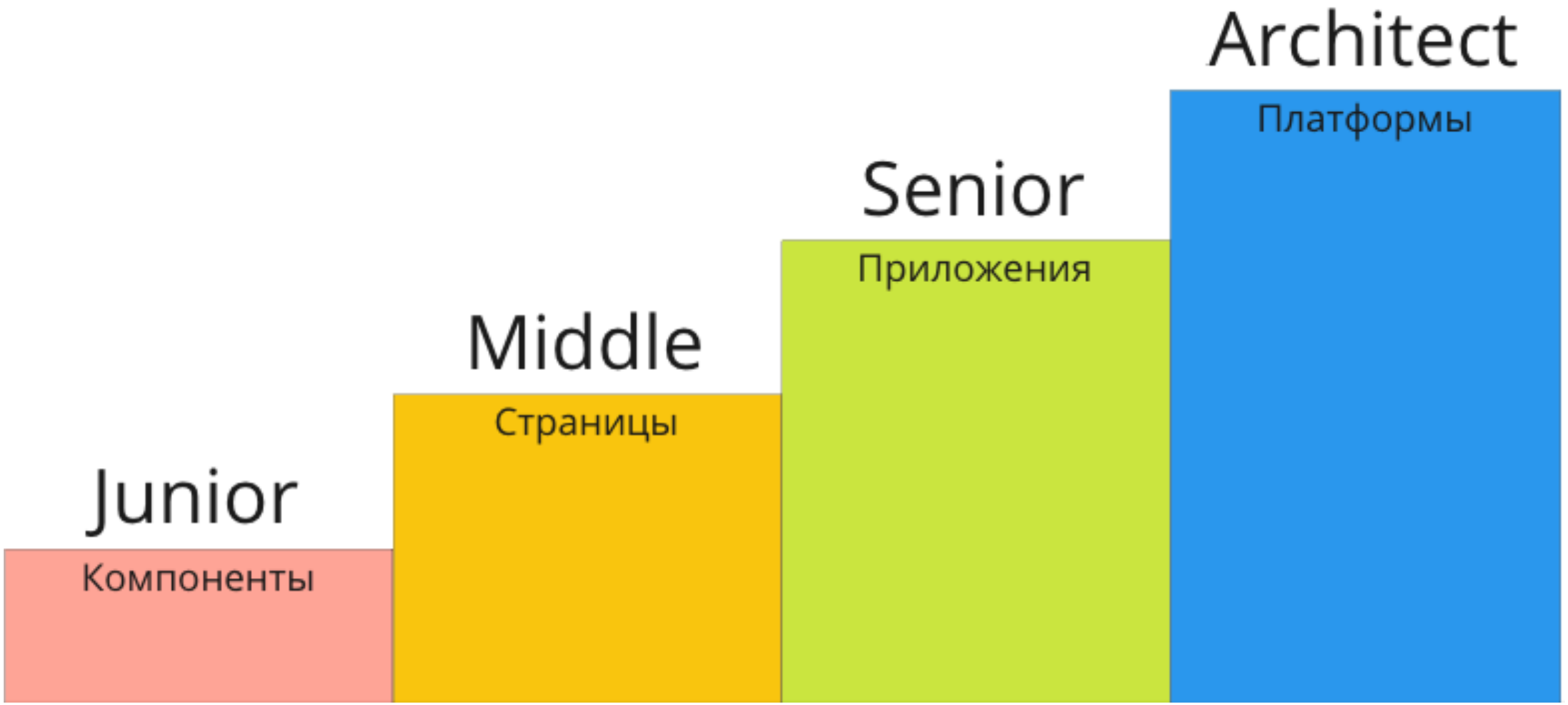
Инструментарий solution-архитектора: правильные решения в условиях неопределенности

