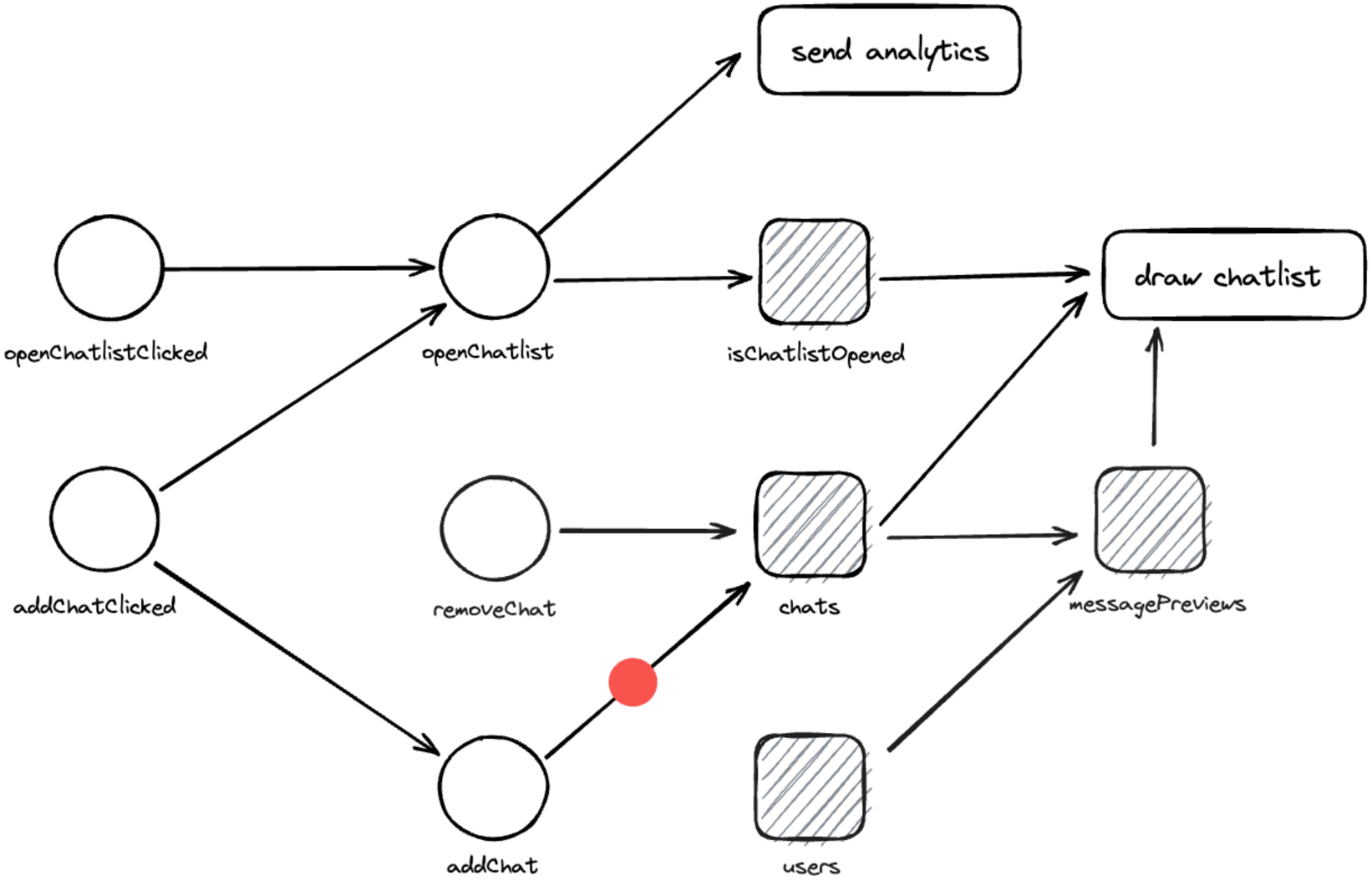
Может иметь несколько триггеров и последствий



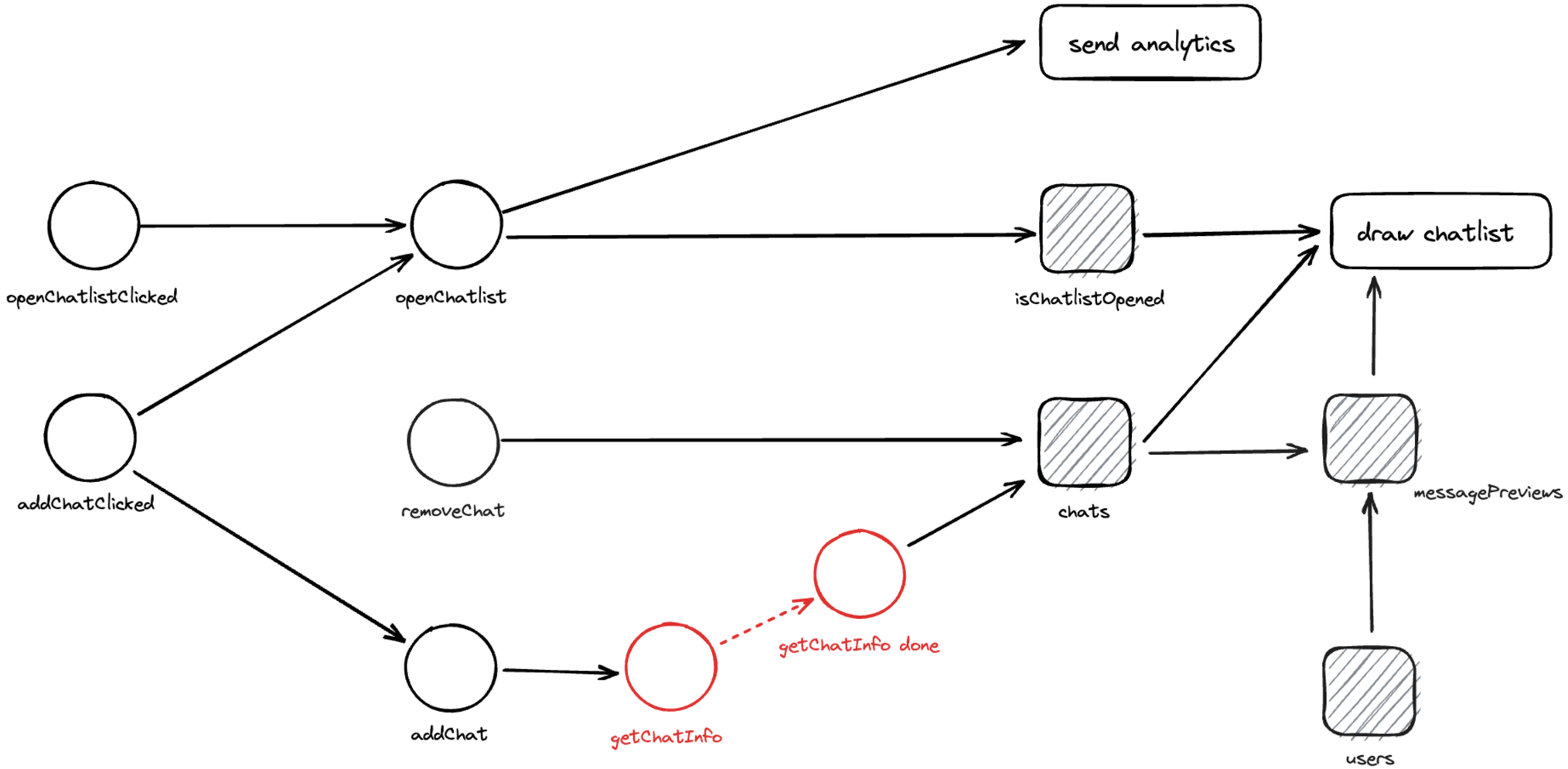
Может вычисляться на основе других сторов



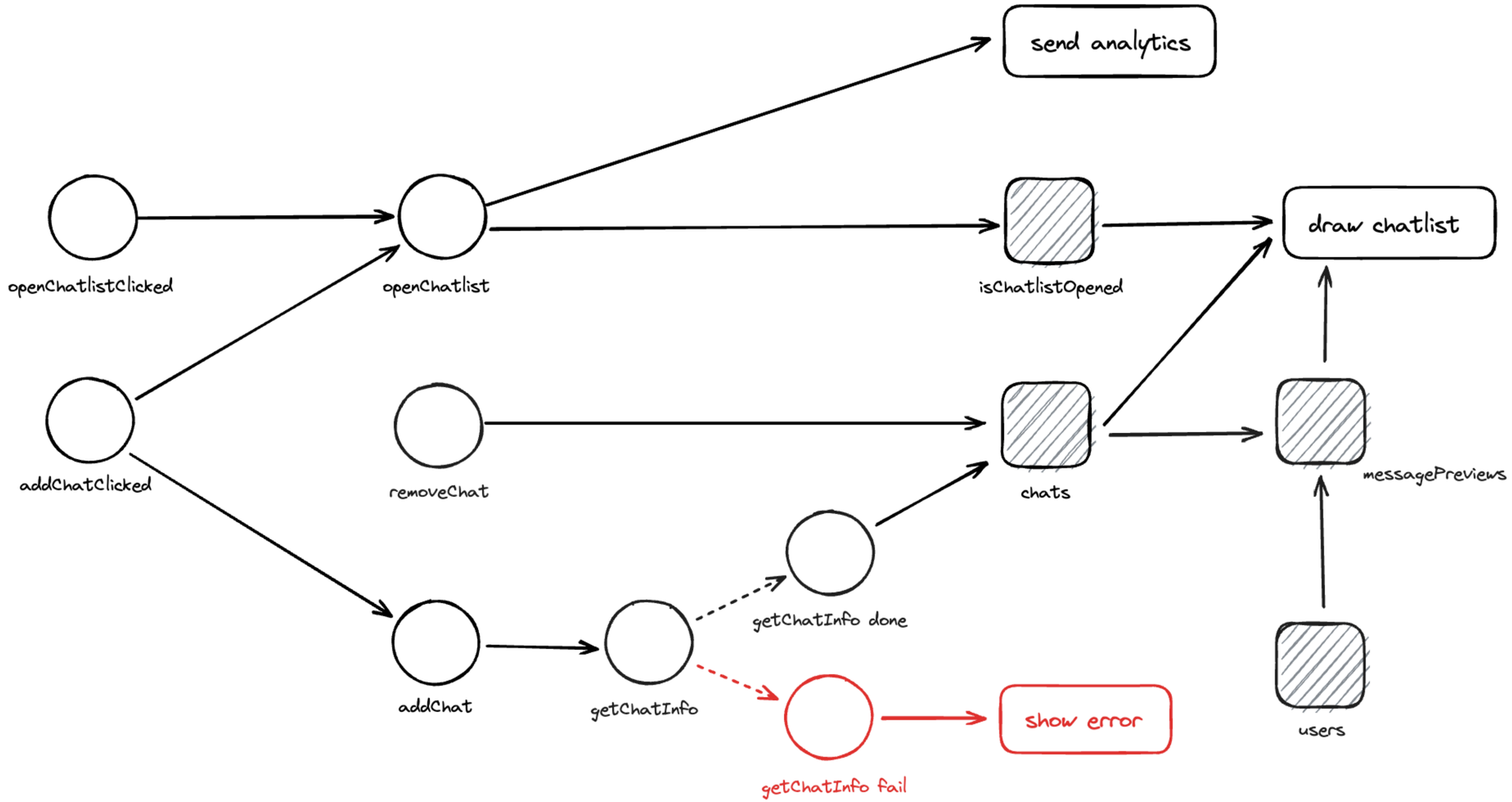
Пример: проектирование мессенджера



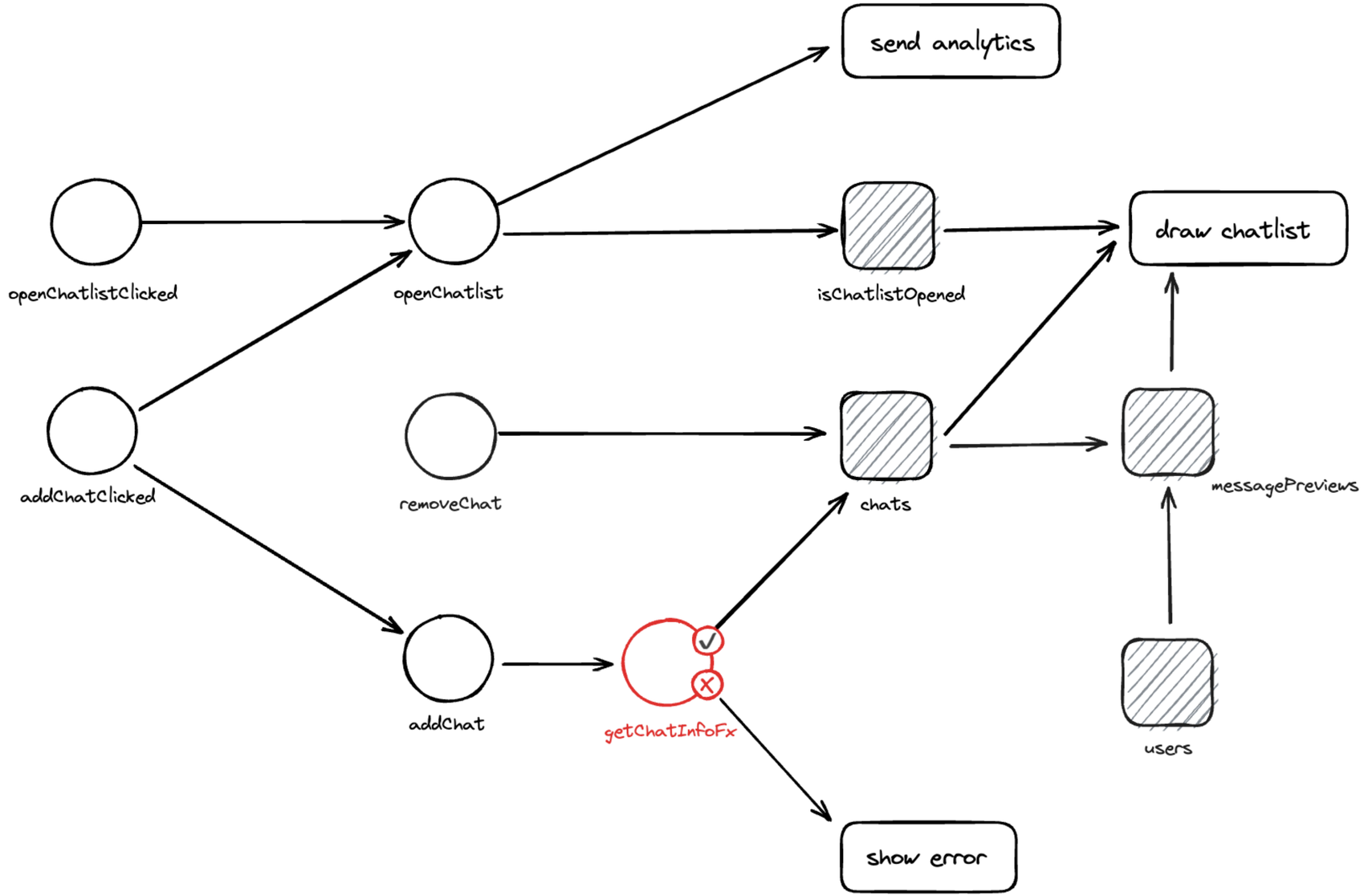
Пример: проектирование мессенджера



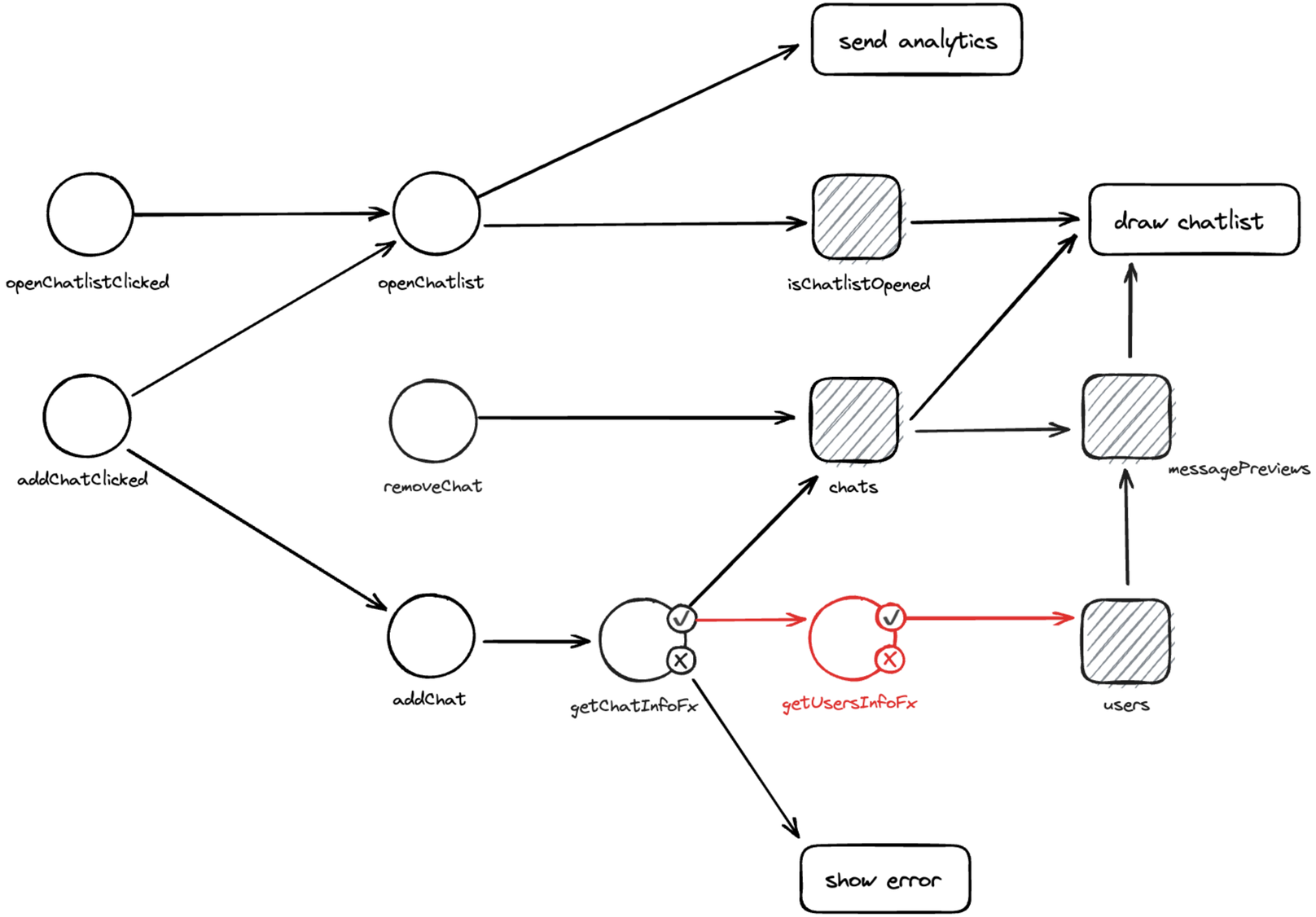
Пример: проектирование мессенджера



Пример: проектирование мессенджера



Пример: проектирование мессенджера



Концепция: эффект (effect)

Управляет работой с внешним миром

Концепция: эффект (effect)

Управляет работой с внешним миром

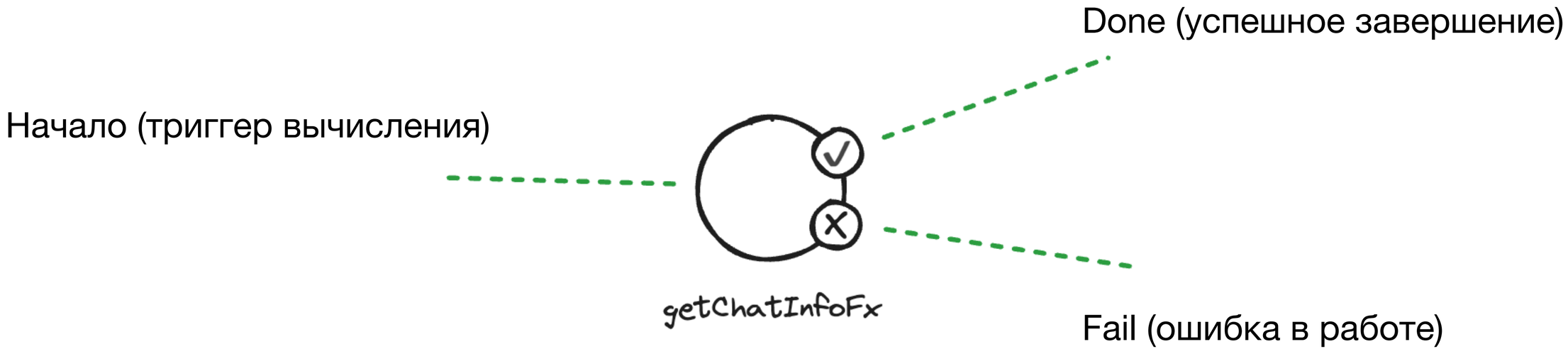
Сайд-эффект это вычисление которое имеет последствия

- Асинхронные операции
- Чтение и запись глобальных переменных
- Прямая работа с DOM, Web API
- Ресурсоемкие вычисления

Концепция: эффект (effect)