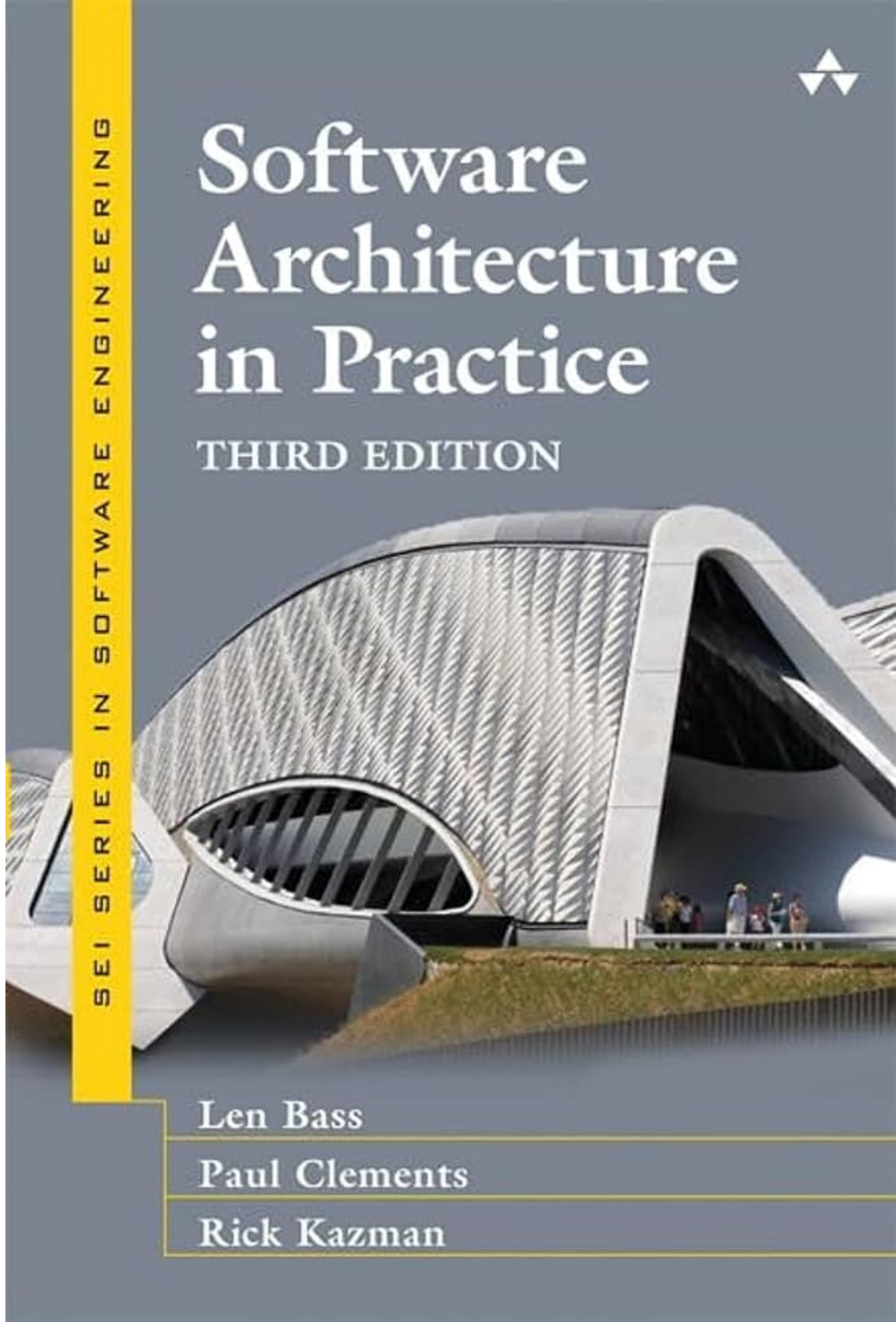
Иван
Малюгин
Билайн

предисловие

путь разработчика



архитектура программного обеспечения на практике




Архитектура . . .
Определение стейкхолдеров
Сбор и скоринг требований
Бюджетирование и определение рисков
ну его к черту . . .
. . . буду техническим экспертом

оказалось все наоборот

«Прежде всего, архитектор программного обеспечения — это программист; и он продолжает оставаться таковым. Не верьте, если кто-то вам скажет, что архитекторы не занимаются программированием и решают проблемы более высокого уровня»

Роберт Мартин

«Архитектор, сидящий в башне из слоновой кости, будет либо серьезным препятствием для микросервисной архитектуры, либо превратится в формальную, ничего не решающую должность»