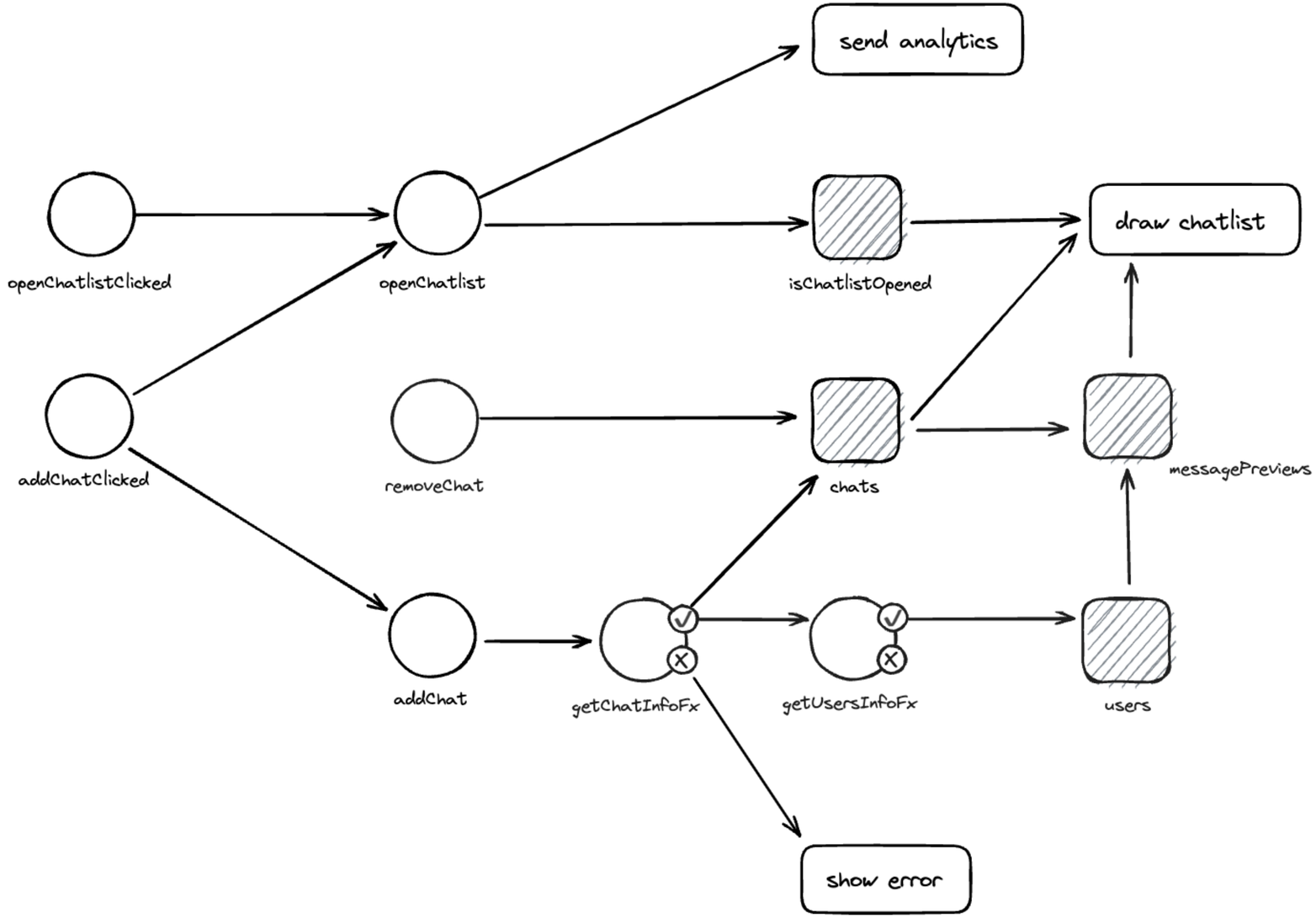
Управляет работой с внешним миром

Состоит из трёх триггеров и функции-обработчика

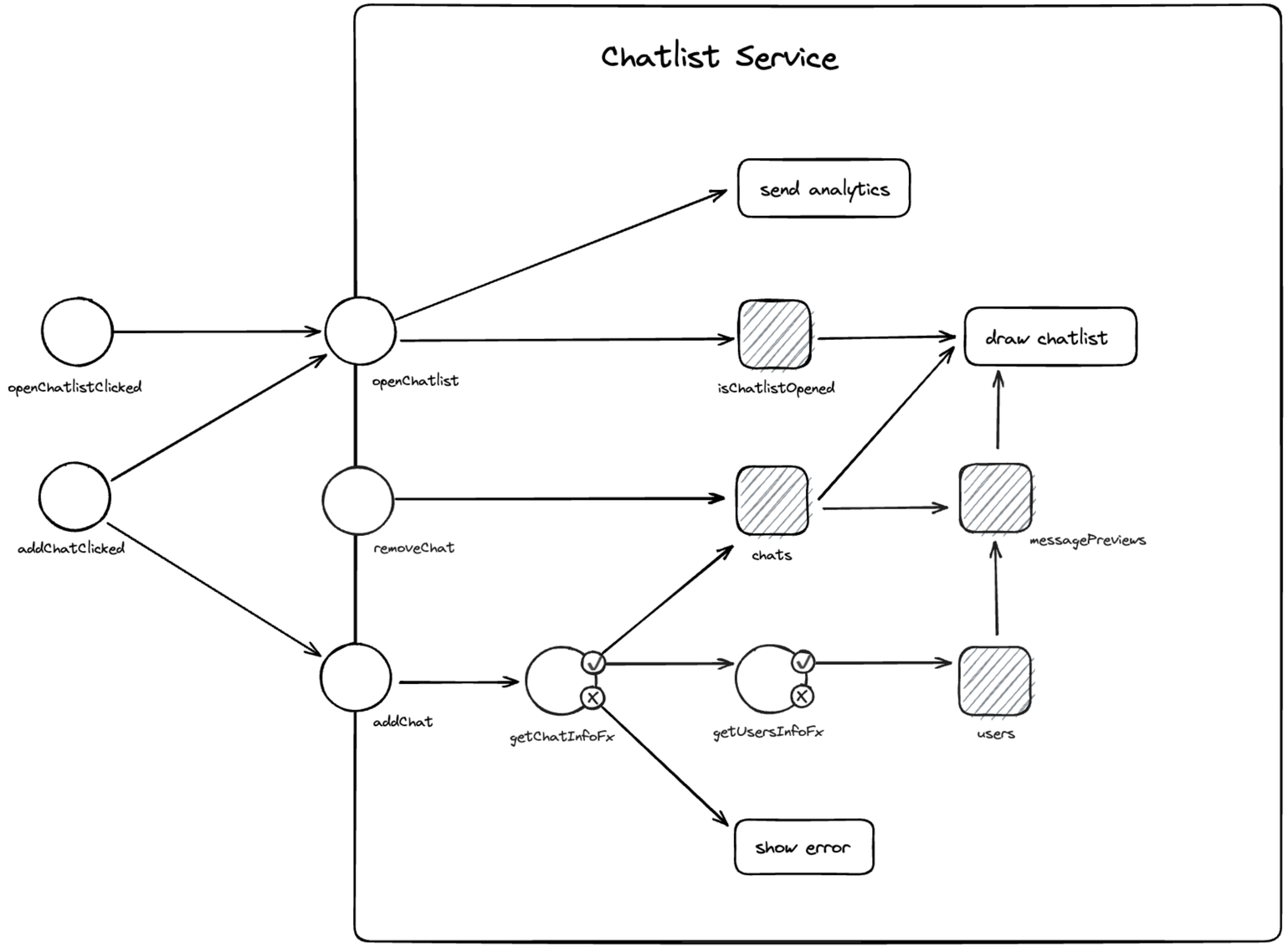


Композируется по тем же правилам что и другие сущности

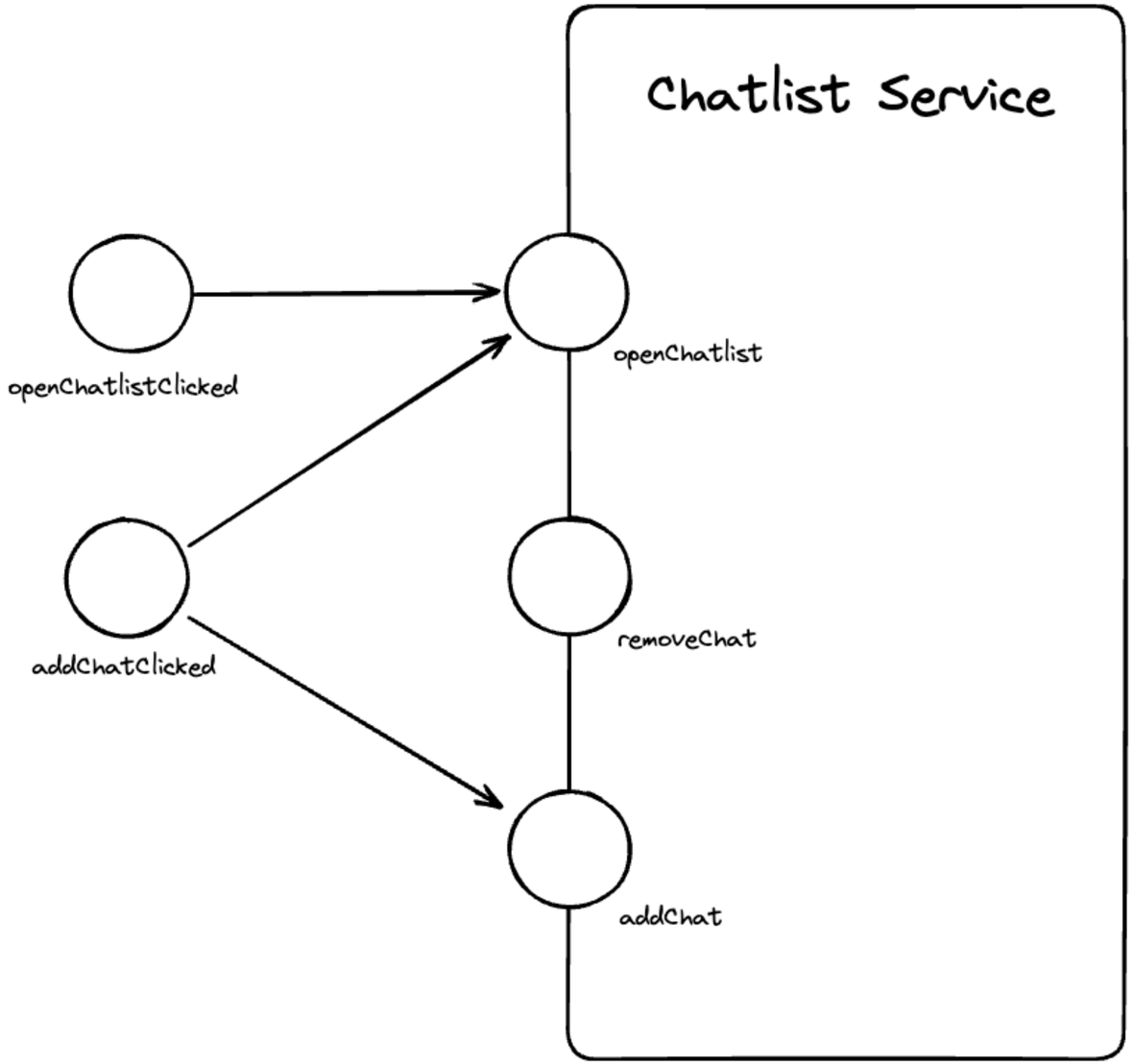
Пример: проектирование мессенджера



Пример: проектирование мессенджера



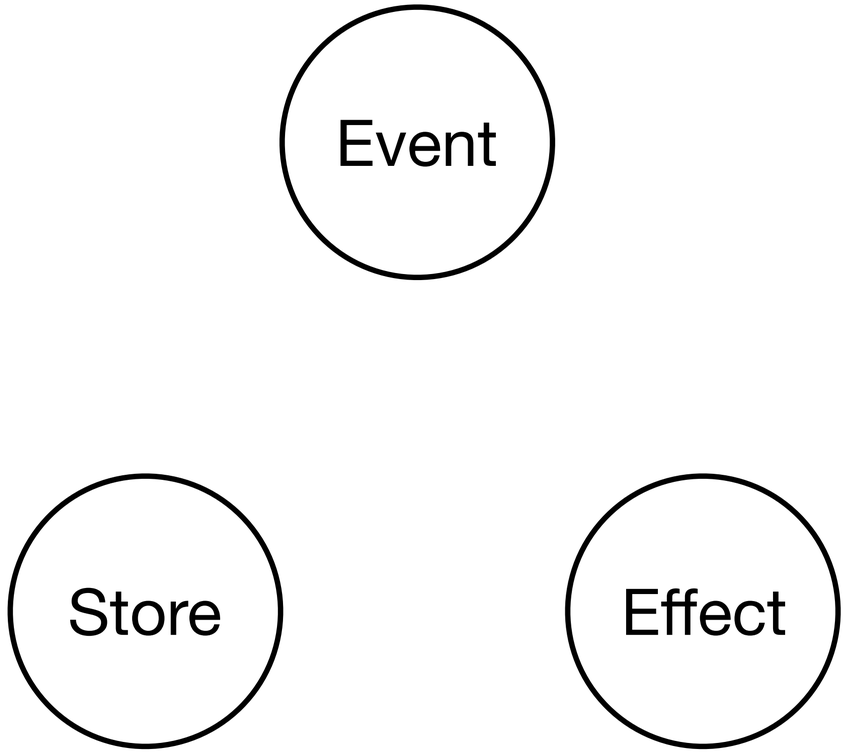
Пример: проектирование мессенджера



Effector - data flow manager

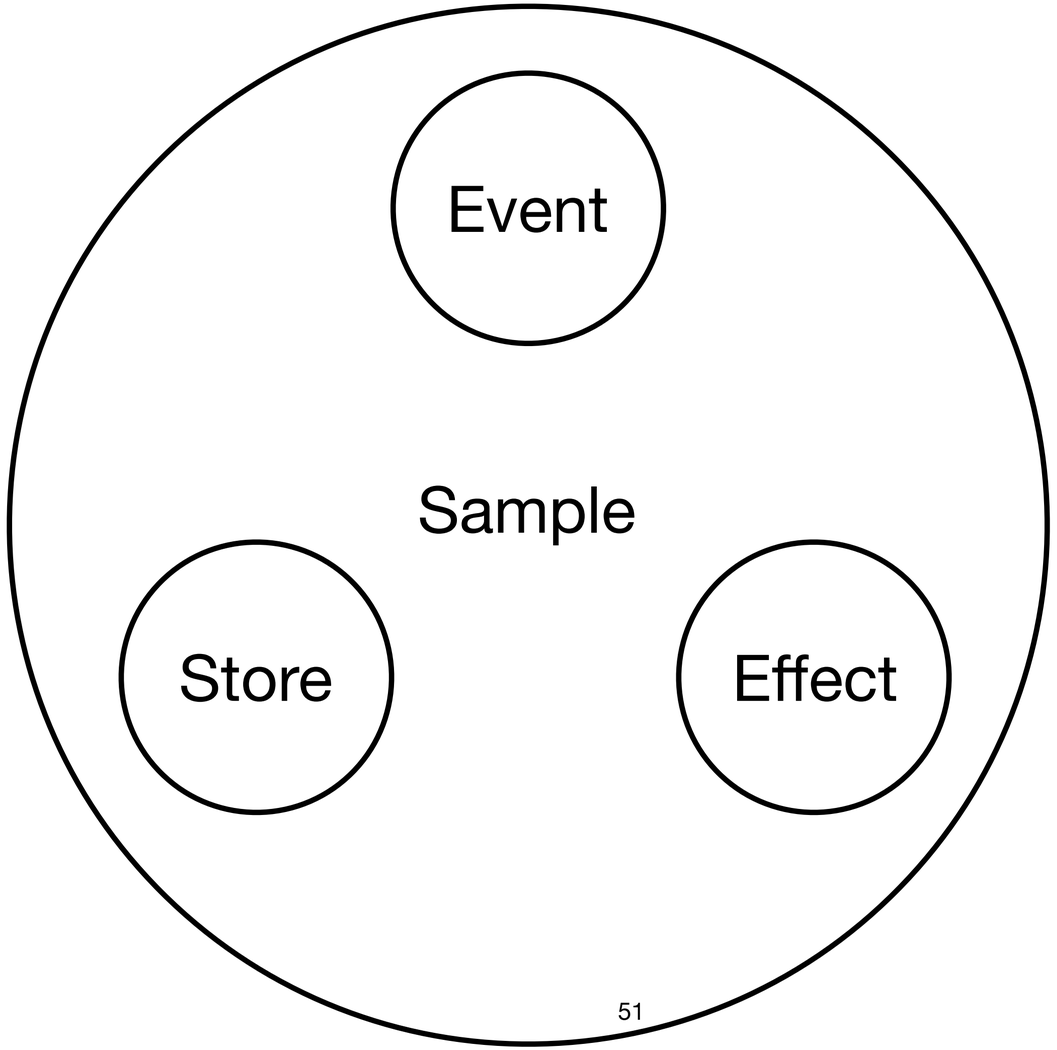
Управление состоянием в стейт менеджере это лишь часть задачи

Три сущности для описания приложения



Три сущности для описания приложения

И одна для связи между ними



Effector — декларативное описание логики приложений

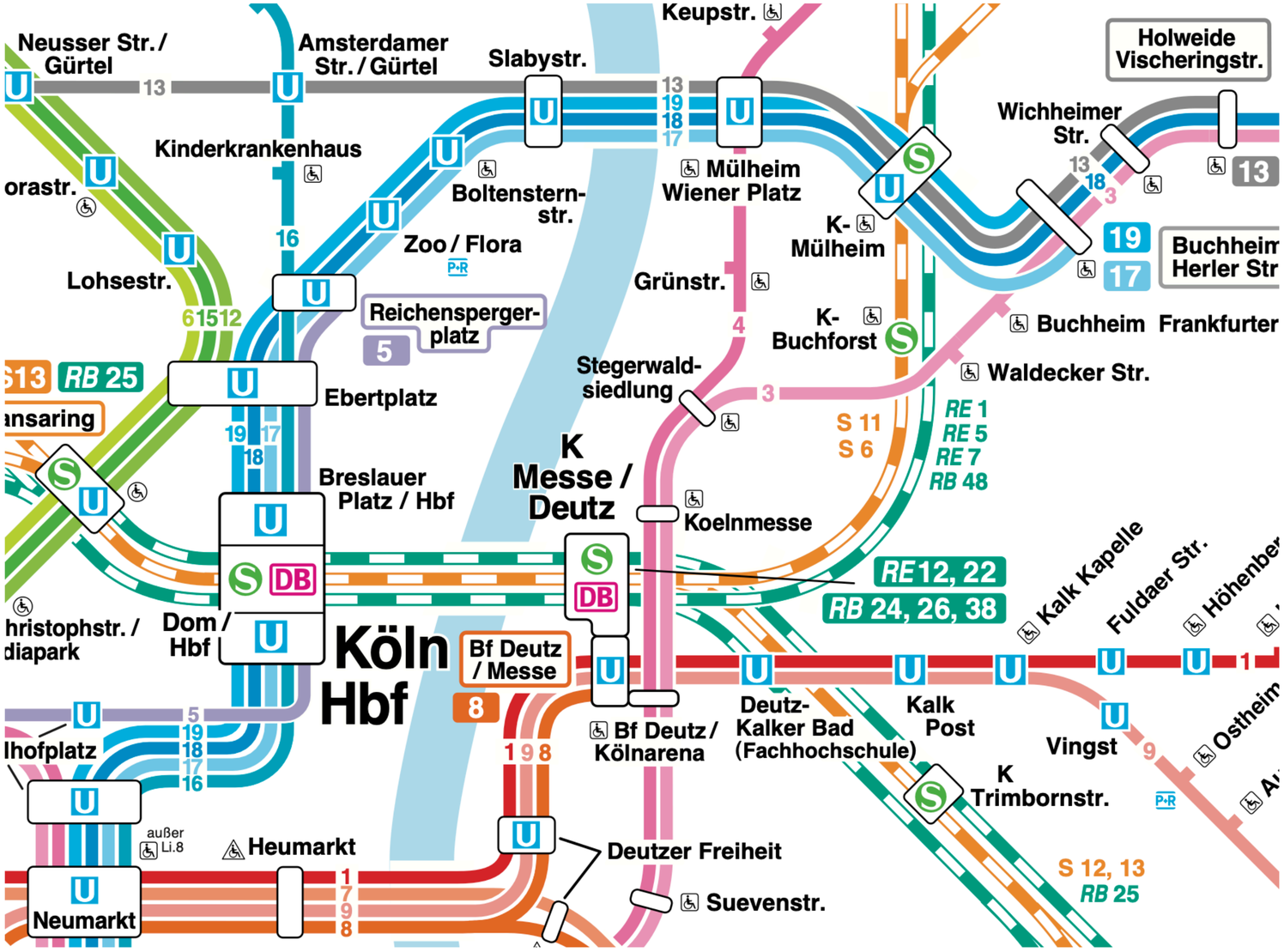
```
sample({
  // В какой момент? При клике на кнопку (событие)
  clock: clickSubmit,
  // При каком условии? Если форма валидна (состояние)
  filter: $isFormValid,
  // Какие данные необходимы? Состояние формы
  source: $formState,
  // Что сделать с данными? Отправить на сервер (сайд-эффект)
  target: submitFormFx,
})
```

Теперь у нас есть инструмент для проектирования бизнес-логики и концепции для её визуализации

Вторая проблема: как визуализировать?

Визуализация - задача, которая находится на стыке программирования, дизайна и теории графов

С похожей задачей сталкиваются при проектировании карт метро



An Improved Algorithm for the Metro-Line Crossing Minimization Problem, Martin Nöllenburg, 2009

Проблему визуализации изучают более 40 лет

Further Towards Unambiguous Edge Bundling: Investigating Power-Confluent Drawings for Network Visualization

Jonathan X. Zheng, Samraat Pawar, and Dan F. M. Goodman

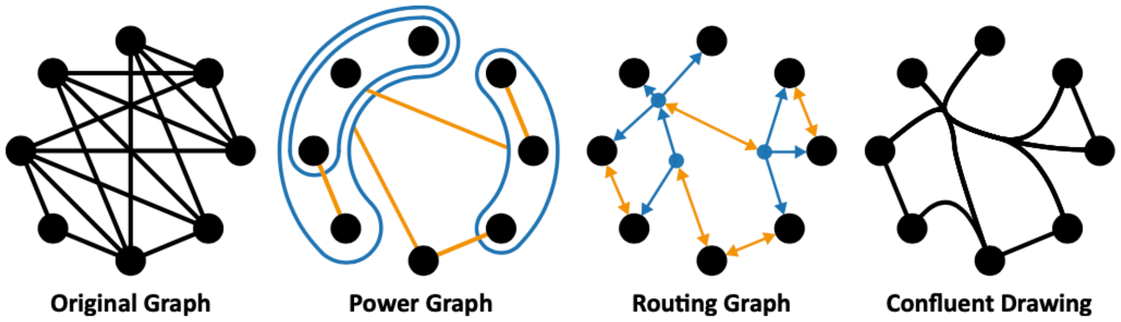


Fig. 1. An illustration of our solution to the issues with the algorithm of Bach et al. [1], using their original example graph (Fig. 2 in [1]). From left to right: a conventional node-link diagram with vertices arranged around a circle; its power graph decomposition, where edges are compressed by grouping similar nodes together; our novel equivalent routing graph, where the nested structure of groups is represented by a directed tree; the resulting confluent drawing, where original edges are threaded through the routing graph to form bundled junctions.

Abstract— Bach et al. [1] recently presented an algorithm for constructing confluent drawings, by leveraging power graph decomposition to generate an auxiliary routing graph. We identify two issues with their method which we call the node split and short-circuit problems, and solve both by modifying the routing graph to retain the hierarchical structure of power groups. We also classify the exact type of confluent drawings that the algorithm can produce as 'power-confluent', and prove that it is a subclass of the previously studied 'strict confluent' drawing. A description and source code of our implementation is also provided, which additionally includes an improved method for power graph construction.

Index Terms—Graph drawing, power graph decomposition, edge bundling, confluent drawing, optimization

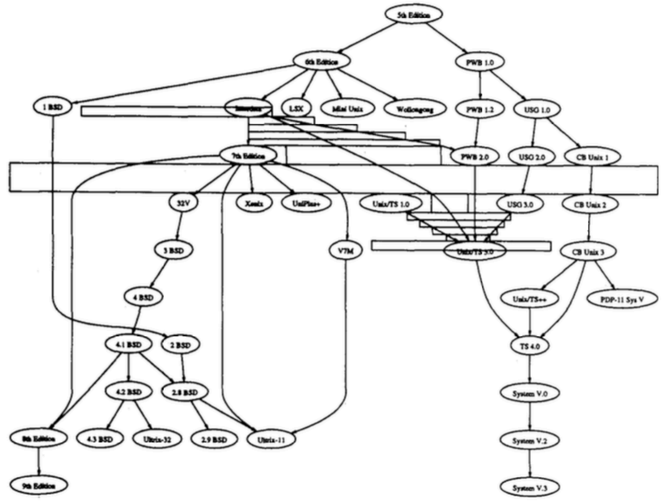


Fig. 16. Region for a spline (0.48 s user time Sun 4/280).

B_i has edges in common with B_{i-1} and B_{i+1}; q and r are points on or inside the first and last box respectively, find $s_0, \ldots, s_n$ and $BB_0, \ldots, BB_m$, where s_i are the control points of a piecewise Bezier curve and BB_i are boxes parallel to the coordinate axes. The curve must have q and r as its endpoints. θ_q and θ_r are optional; if they are specified, then the curve must have the given slope at the corresponding endpoint. The BB_i correspond to the B_i and are the smallest boxes that contain the generated splines.

We next describe the two parts of the algorithm.

A. Finding the Region

There are three kinds of edges in the drawing: edges between nodes on different ranks, flat edges between different nodes on the same rank, and self-edges or loops.

Different Ranks: In practice, most edges connect nodes on different ranks. The region for this kind of edge has a few boxes near its tail port, then an alternating sequence of inter-

first because they can often be drawn as straight lines, but the order does not seem to affect the drawing quality much.

There are three details that can help to improve the appearance of the splines. First, when edges cross, they should not constrain each other too much, otherwise, a spline may have an awkward, sharp turn. This is easily avoided by making an adjustment to the boxes. When setting the size of a box, we ignore virtual nodes to the left or right that correspond to edges that cross within two ranks. Crossings further away are not considered because unintended multiple crossings can occur when the boxes become too sloppy.

Second, when an edge has a section that is almost vertical, it looks better to just draw it as a vertical line. This is most obvious when edges run alongside each other, because parallel line segments look better than long segments with slightly different slopes. When the region finding procedure detects a long vertical section, it terminates the current region, draws its spline, draws the vertical line segment, and finally begins the

Проблему визуализации изучают более 40 лет

Список основных проблем:

- Расположение блоков
  - Стабильное расположение при разворачивании/сворачивании блоков
  - Компактность расположения блоков
- Расположение текста
- Прокладывание (роутинг) связей между сущностями
  - Минимизация пересечений линий
  - Бандлинг (склеивание) линий в зависимости от контекста
- Расположение портов (точек входа и выхода из блока)

Решение: комбинация алгоритмов, действующая вместе над конкретной задачей

Для роутинга линий требуются вычисления в целочисленном поле

Значит алгоритм расстановки блоков тоже должен работать в нём (treemap не подойдет)

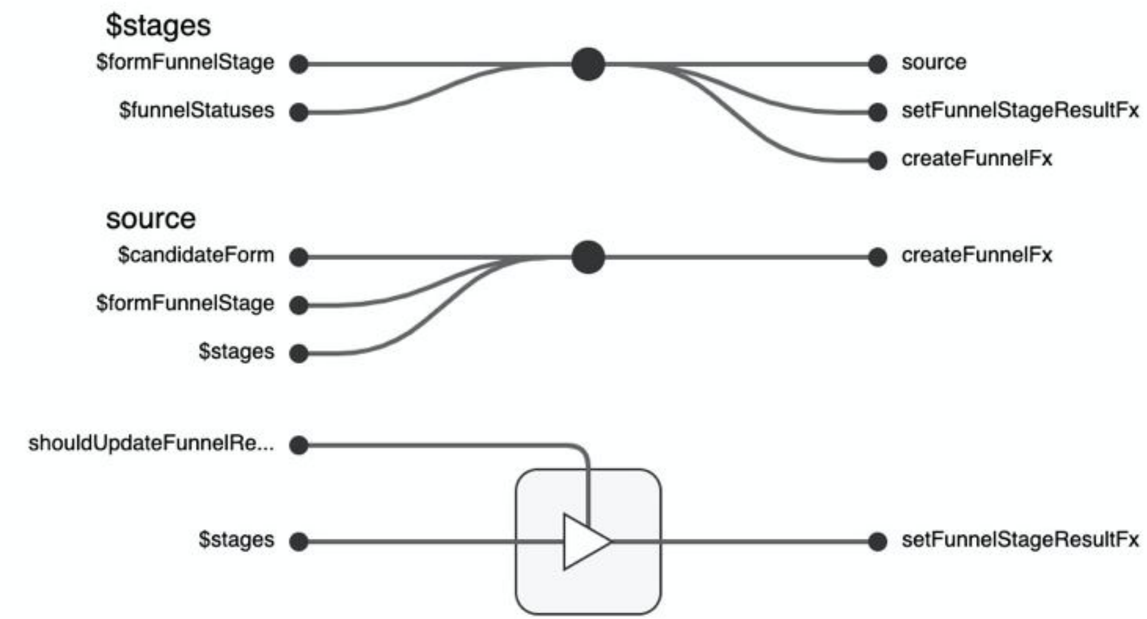
Под условия подходят:

- **bin packing** для блоков
- **A*** для роутинга линий

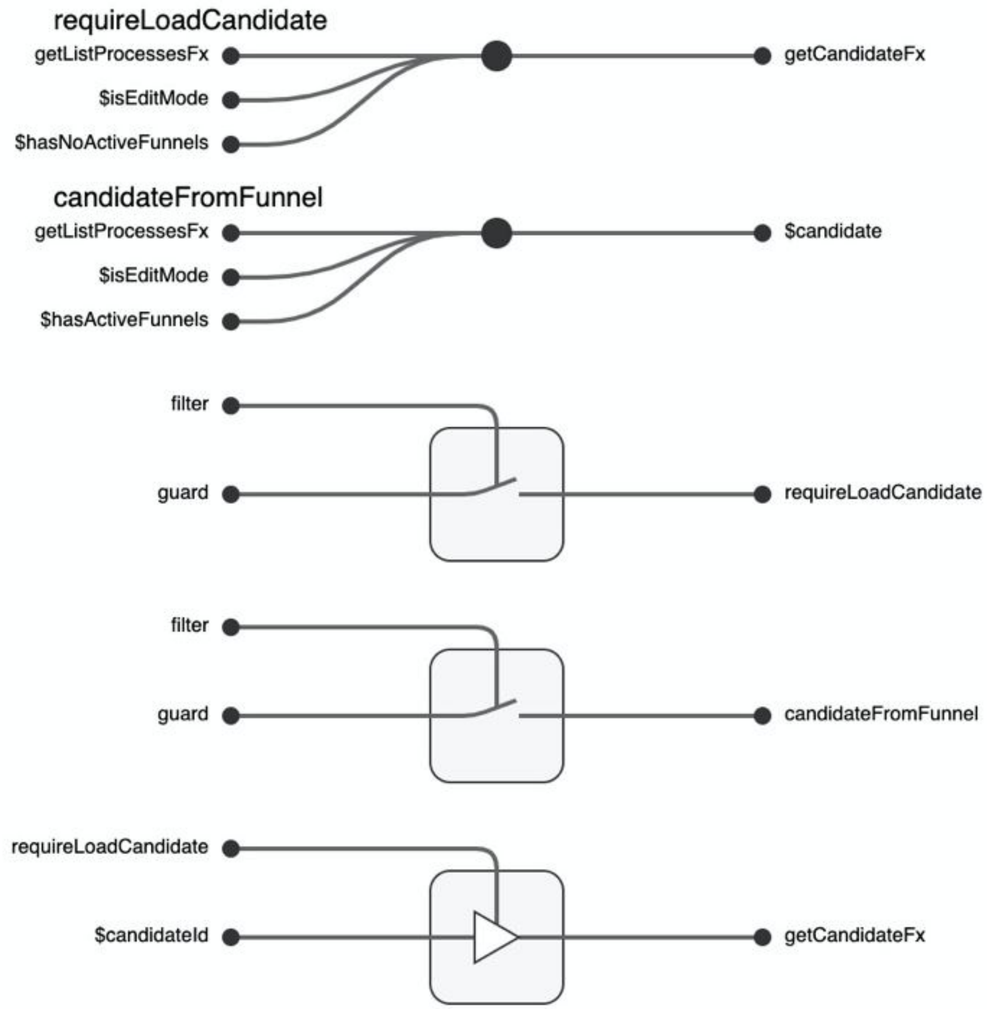
Поэтому существующие решения не подходят, ни низкоуровневые, вроде d3, ни общие, вроде yFiles, GoJS или Cytoscape

Визуализация связей с конкретным юнитом

/features/candidate-management/scenarios/save

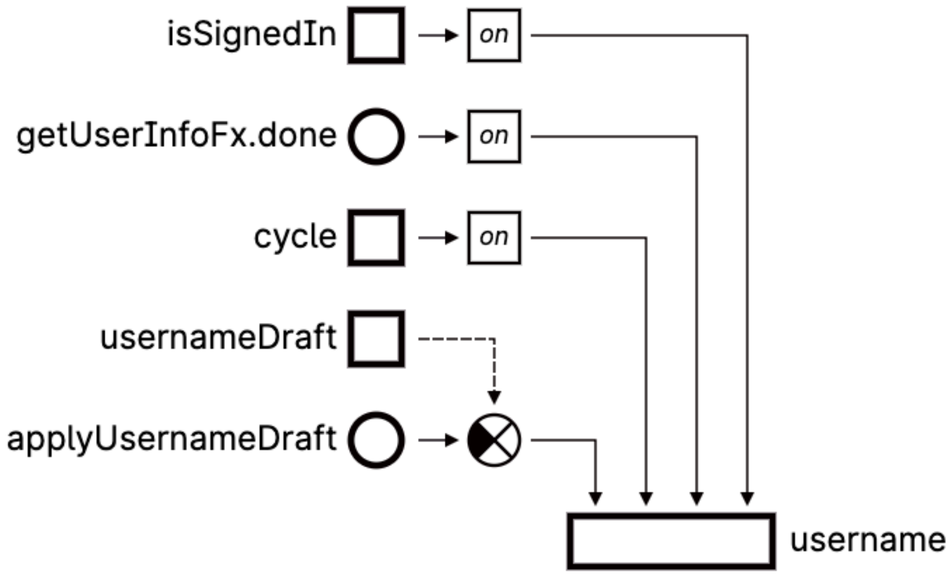


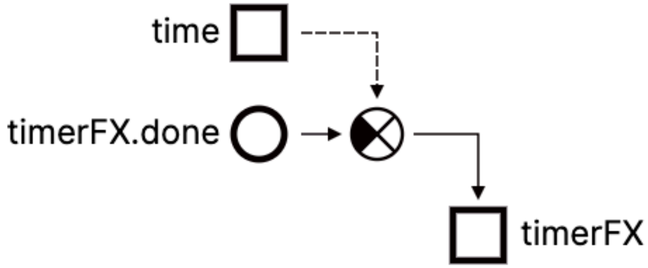
/features/candidate-management/scenarios/load-profile

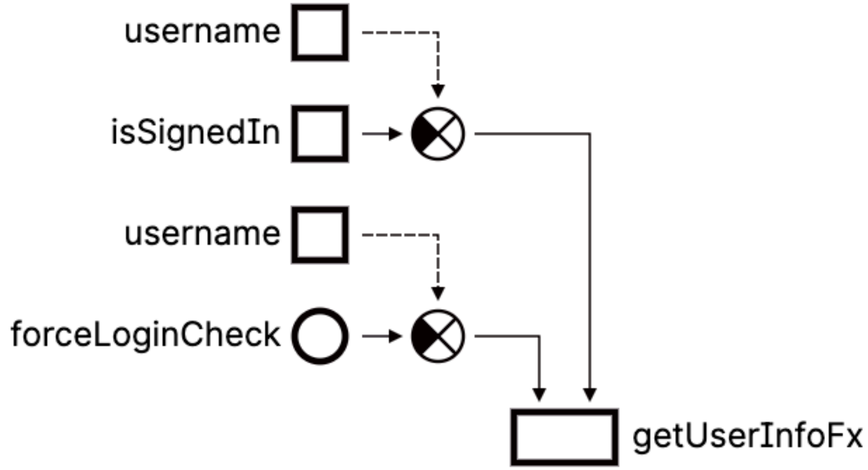


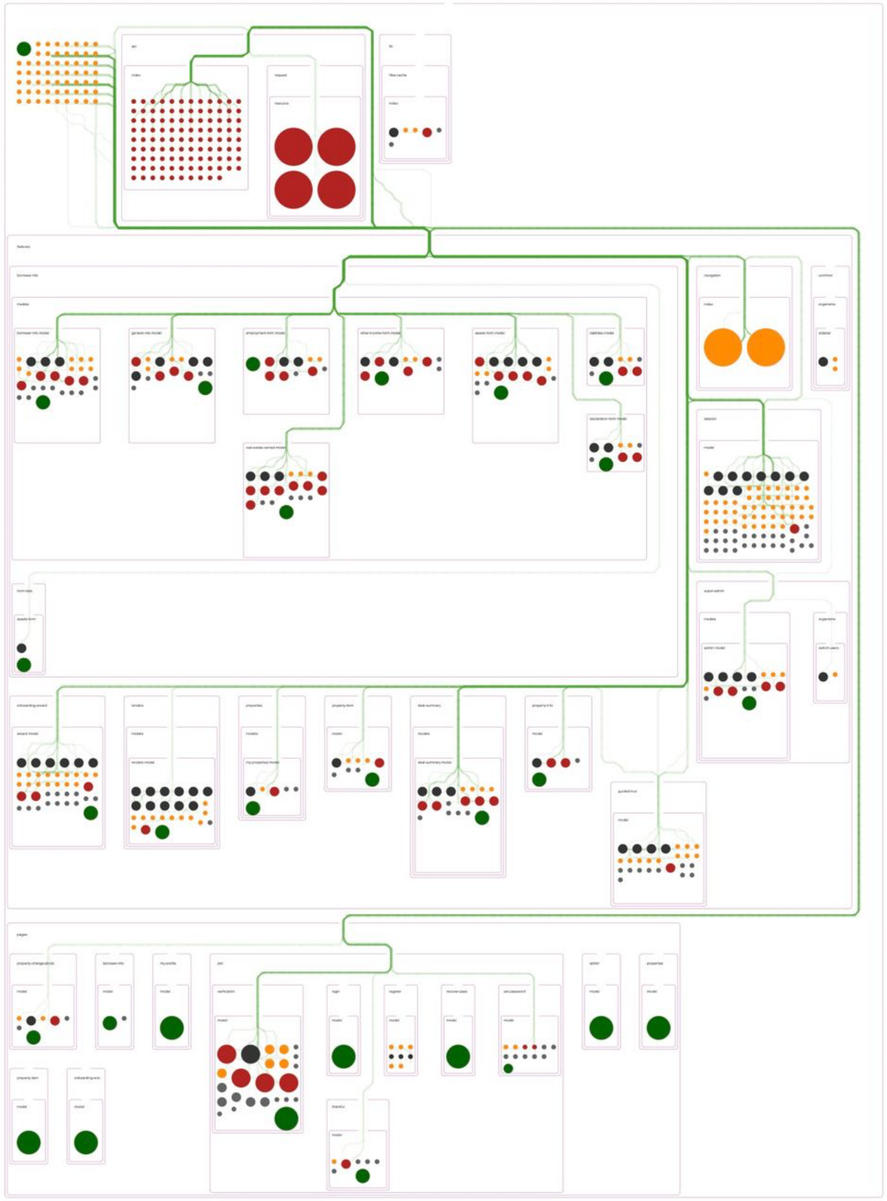
Визуализация связей с конкретным юнитом

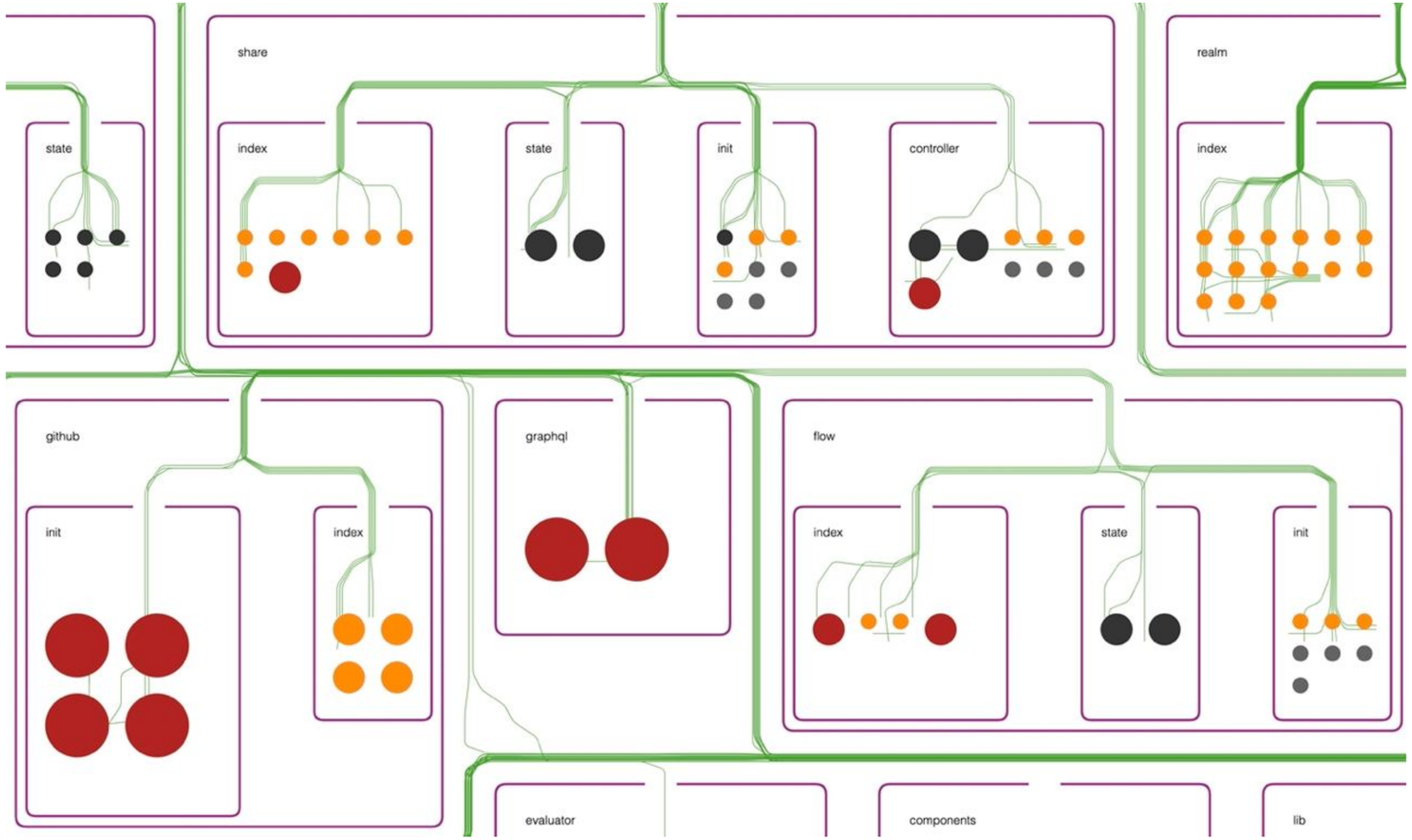
Пробую разные варианты











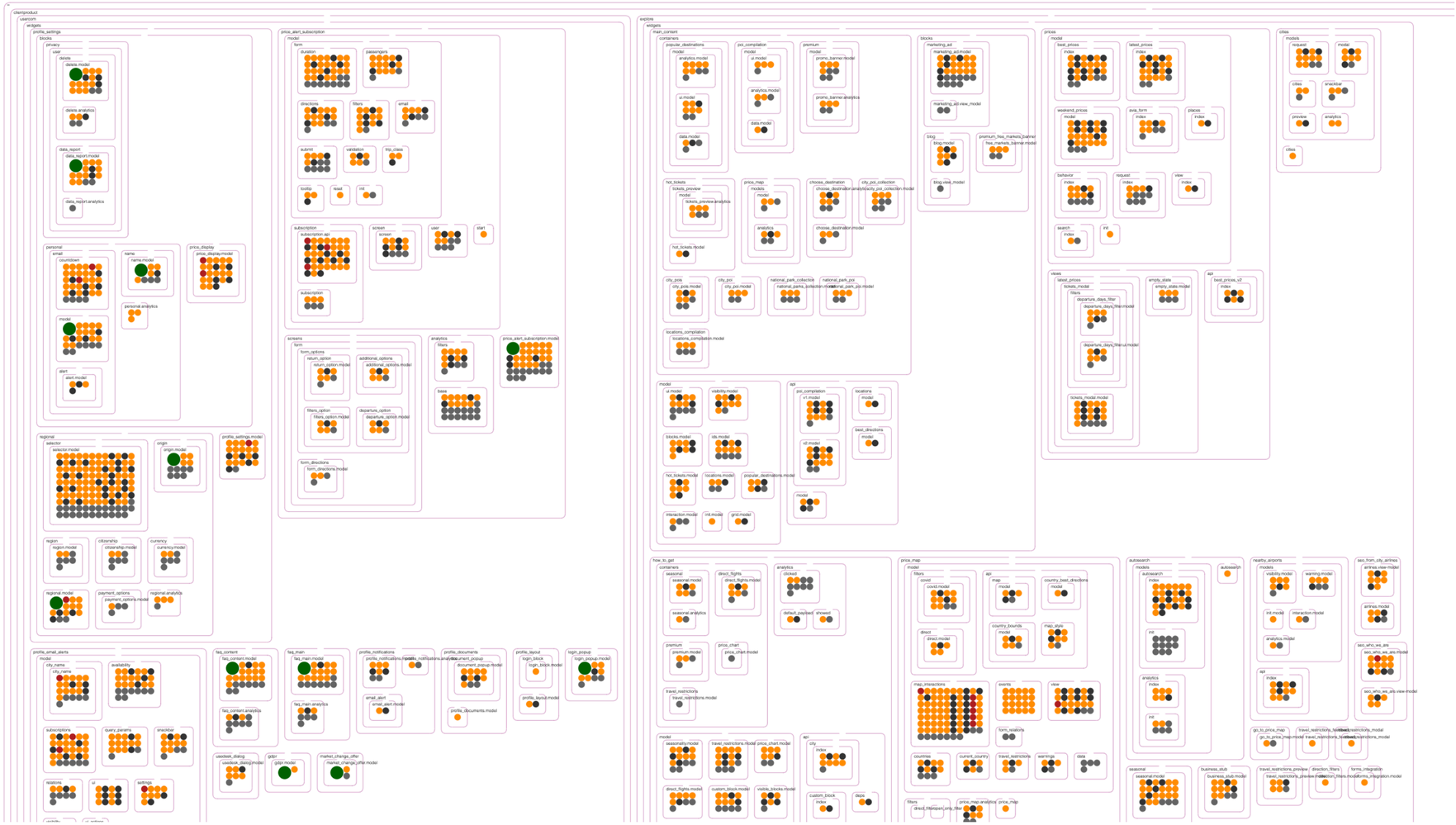
Общий вид приложения

Приложение aviasales

1 000 000 строк кода

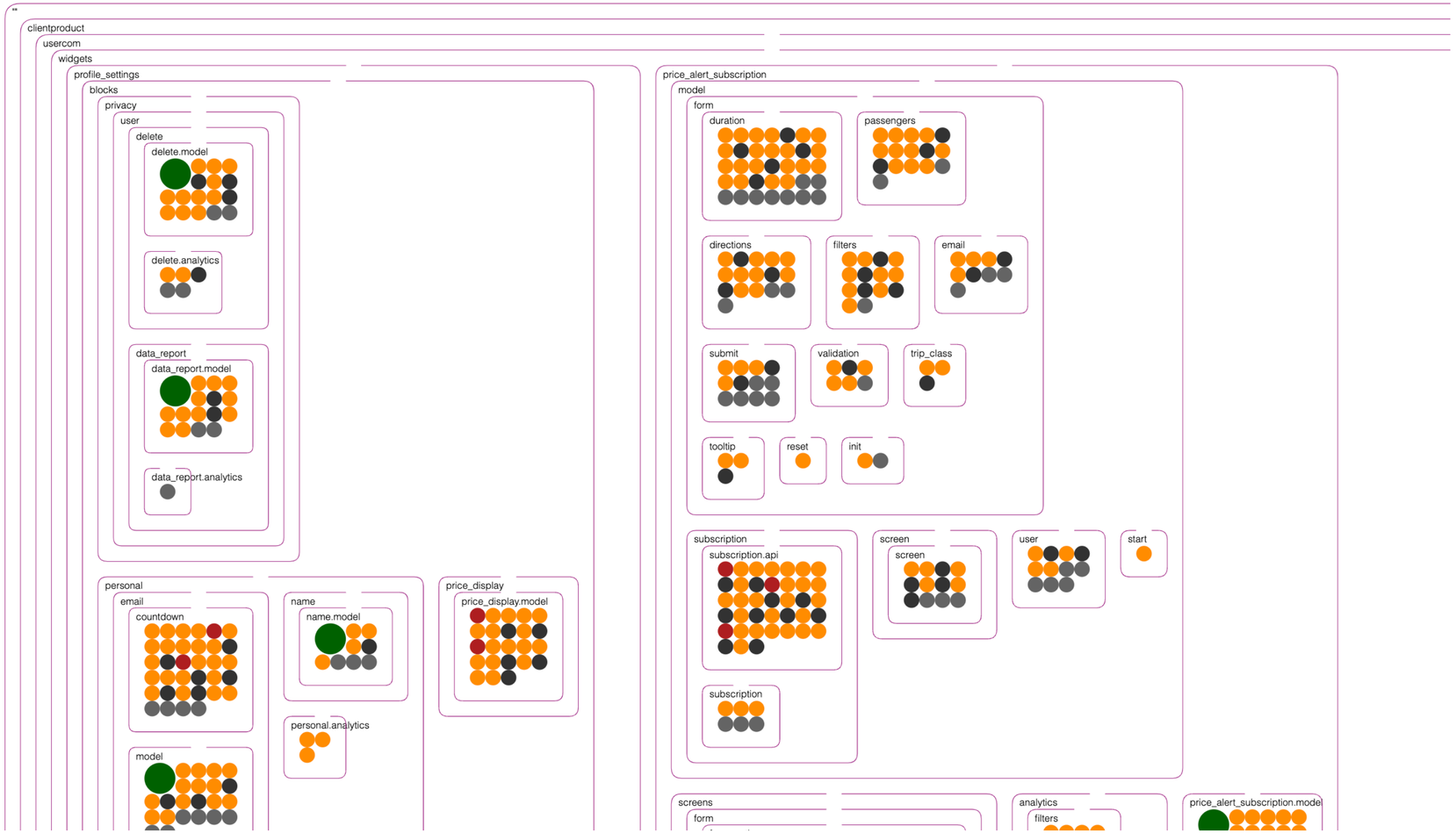
Приложение aviasales

1 000 000 строк кода



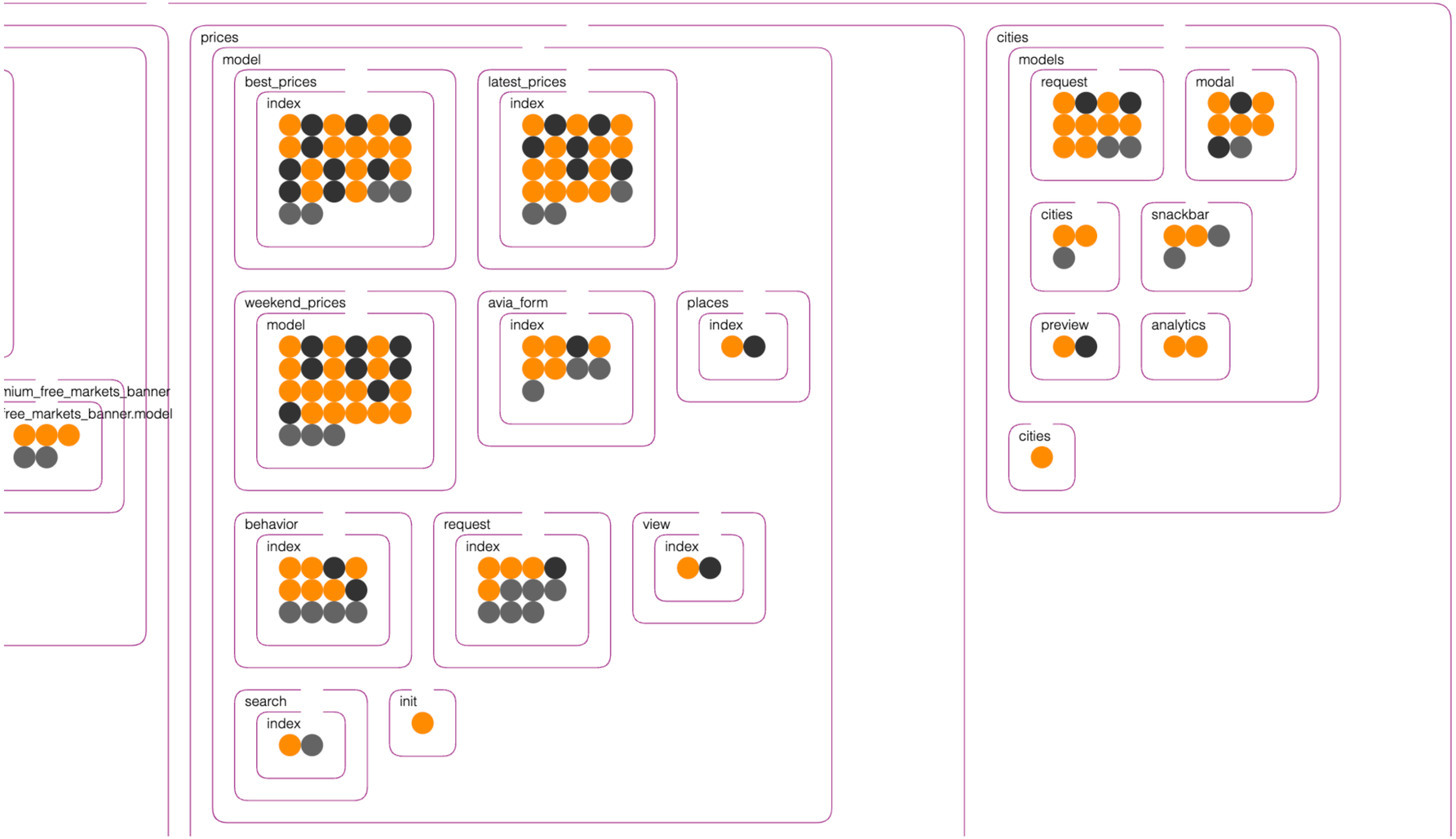
Приложение aviasales

1 000 000 строк кода



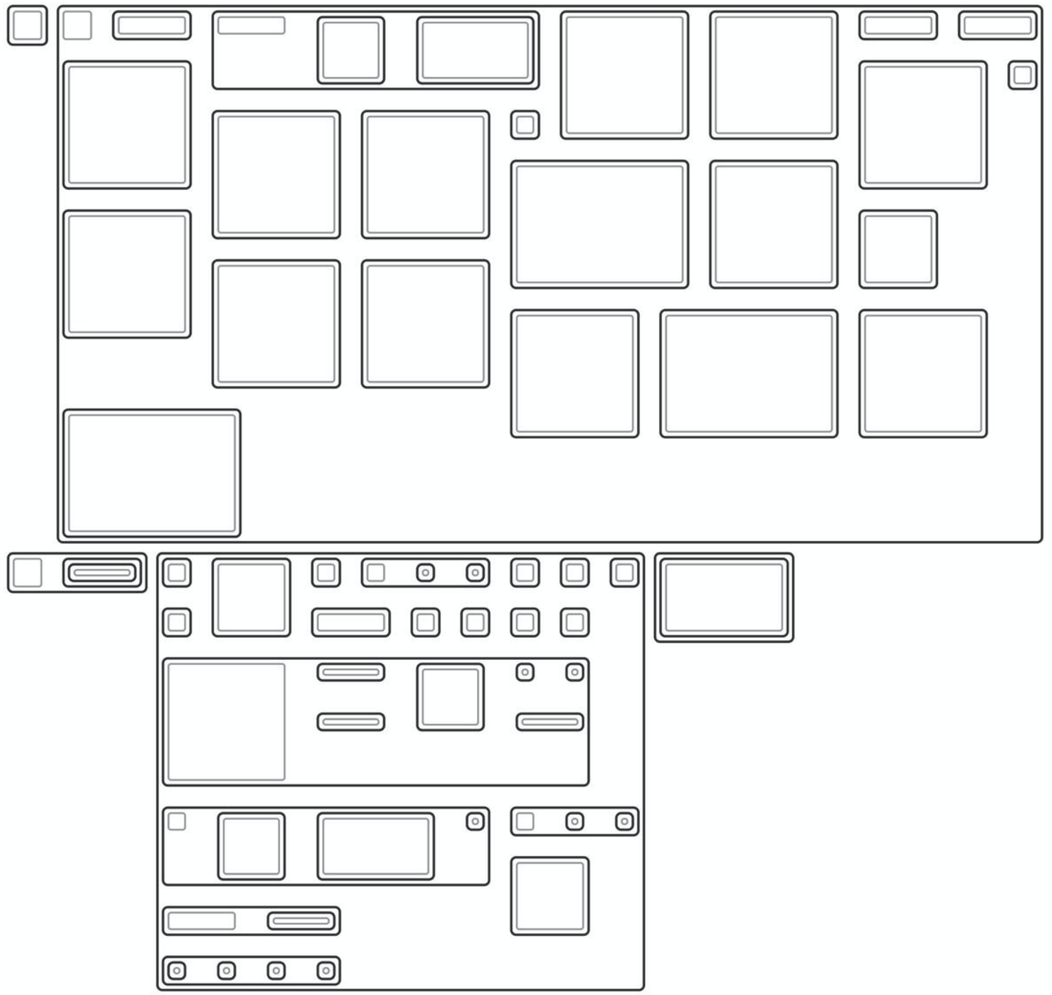
Приложение aviasales

1 000 000 строк кода



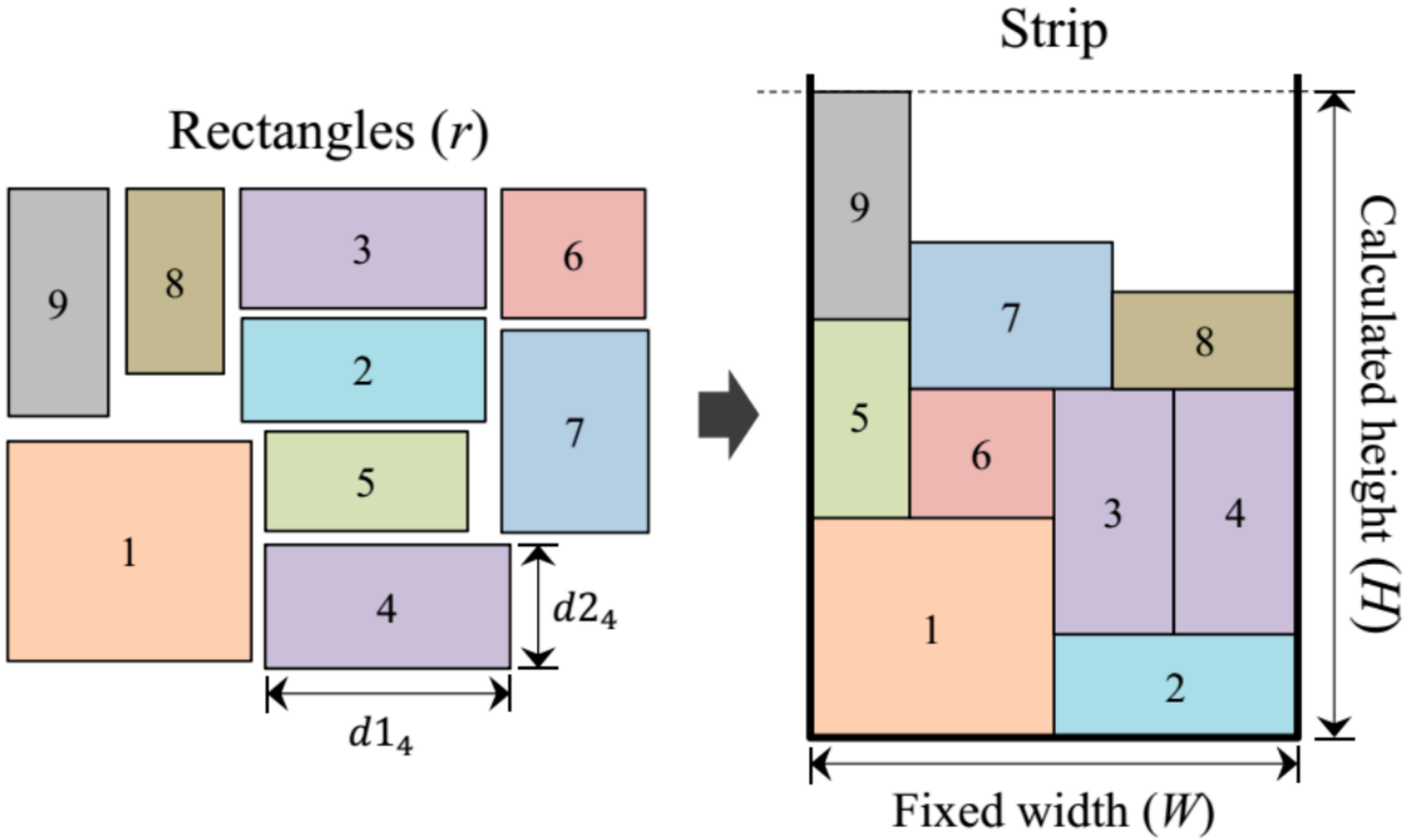
Проблемы визуализаций

Расположение блоков



Расположение блоков: алгоритм bin packing

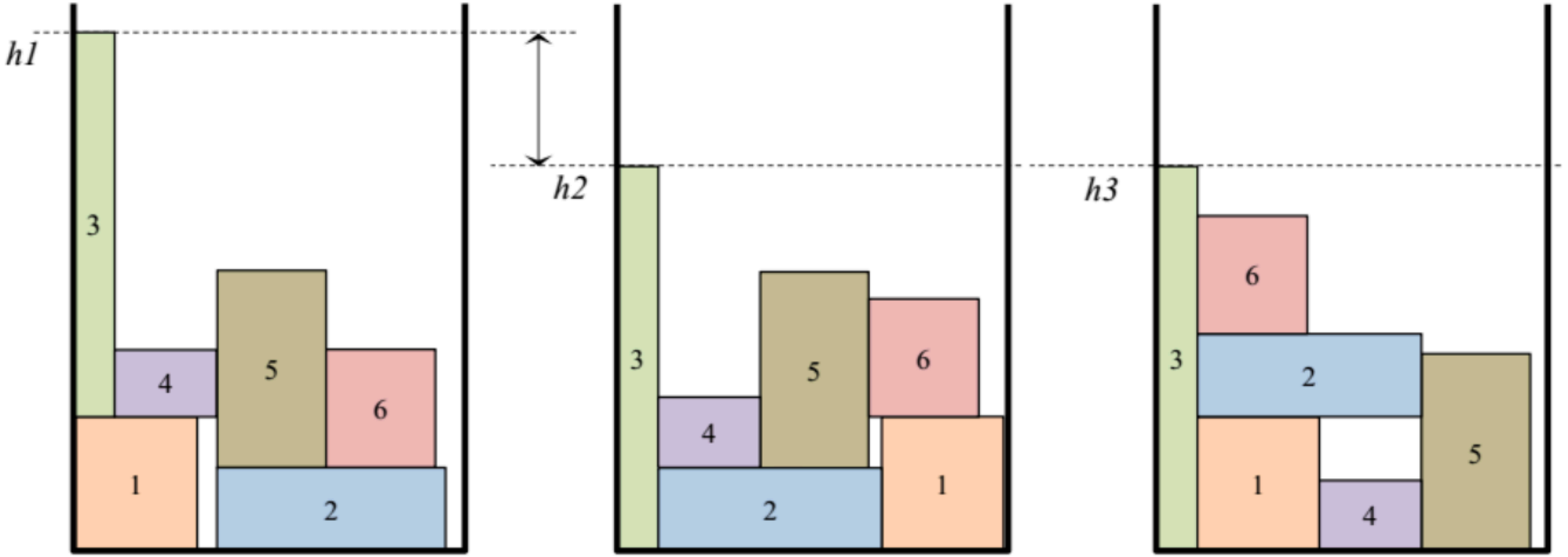
Он NP-полный, перебор не сработает



Two-Dimensional Rectangular Strip Packing Problem, Alvaro Luiz Neuenfeldt Junior, 2017

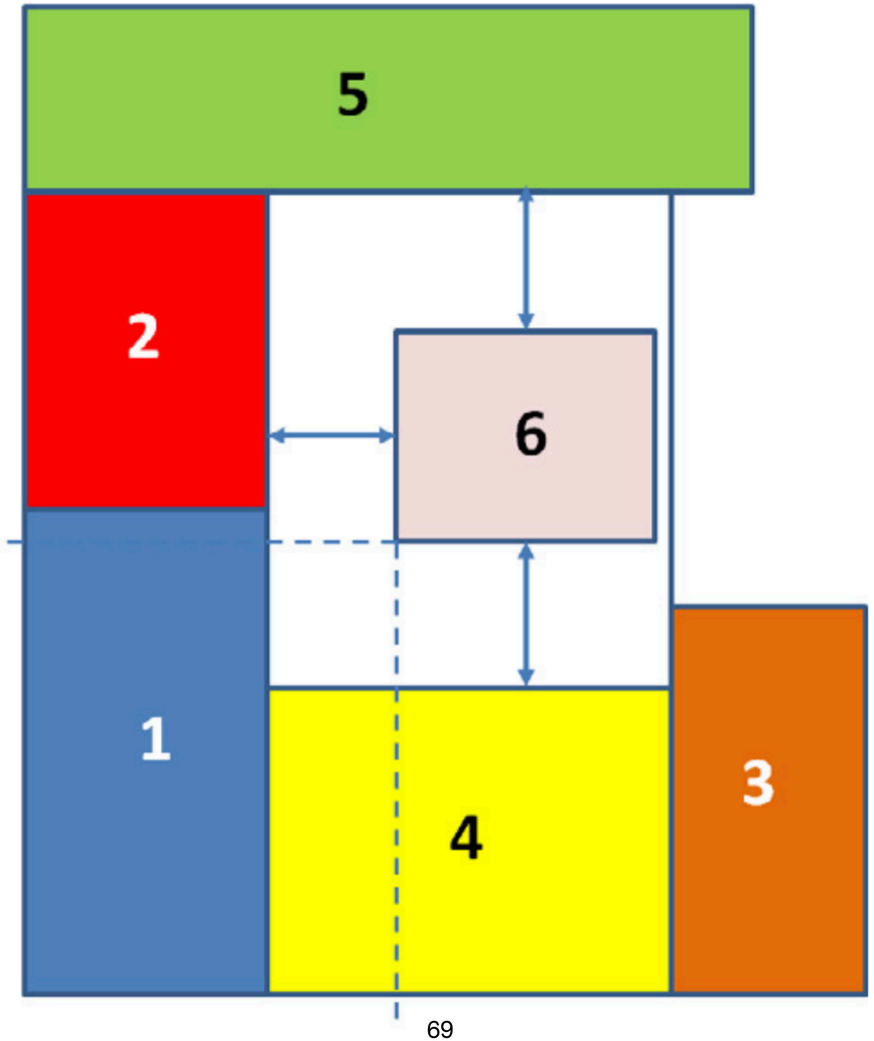
Расположение блоков: алгоритм bin packing

Идеального расположения не существует



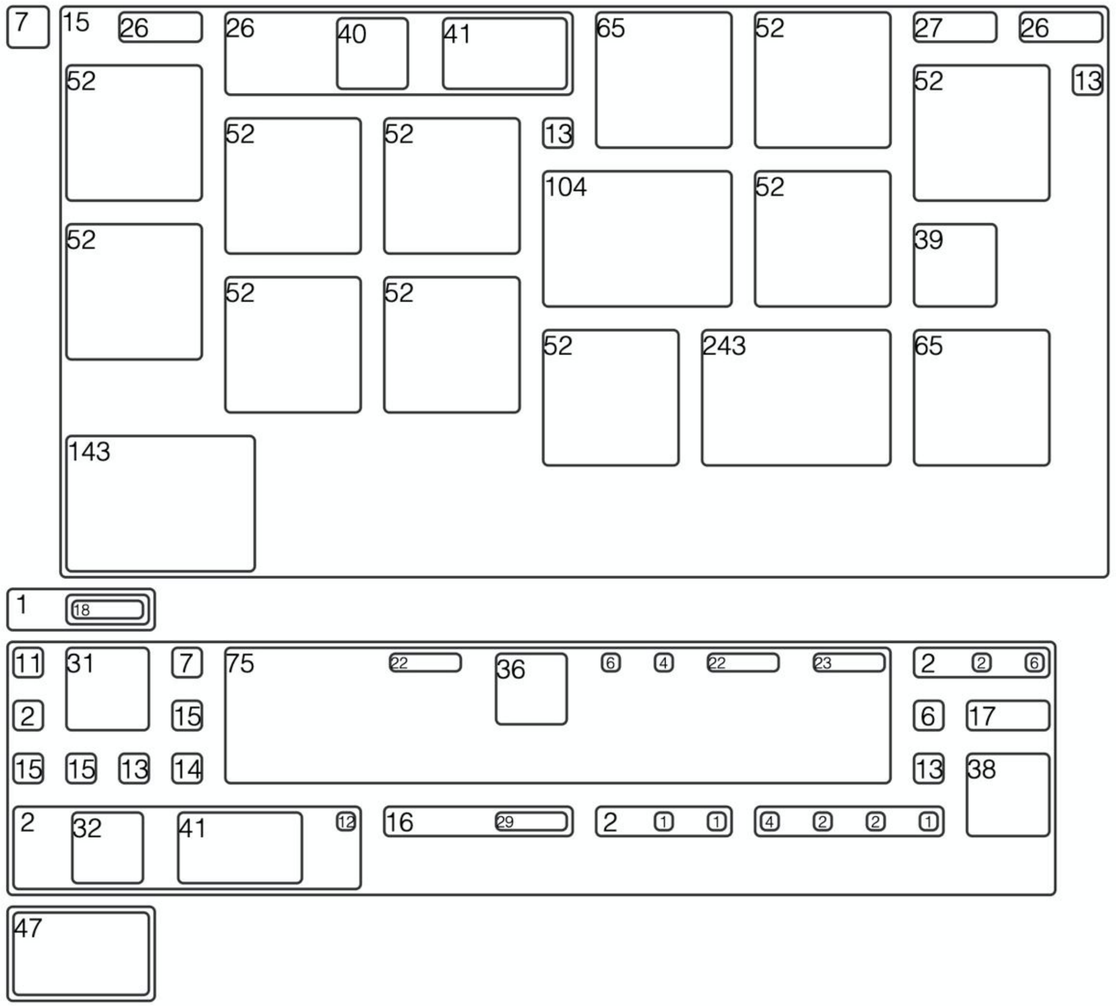
Расположение блоков: алгоритм bin packing

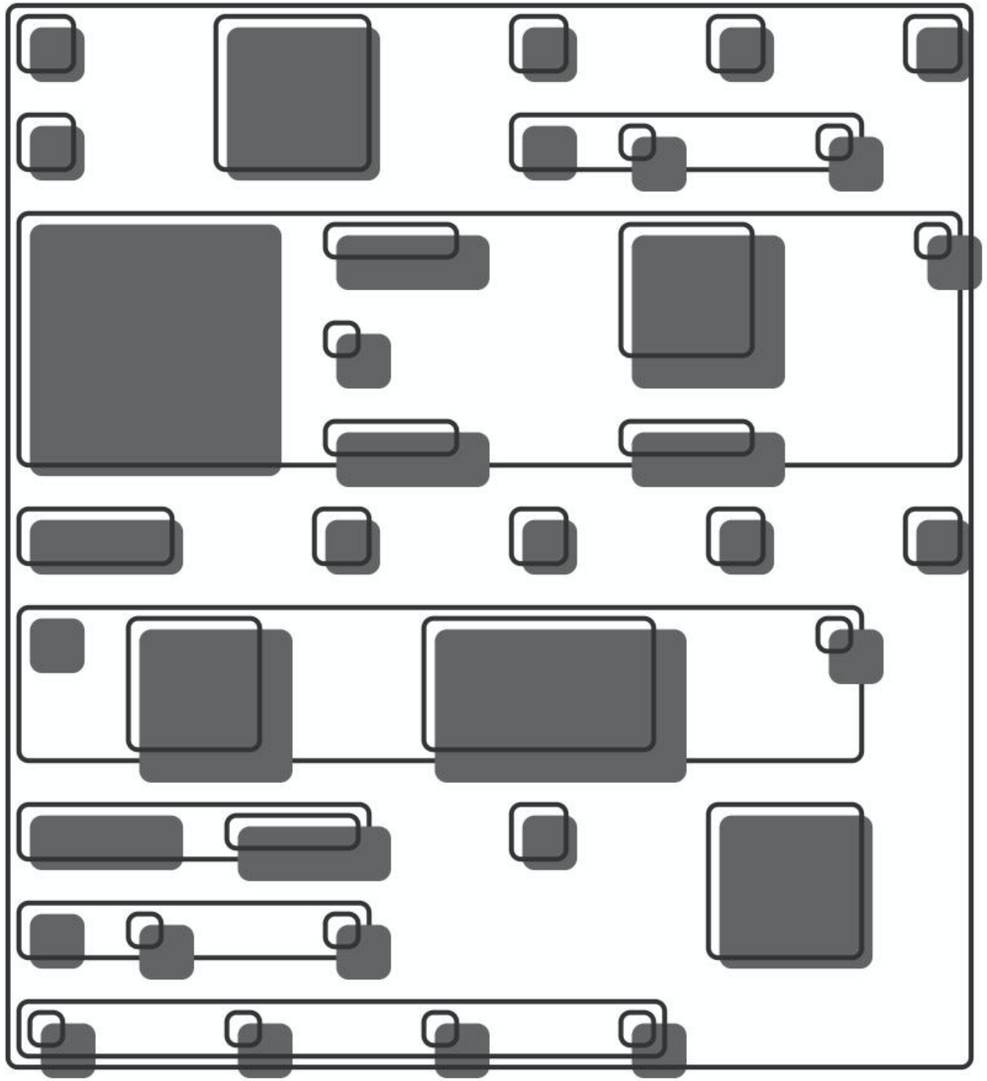
Разворачивание свернутого блока не должно менять лейаут



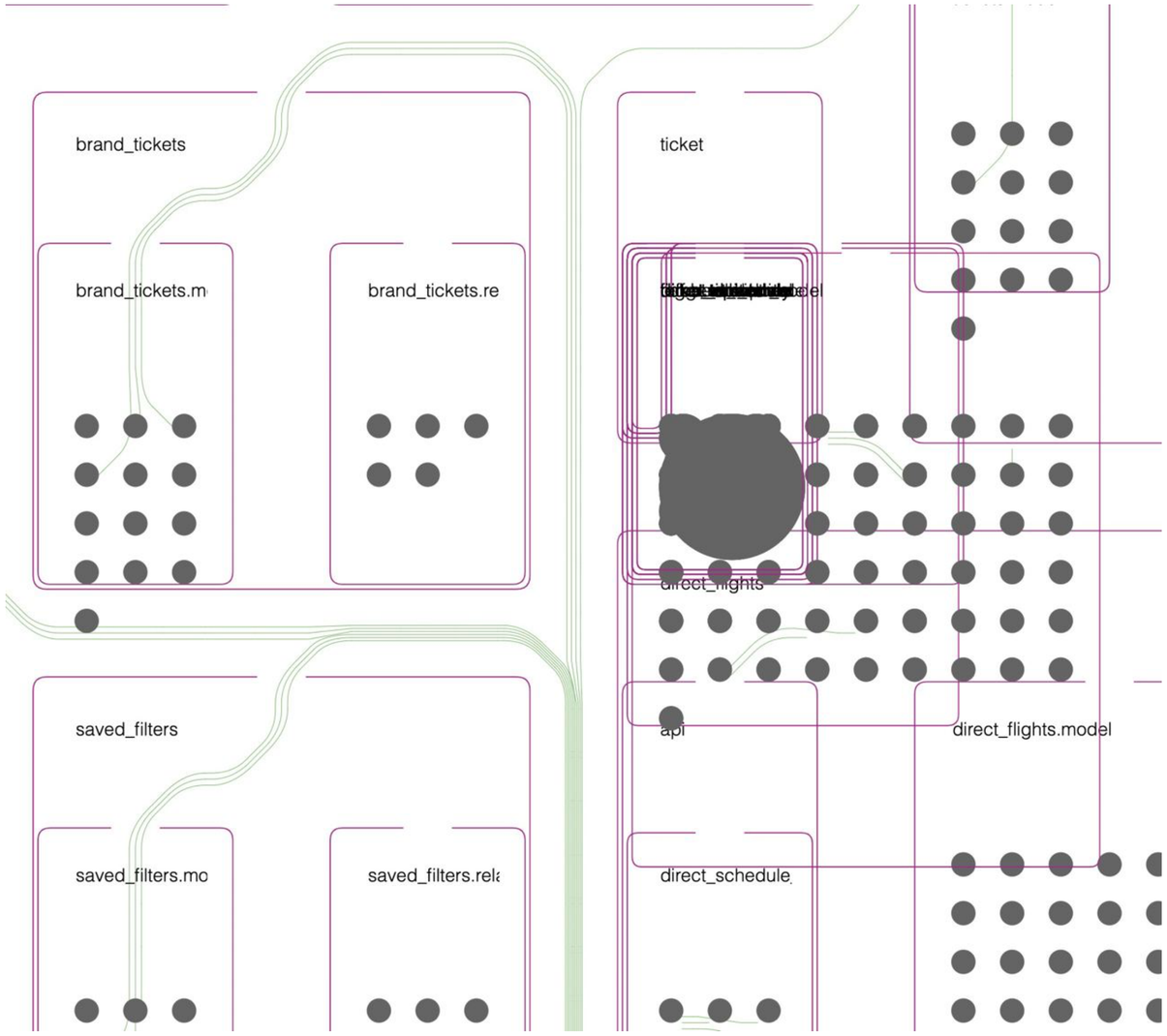
Расположение блоков: алгоритм bin packing

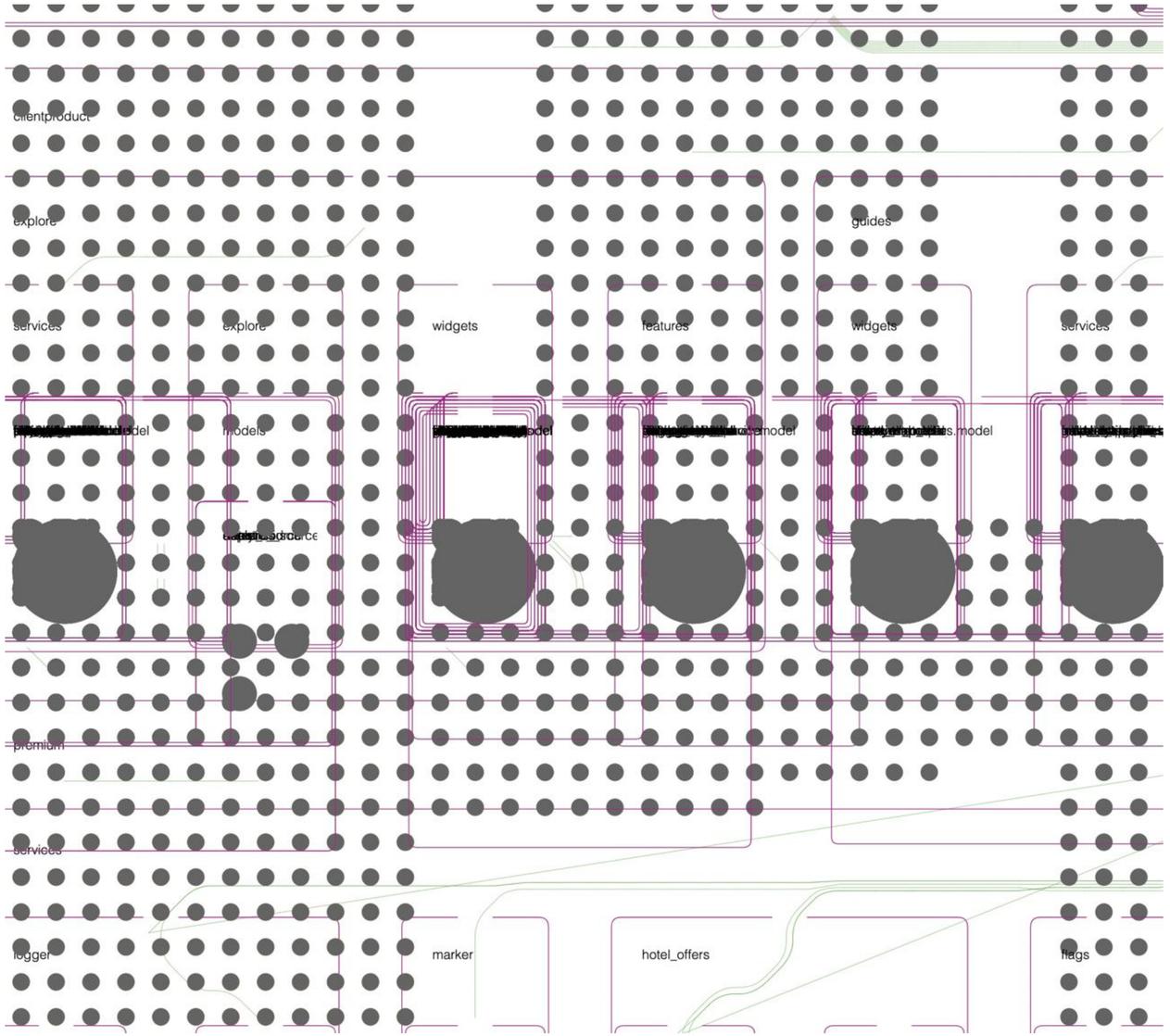
Поддержка иерархичности





Визуальные баги





Визуальные баги

Расположение блоков: итог

Для работы помогает алгоритм, который работает в дискретном пространстве: **bin packing**

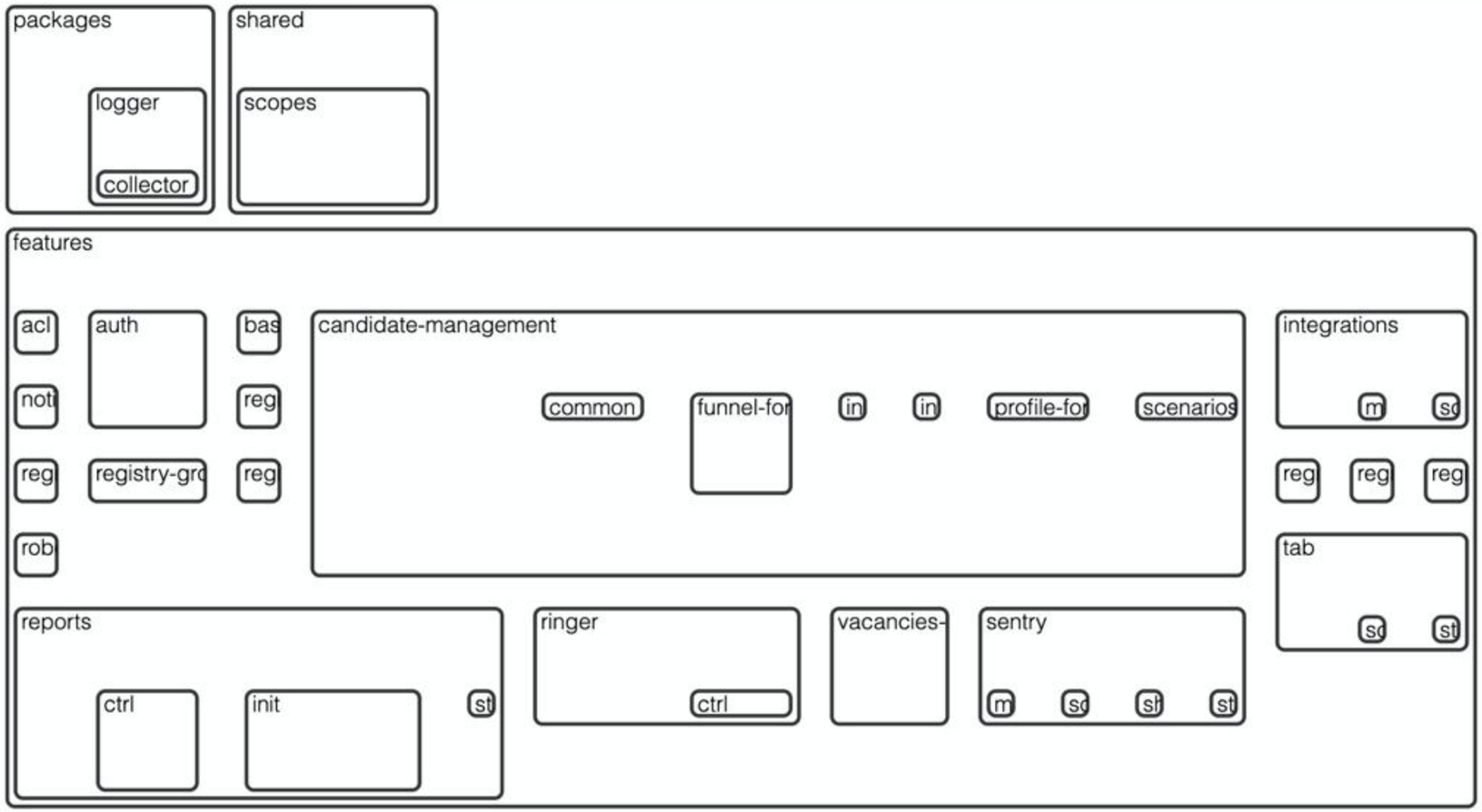
При неаккуратном написании имеет квадратную сложность

К счастью, в js есть качественная и эффективная реализация: potpack от mapbox *(Владимир Агафонкин, ты очень крутой🙏)*

https://github.com/mapbox/potpack

Проблемы визуализаций

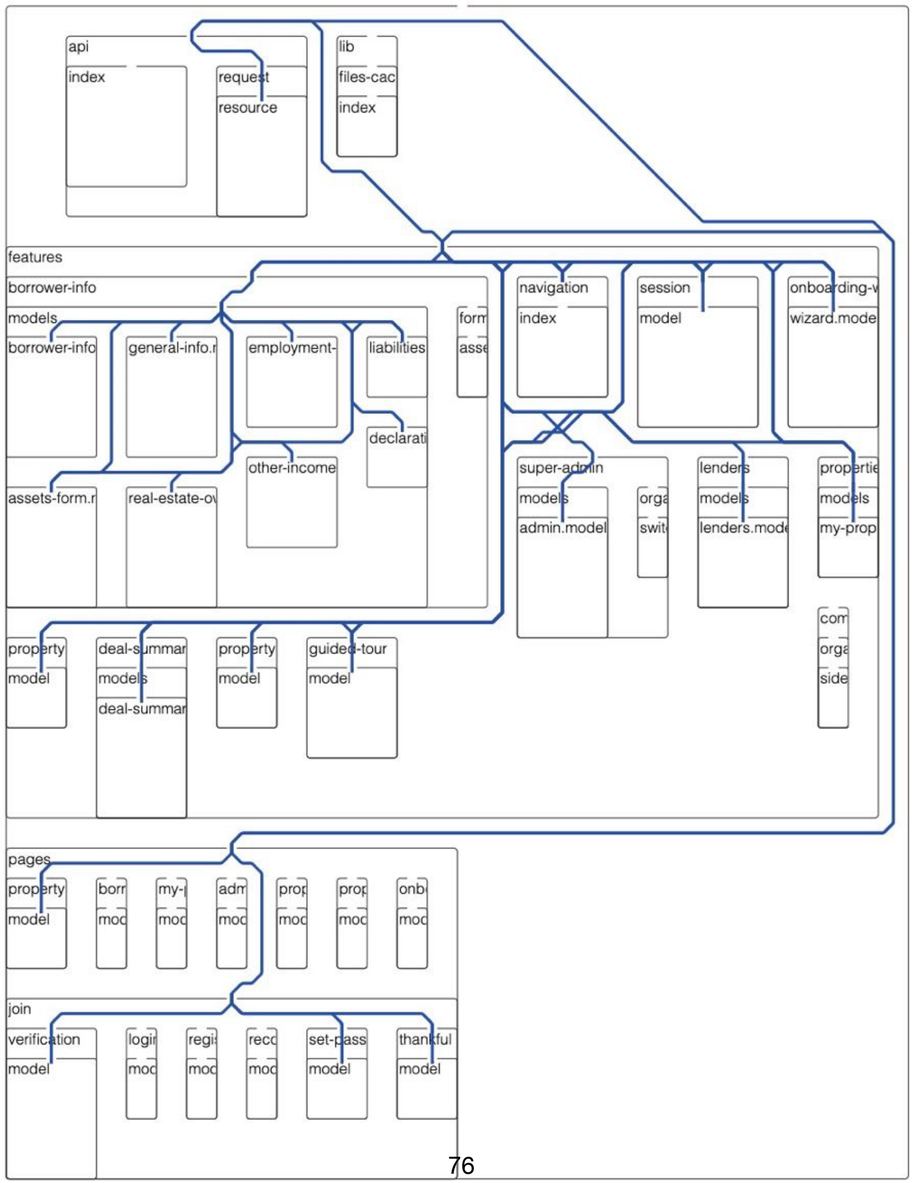
Расположение текста



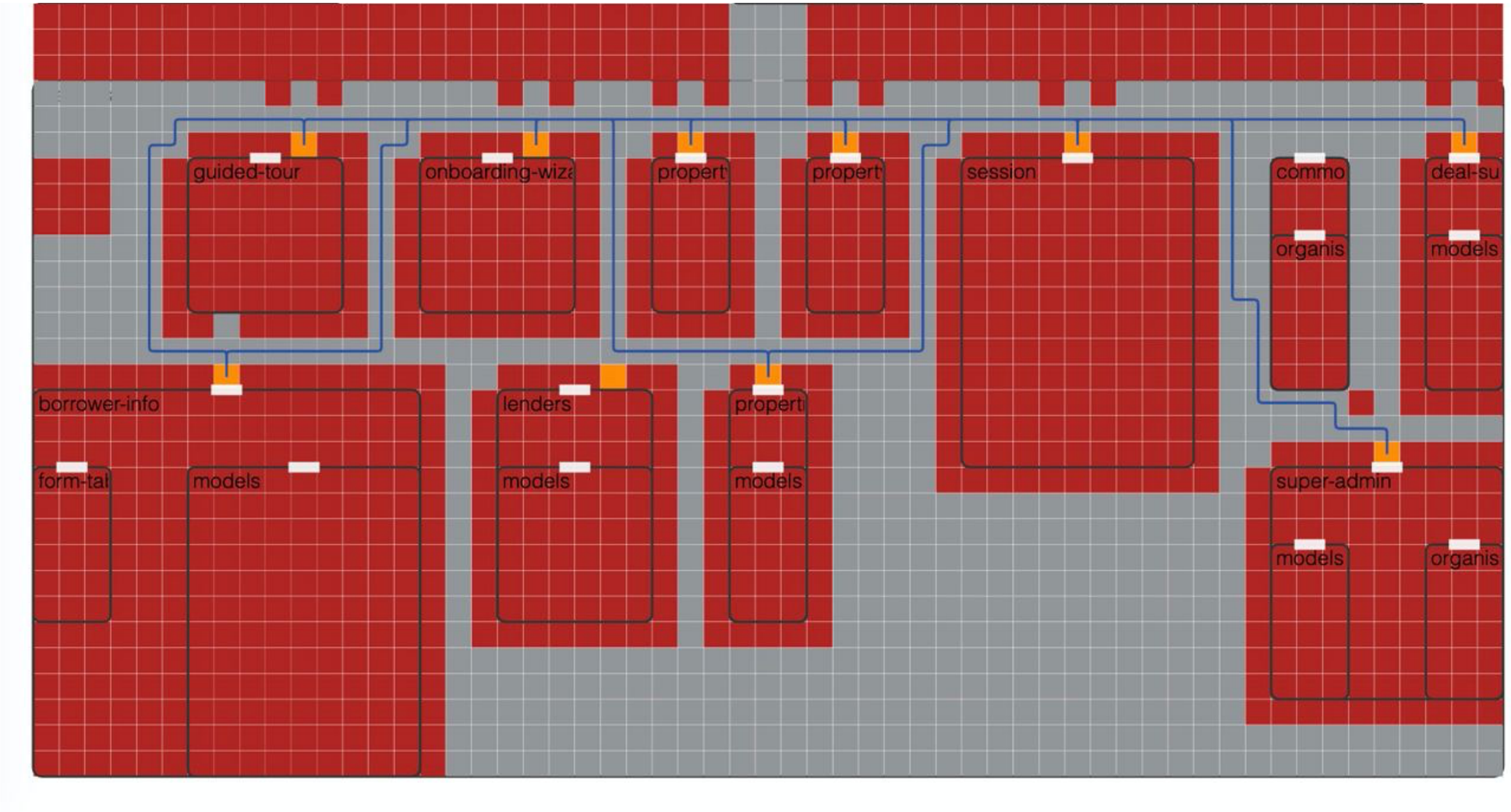
Проблемы визуализаций

Прокладывание путей через блоки

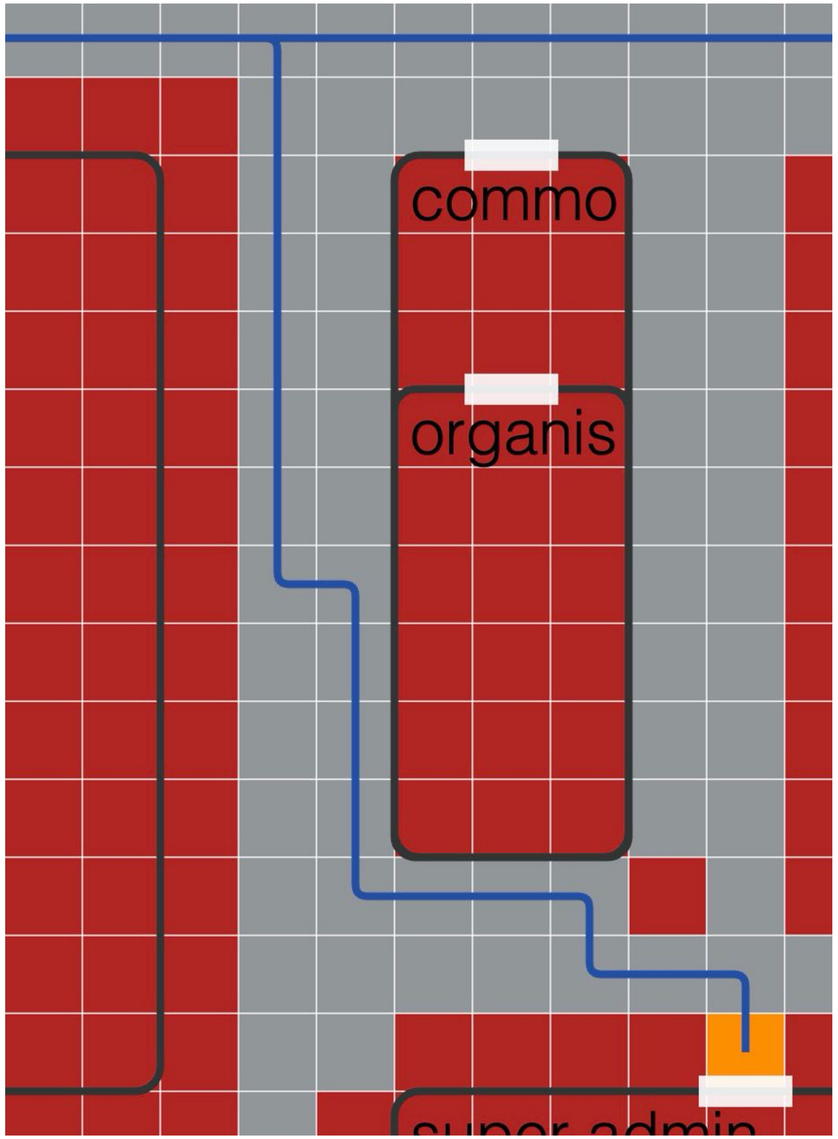
Прокладывание путей через блоки



Алгоритм A*


guided-tour
onboarding-wiza
propert
propert
session
commo
organis
deal-su
models
borrower-info
form-tal
models
lenders
models
propert
models
super-admin
models
organis

Алгоритм A*



Прокладывание путей: итог

Работает в дискретных (целочисленных) координатах

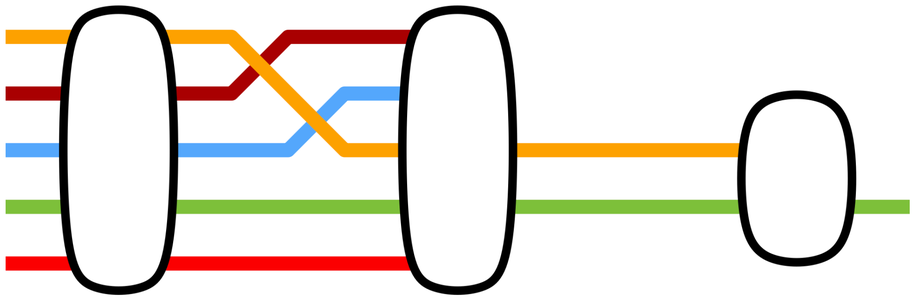
Для работы помогает алгоритм поиска кратчайшего пути: **A***

Требует модификаций в алгоритме расположения блоков:
добавление отступов между блоками чтобы было место для линий

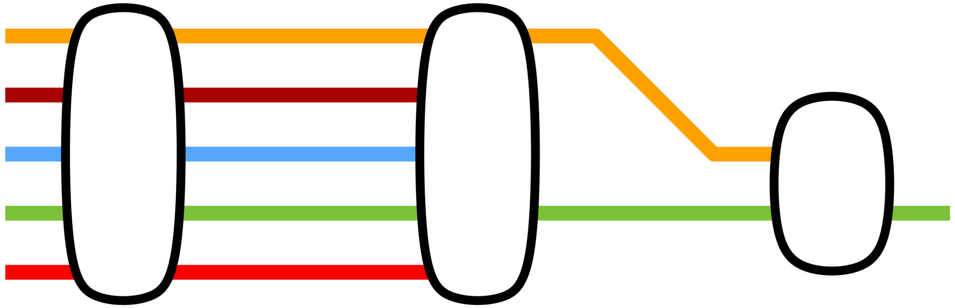
Проблемы визуализаций

Непересекающиеся линии

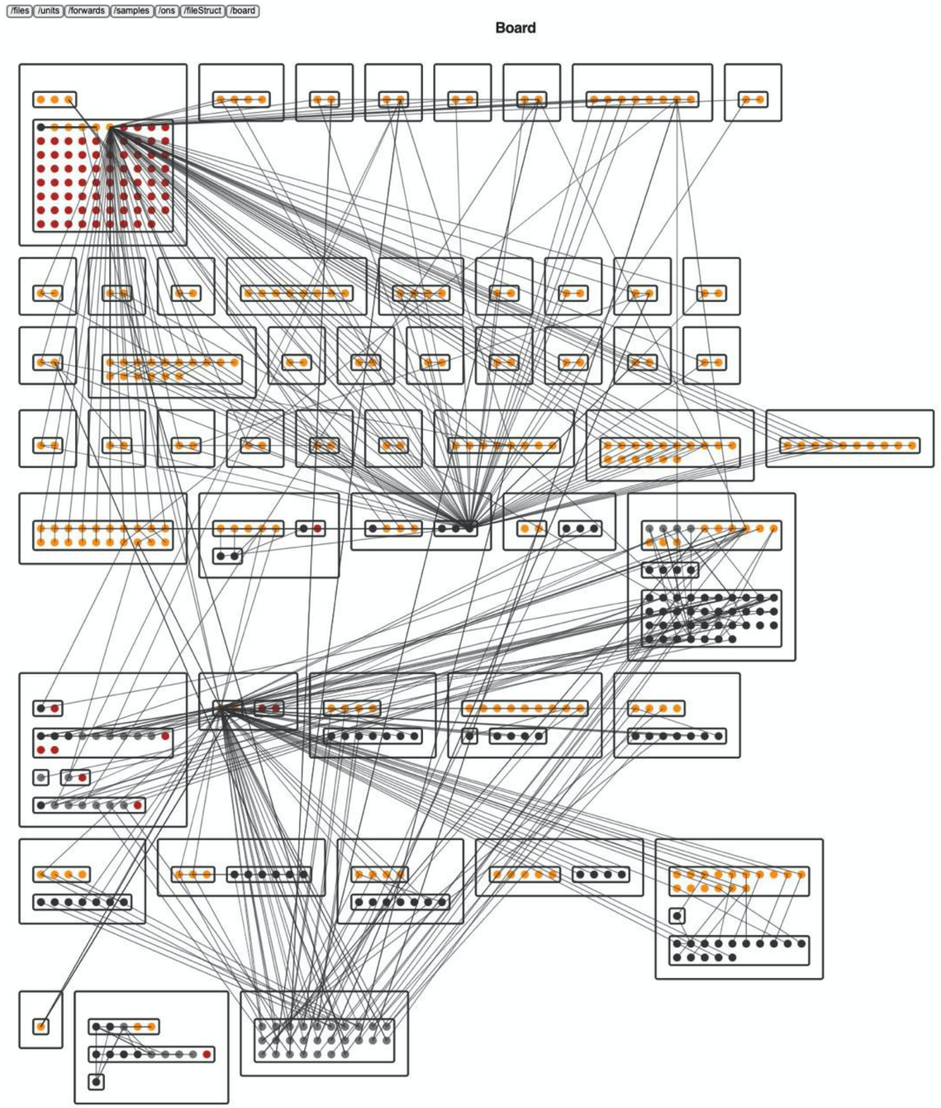
Чем меньше пересечений - тем проще читать схему



Есть пересечения



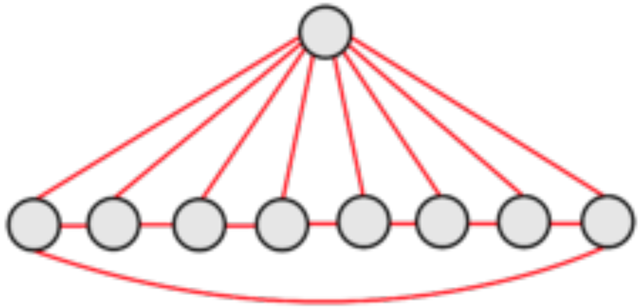
Нет пересечений

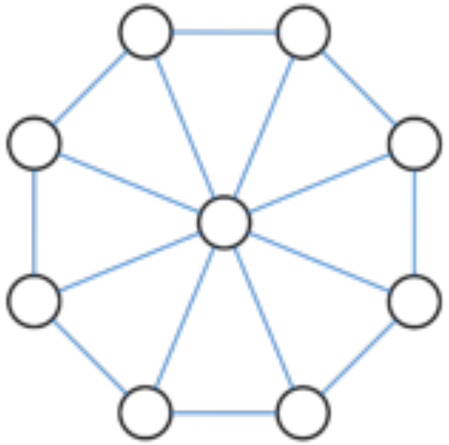


Вариант без роутинга связей

Планарный граф

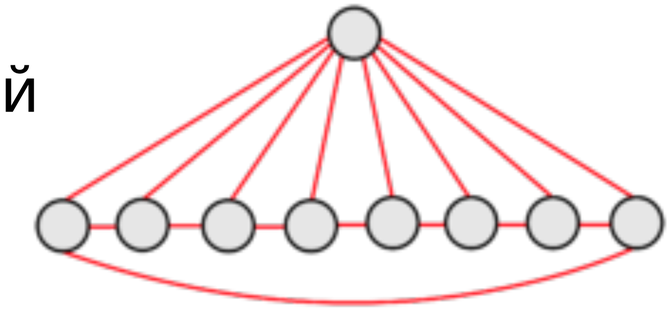
Можно изобразить без пересечений

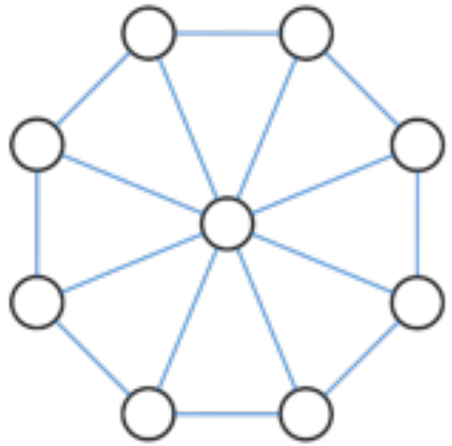




Планарный граф

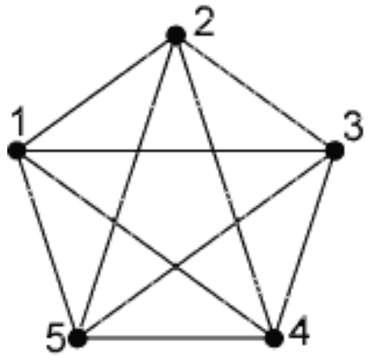
Можно изобразить без пересечений



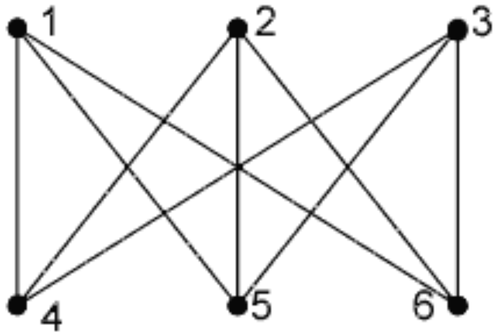


Непланарный граф

Можно изобразить только с пересечениями

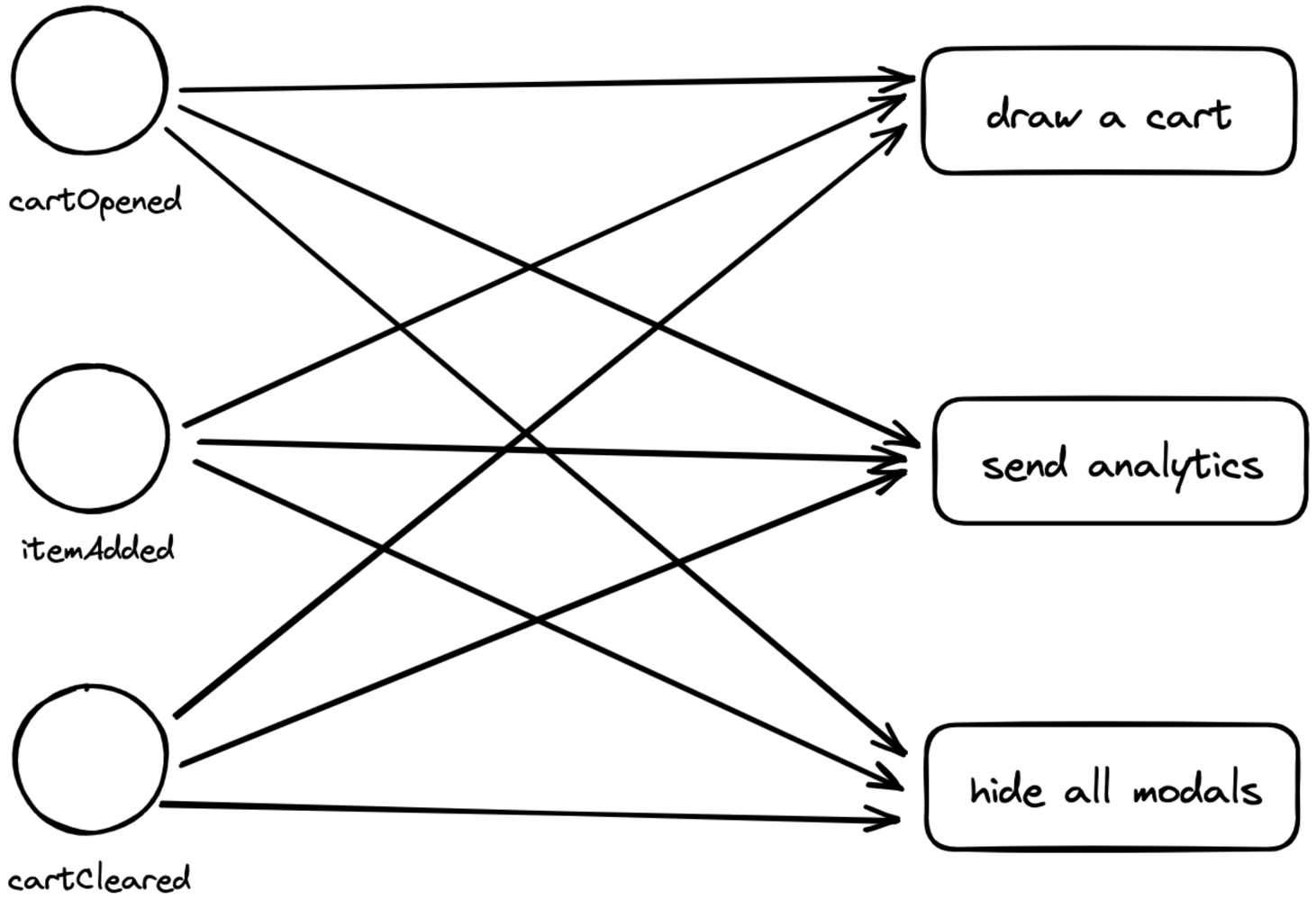


K_5



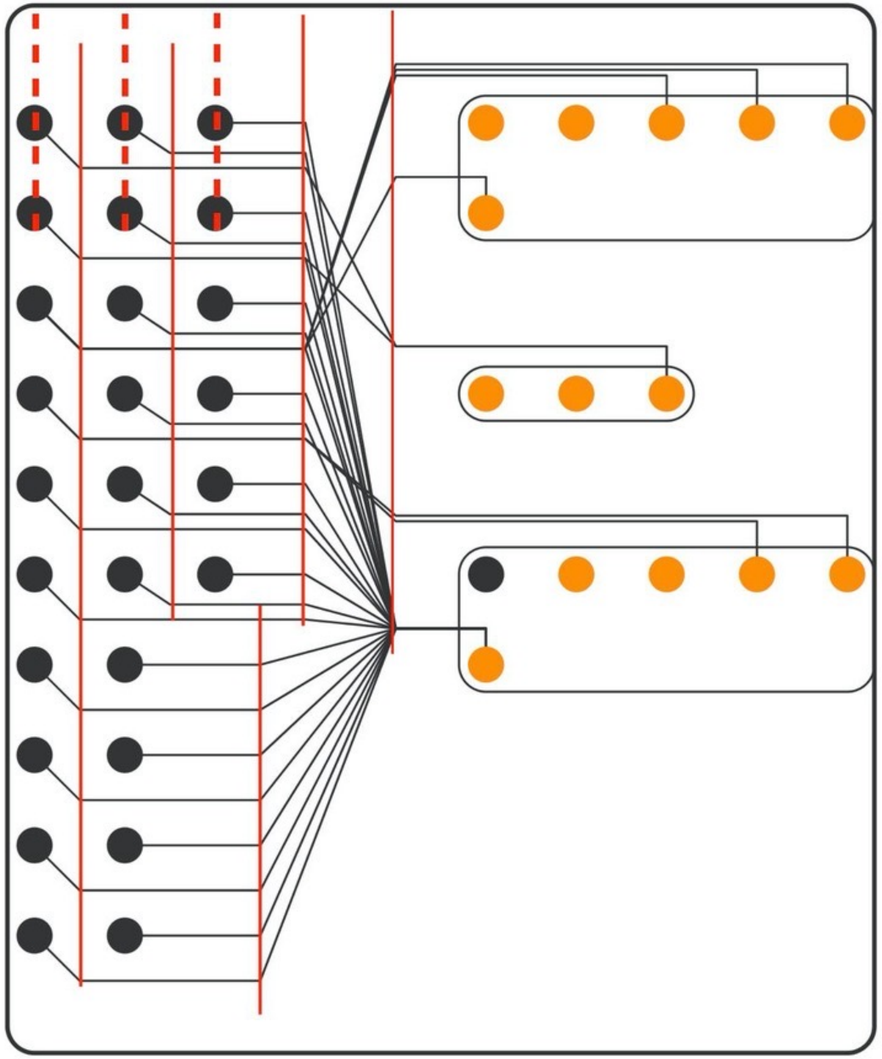
$K_{3,3}$

Finding an elegant linear time planarity testing algorithm is still an unsolved problem in computer science

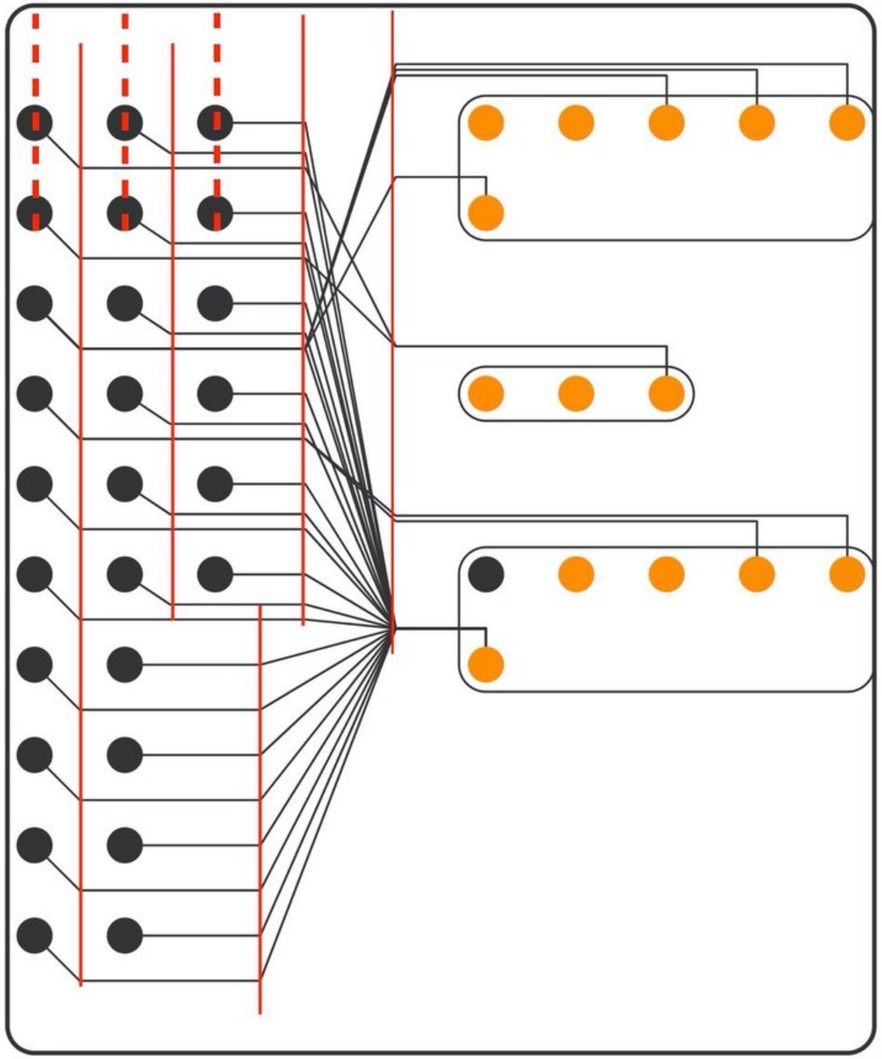


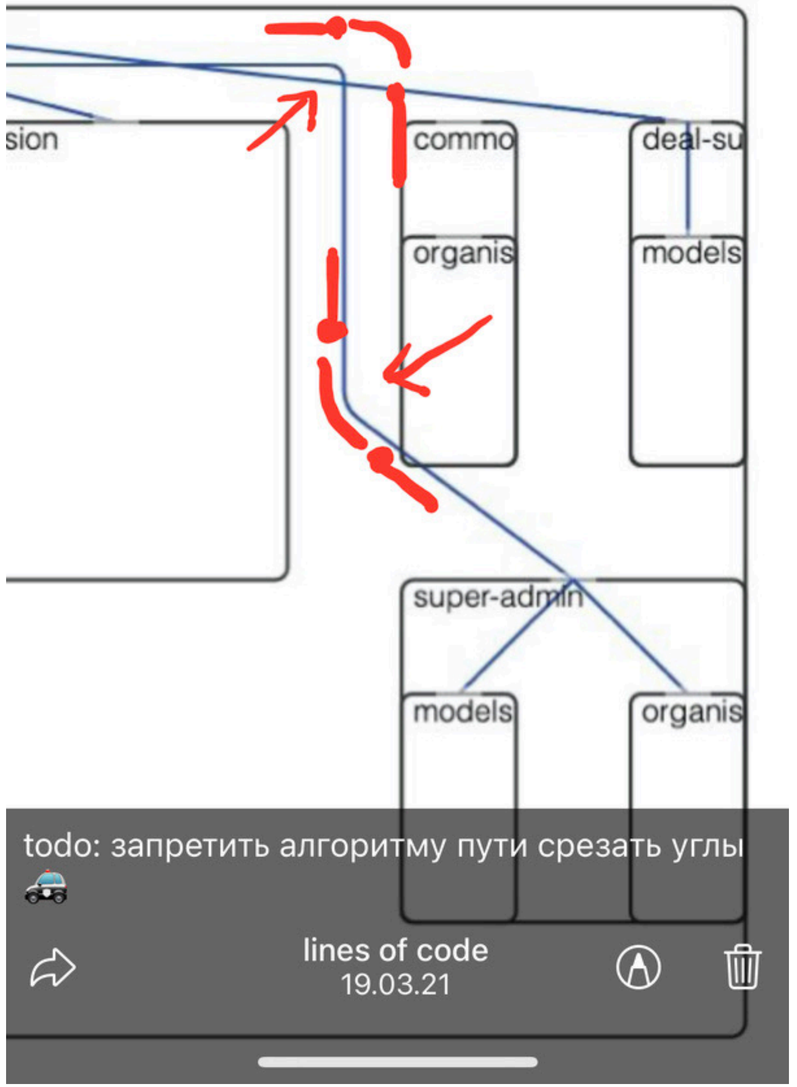
Минимальный непланарный граф на практике

Процесс: анализирую структуру линий



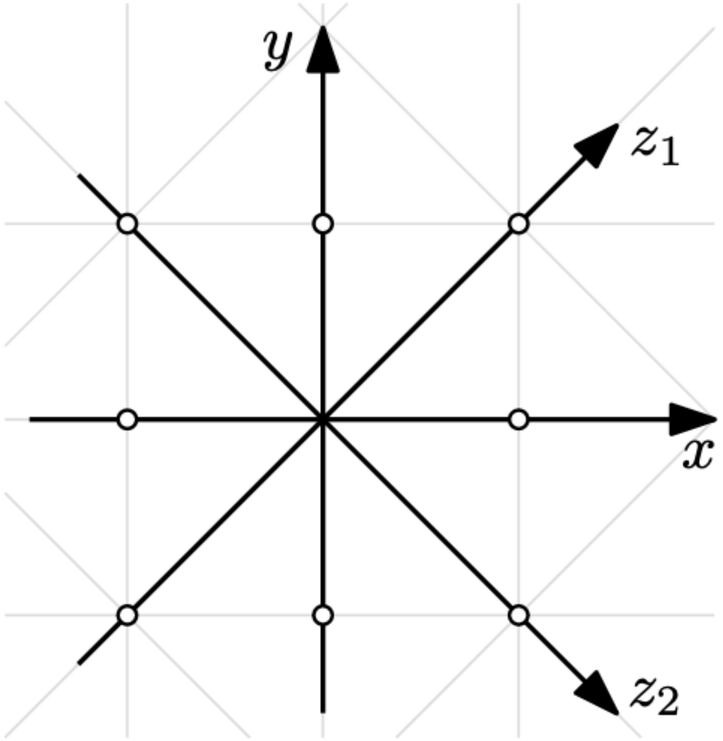
Процесс: анализирую структуру линий

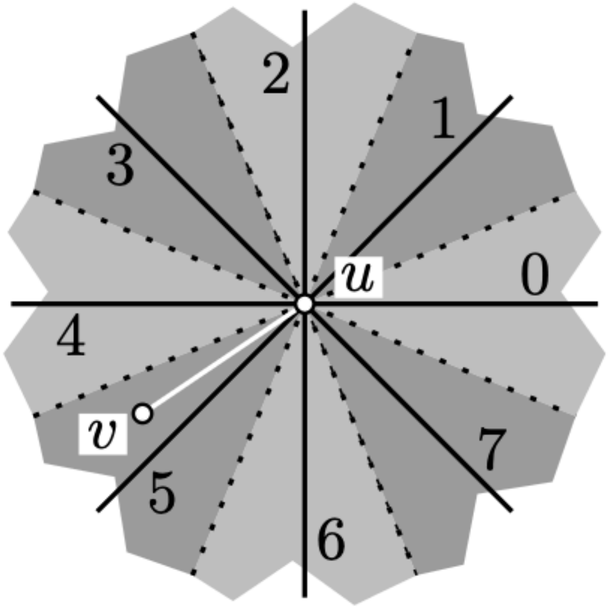




Получается не сразу

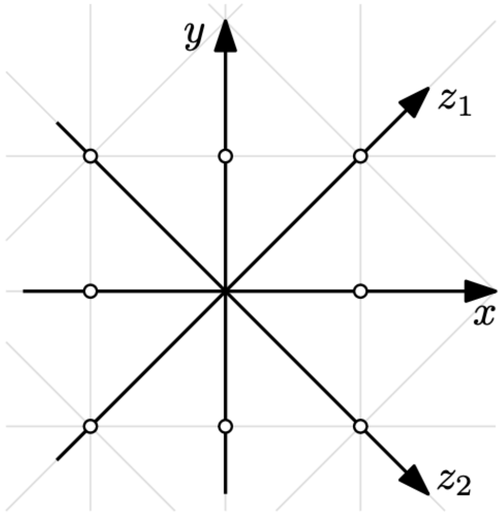
Математическая основа роутинга линий — Octilinear coordinate system

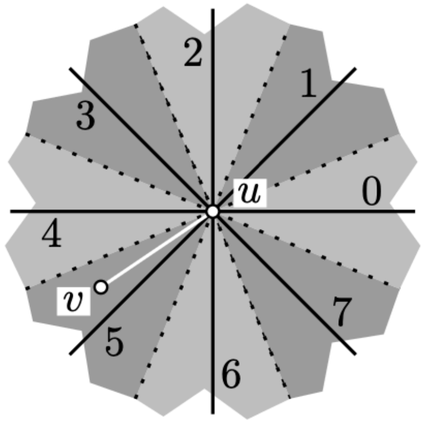


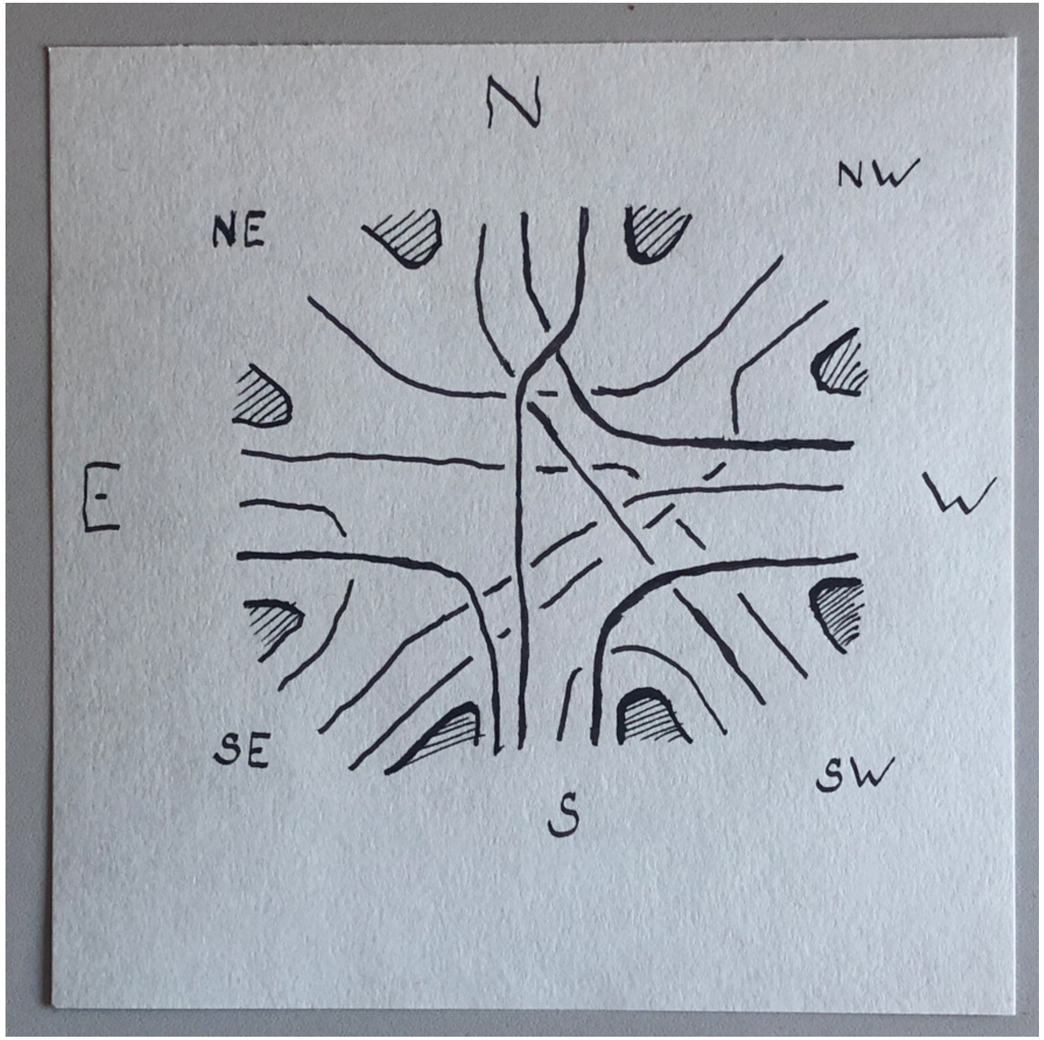


Octilinear graph drawing: Metro map layout, Jonathan Klawitter, 2020

Математическая основа роутинга линий — Octilinear coordinate system

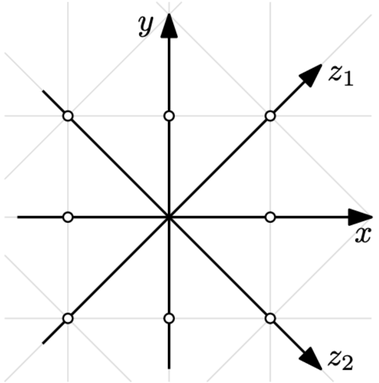


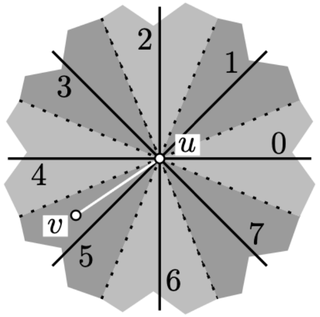


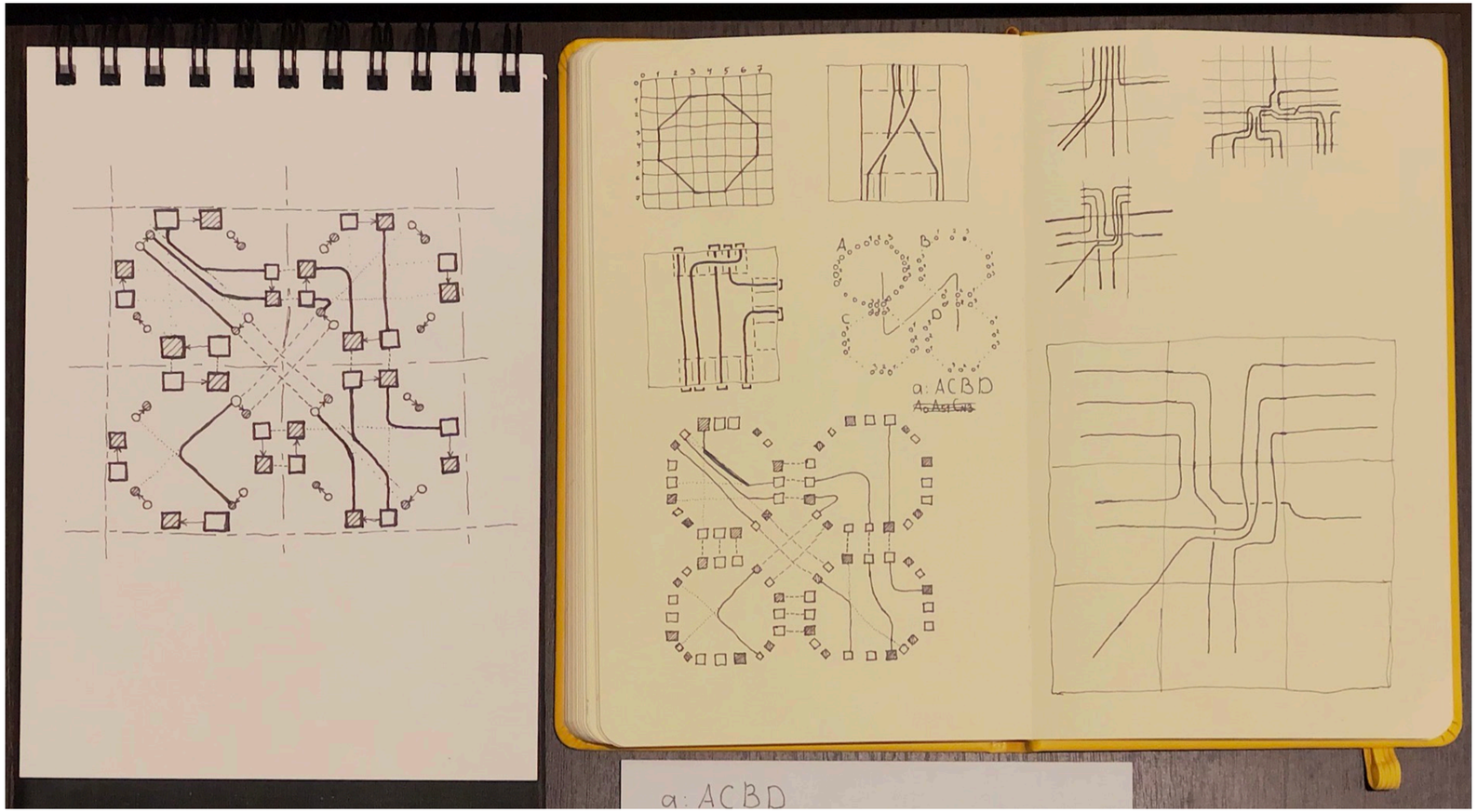


И попытки разобраться в ней

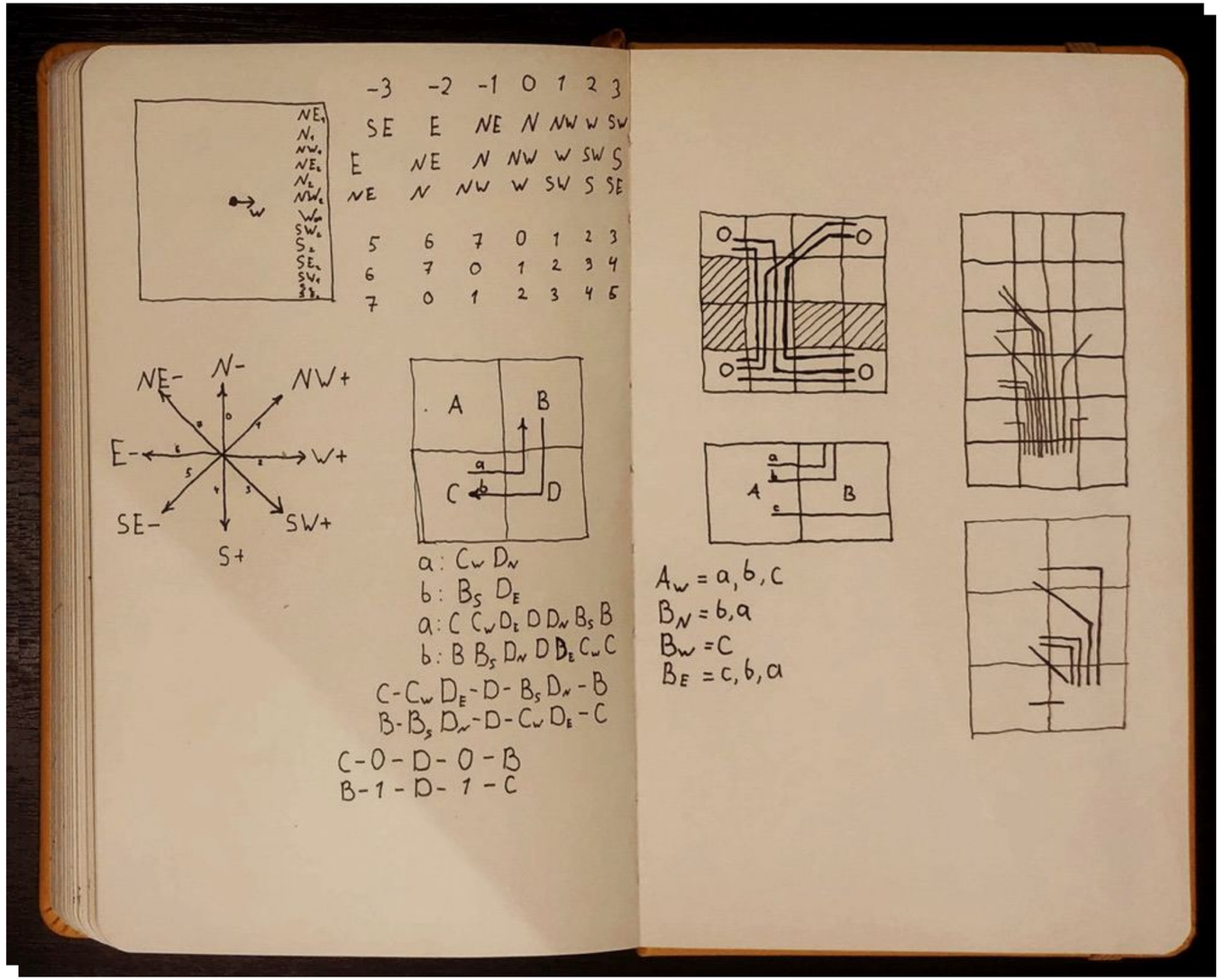
Математическая основа роутинга линий — Octilinear coordinate system



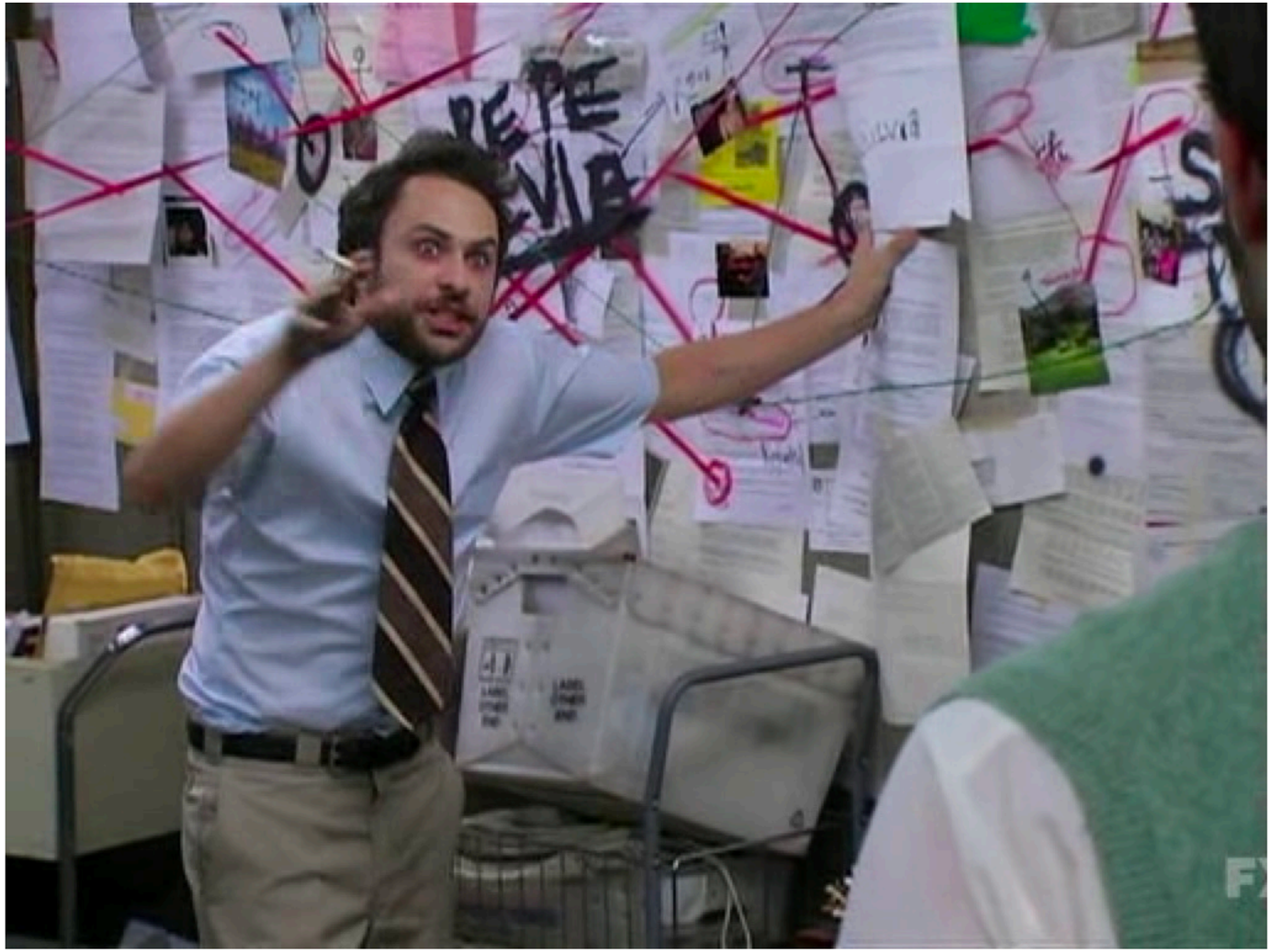


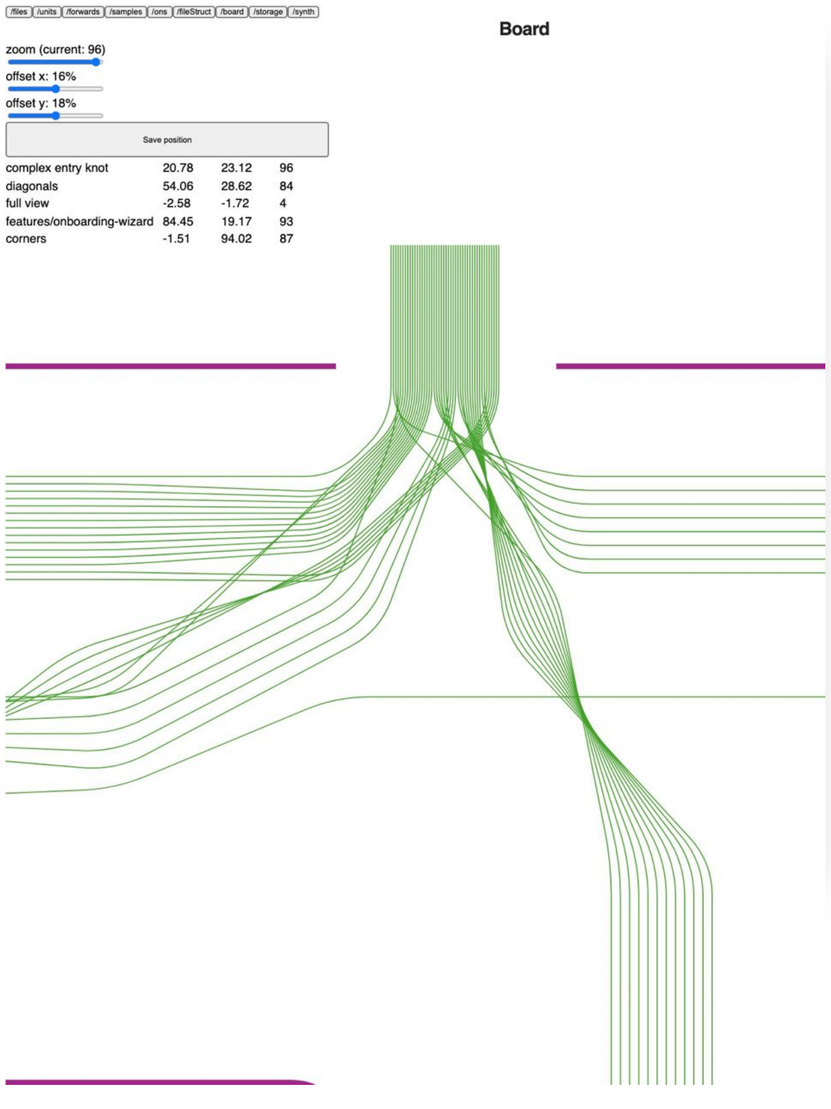


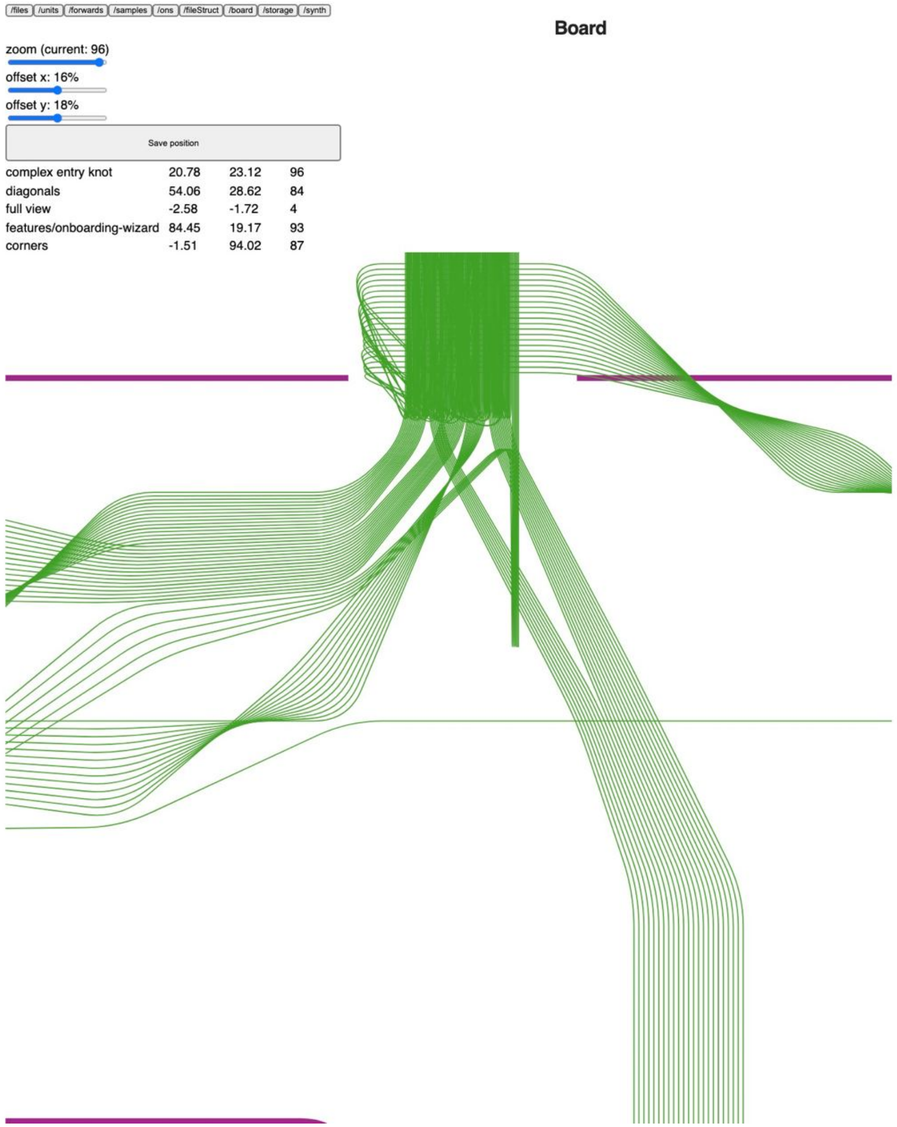
И попытки разобраться в ней

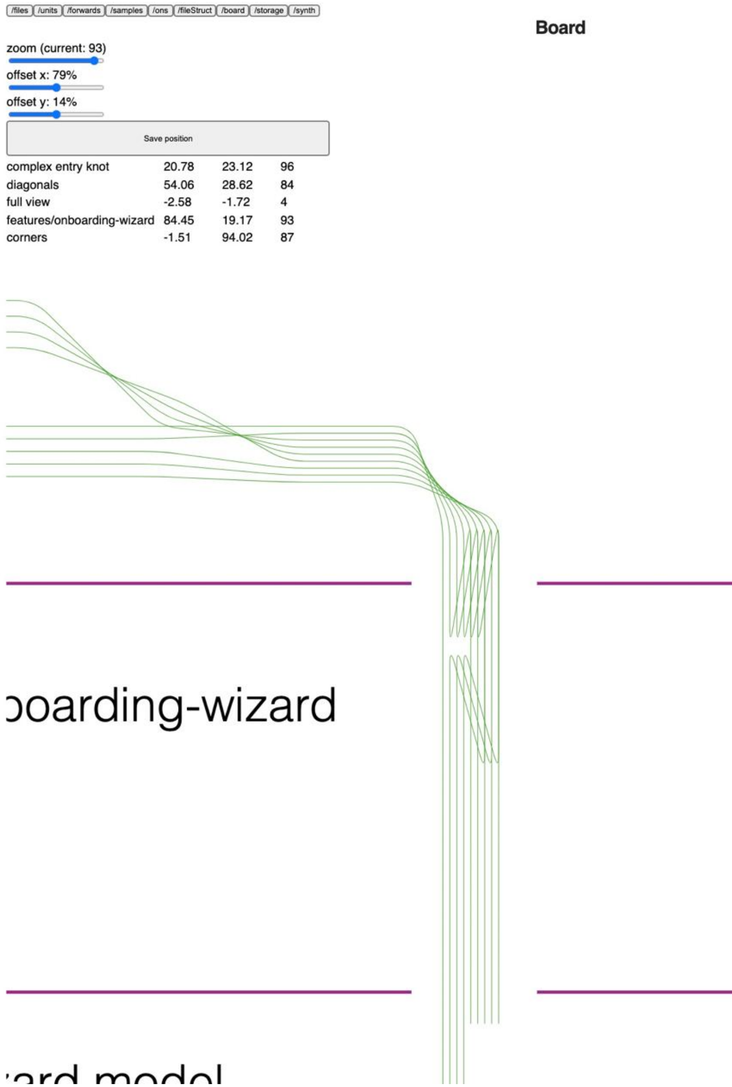


Правила роутинга линий языком математики

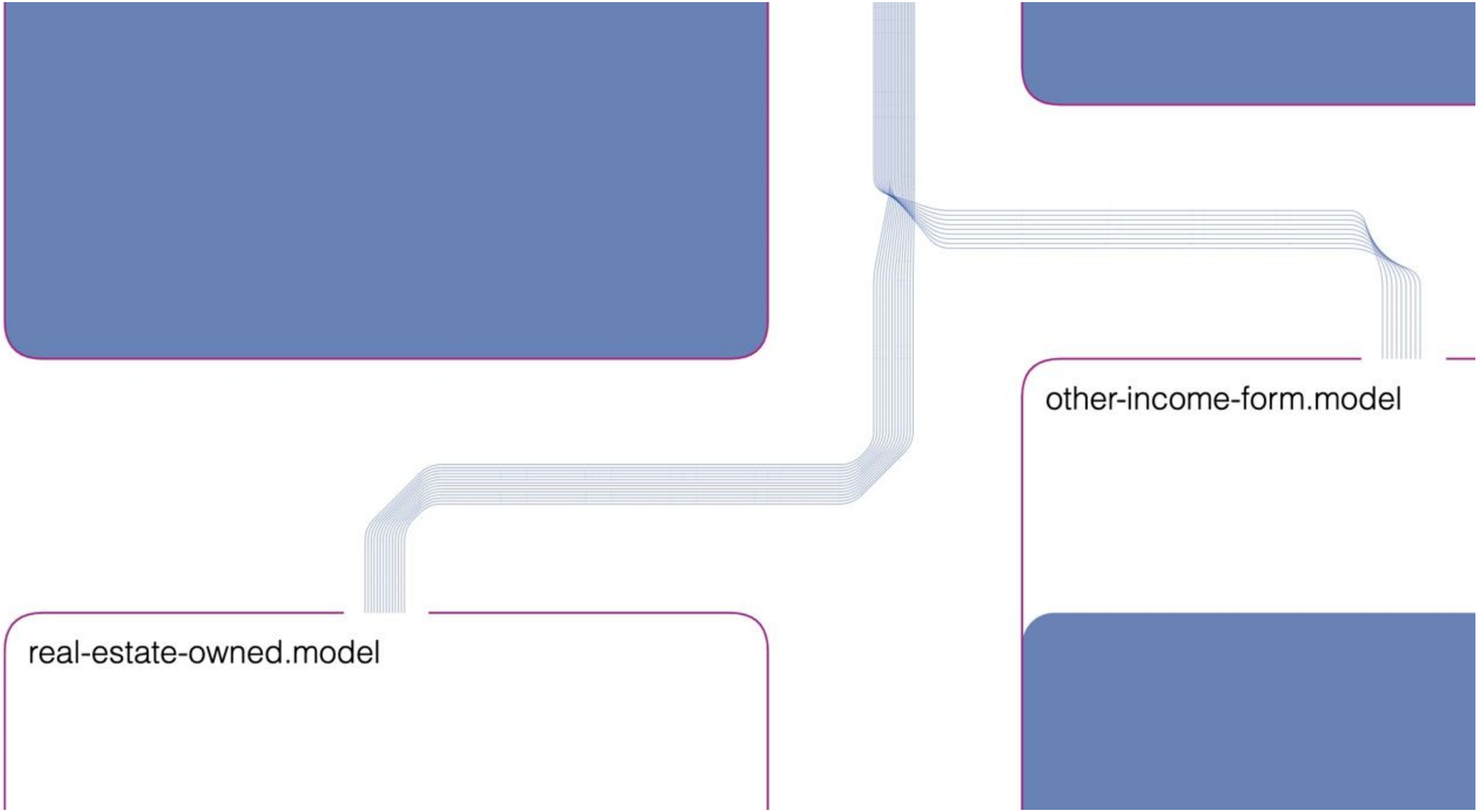




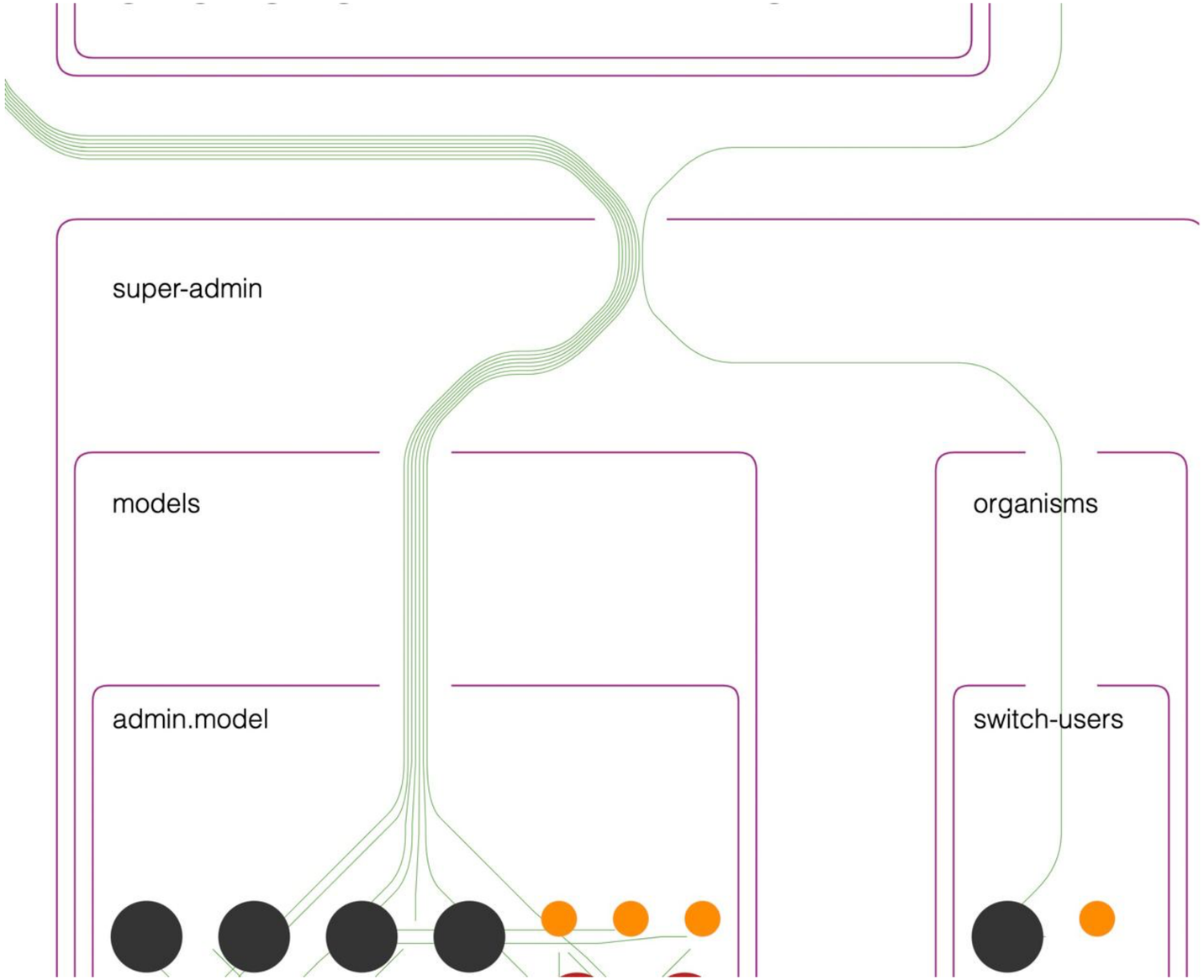




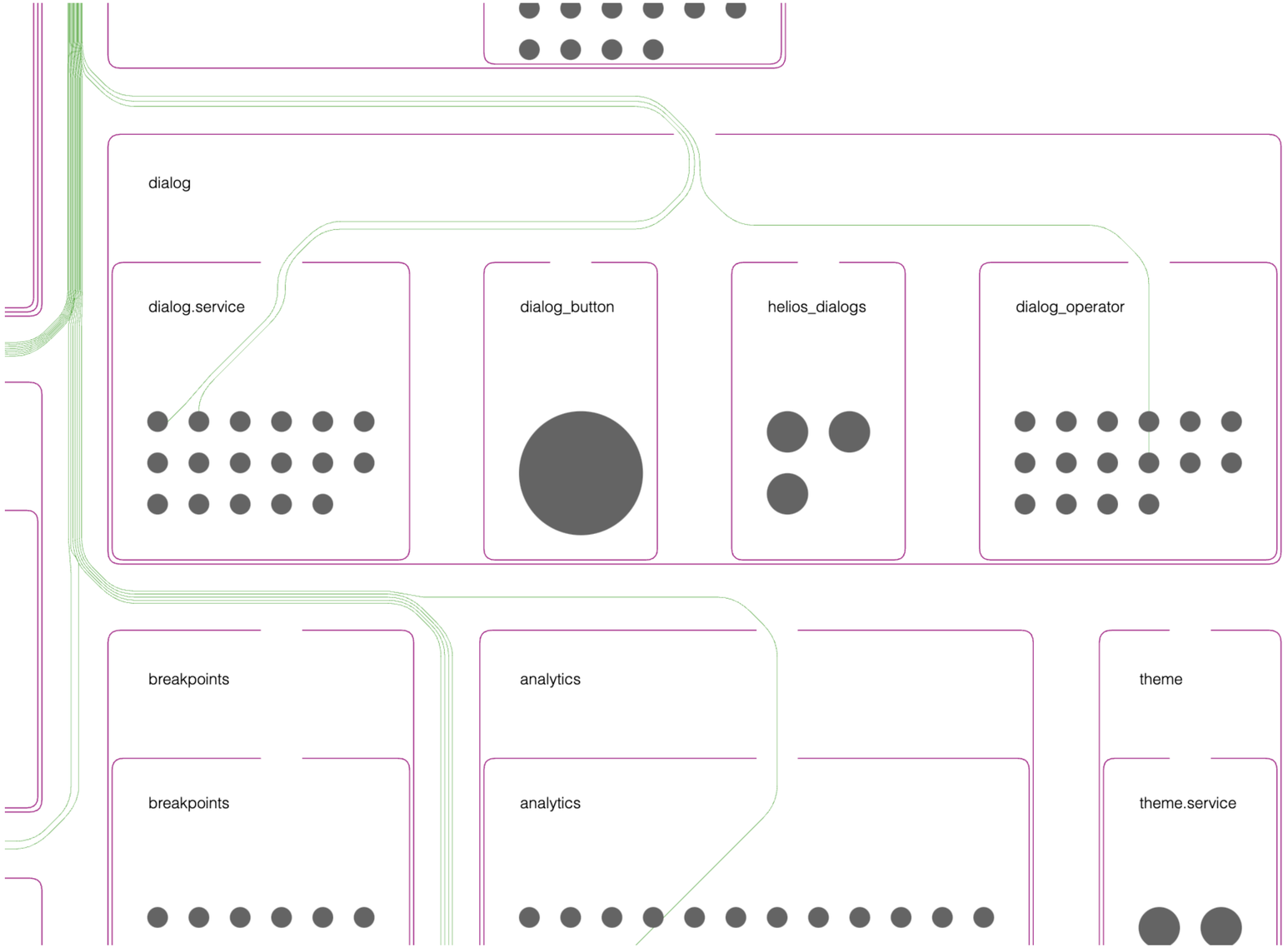
Визуальные баги



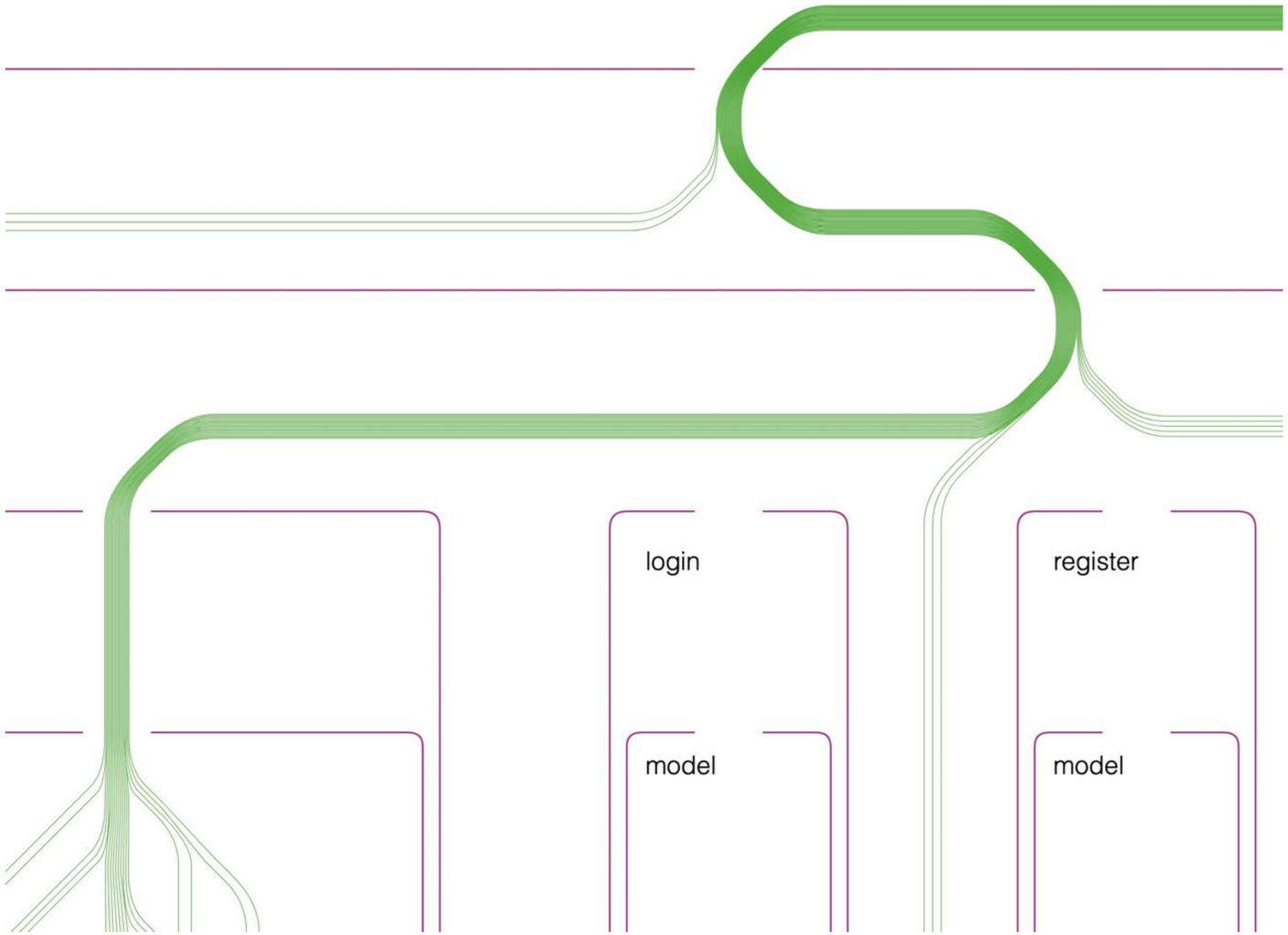
Детали роутинга линий



Детали роутинга линий



Детали роутинга линий



Детали роутинга линий

Роутинг и оптимизация линий: итог

Самая сложная задача из всех

С точки зрения дизайна — одна из самых важных

Цели:

- позволить проследить, какая сущность с какой связана
- минимизировать количество пересечений

Работает в особой системе координат, оперирует относительными направлениями и динамическим количеством линий в клетке

Готовых алгоритмов нет

Может ли помочь ИИ?

Да, но скорее как AlphaGo, чем Midjourney

ИИ уже используется в алгоритмах — он может найти оптимальную компоновку блоков, линий, но результат его расчётов всё равно нужно рисовать вручную.

Задача во многом дизайнерская

Нужно сделать «дизайн систему» — свод подходов и правил для дальнейших действий

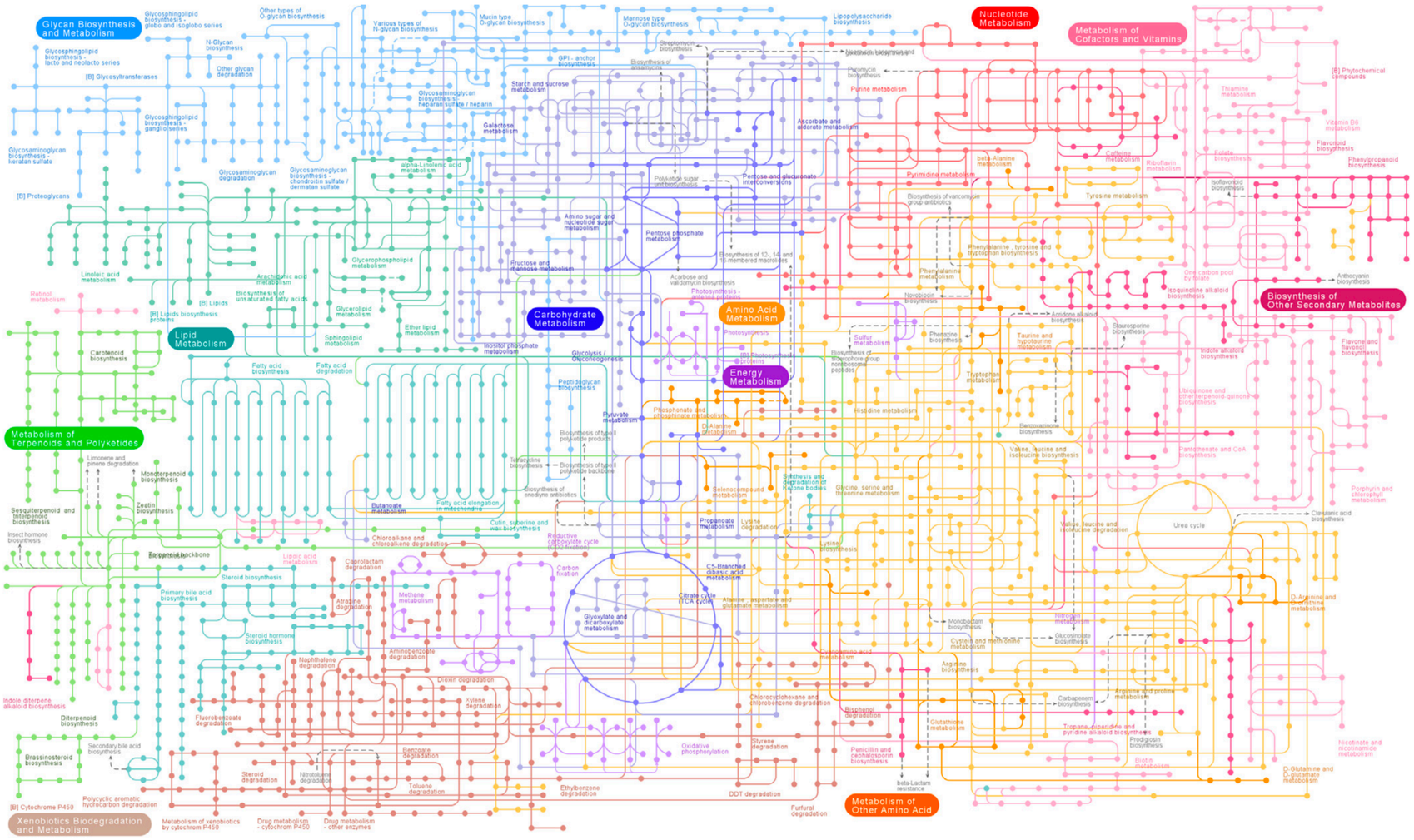
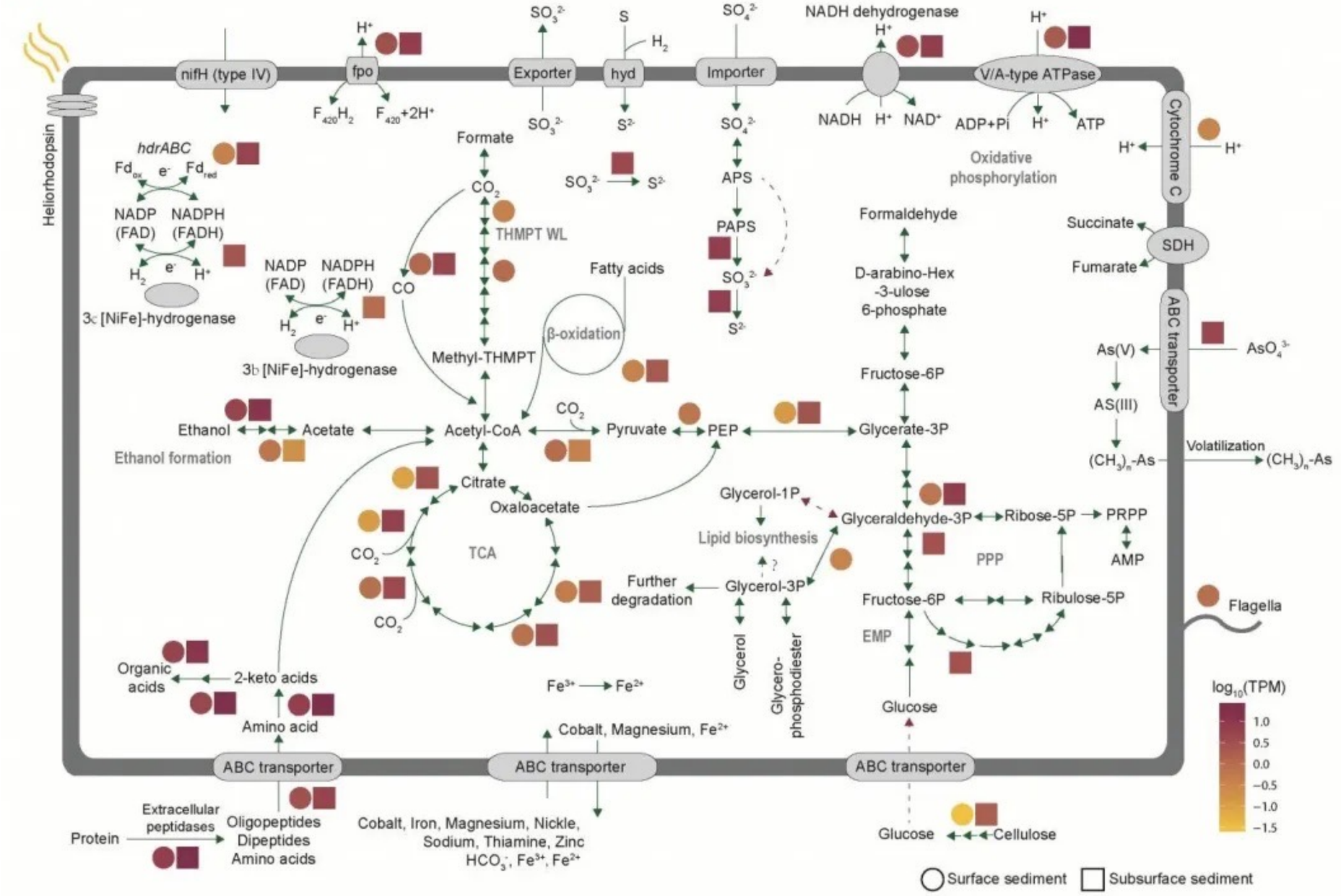
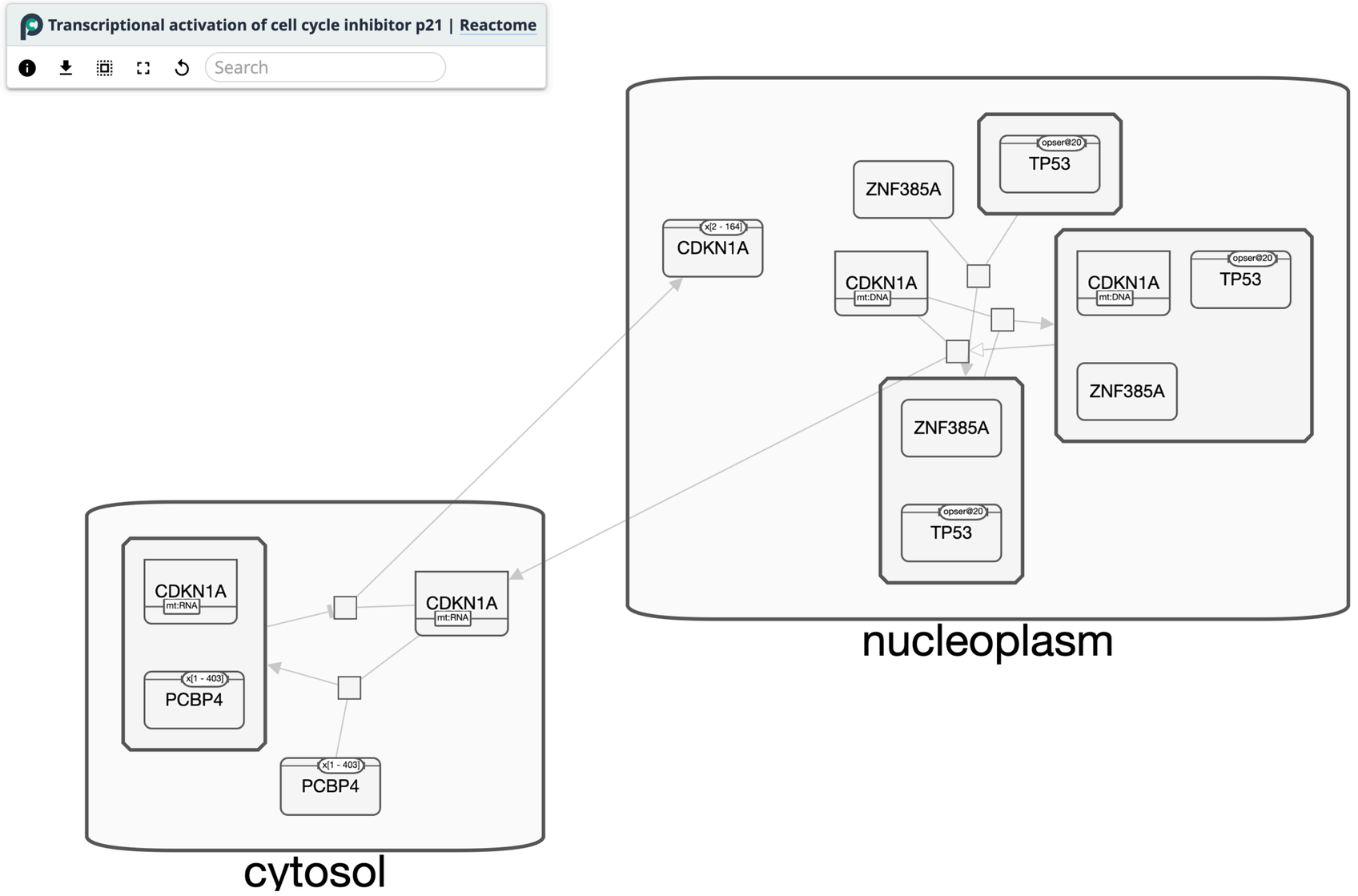


Fig. 2. Engineering-based wring diagram depicting the metabolic pathways included in the Kyoto Encyclopedia of Genes and Genomes (KEGG) database (see Kanehisa and Goto, 2000). Each node in the circuit corresponds to a protein (*source:* http://rest.kegg.jp/get/map01100/image; reproduced with permission).

Биохимия: те же проблемы. Метаболические пути в организме



Биохимия: те же проблемы. Метаболические пути в организме



Newt: визуализатор биохимических путей

github.com/iVis-at-Bilkent/newt

Выводы

Для визуализации необходима основа, подход, который изначально спроектирован для отображения: effector — стейт менеджер, оптимизированный для разработки больших приложений

Я хочу сделать продукт, который бы вывел разработку на новый уровень

Задачу можно решить и я знаю, как. У меня есть план, эффектор — его первая фаза, теперь наступила вторая — разработка самой визуализации

Релиз — в течении пары лет, подписывайтесь на effector news и канал о разработке визуализации

t.me/effector_news — канал с новостями

t.me/lines_of_code_diagrams — канал о разработке визуализации

Спасибо за внимание

effector.dev — сайт проекта 

t.me/effector_news — канал с новостями

t.me/lines_of_code_diagrams — канал о разработке визуализации

t.me/ZeroBias — по личным вопросам