Сэм Ньюмен

«Если моделировщик отстранен от процесса программной реализации, то он не приобретает или быстро теряет интуитивное понимание требований и особенностей реализации»

Эрик Эванс

архитектор - медиатор

бизнес

разработка

архитектура в FE-стеке

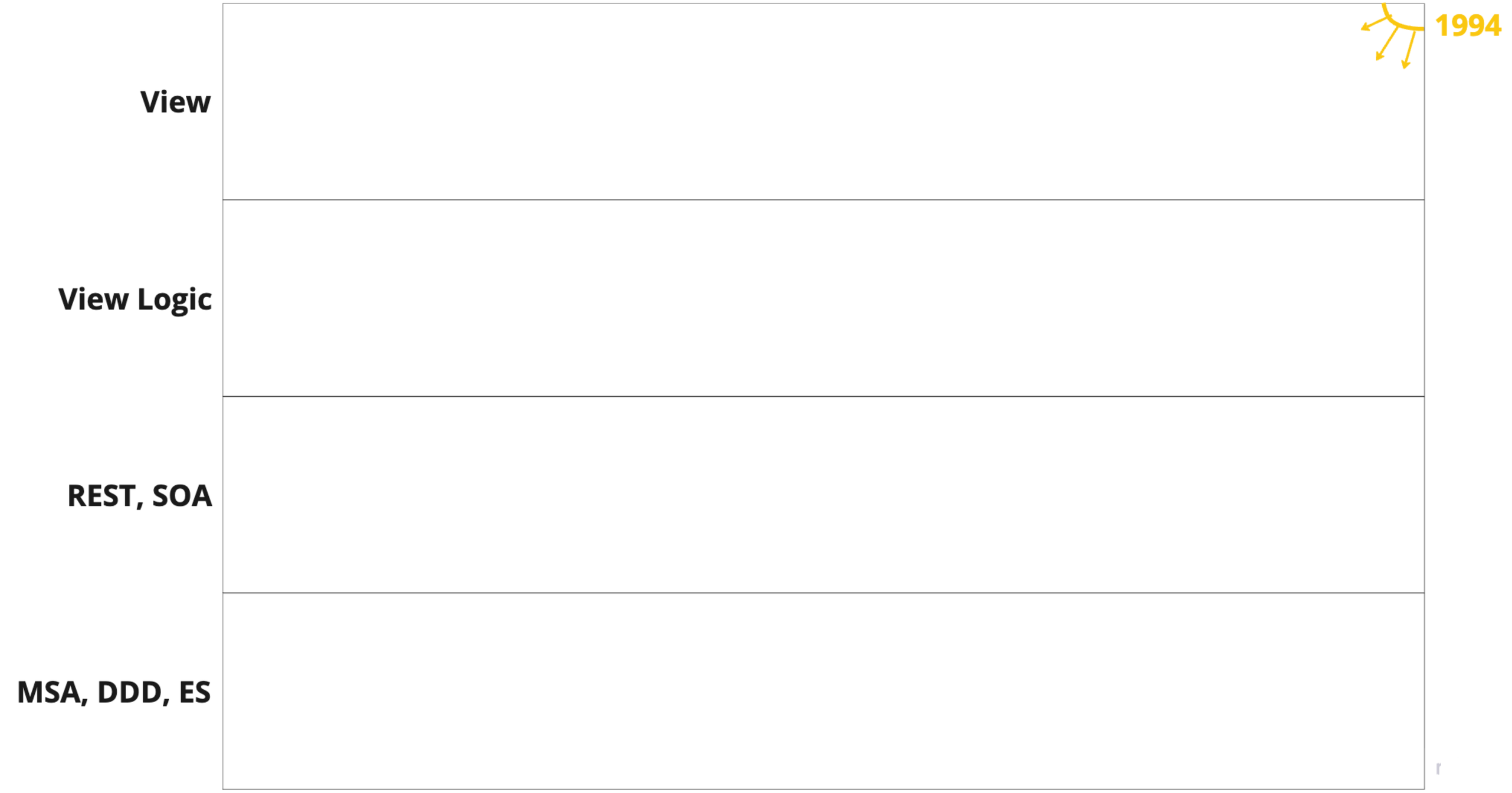
View

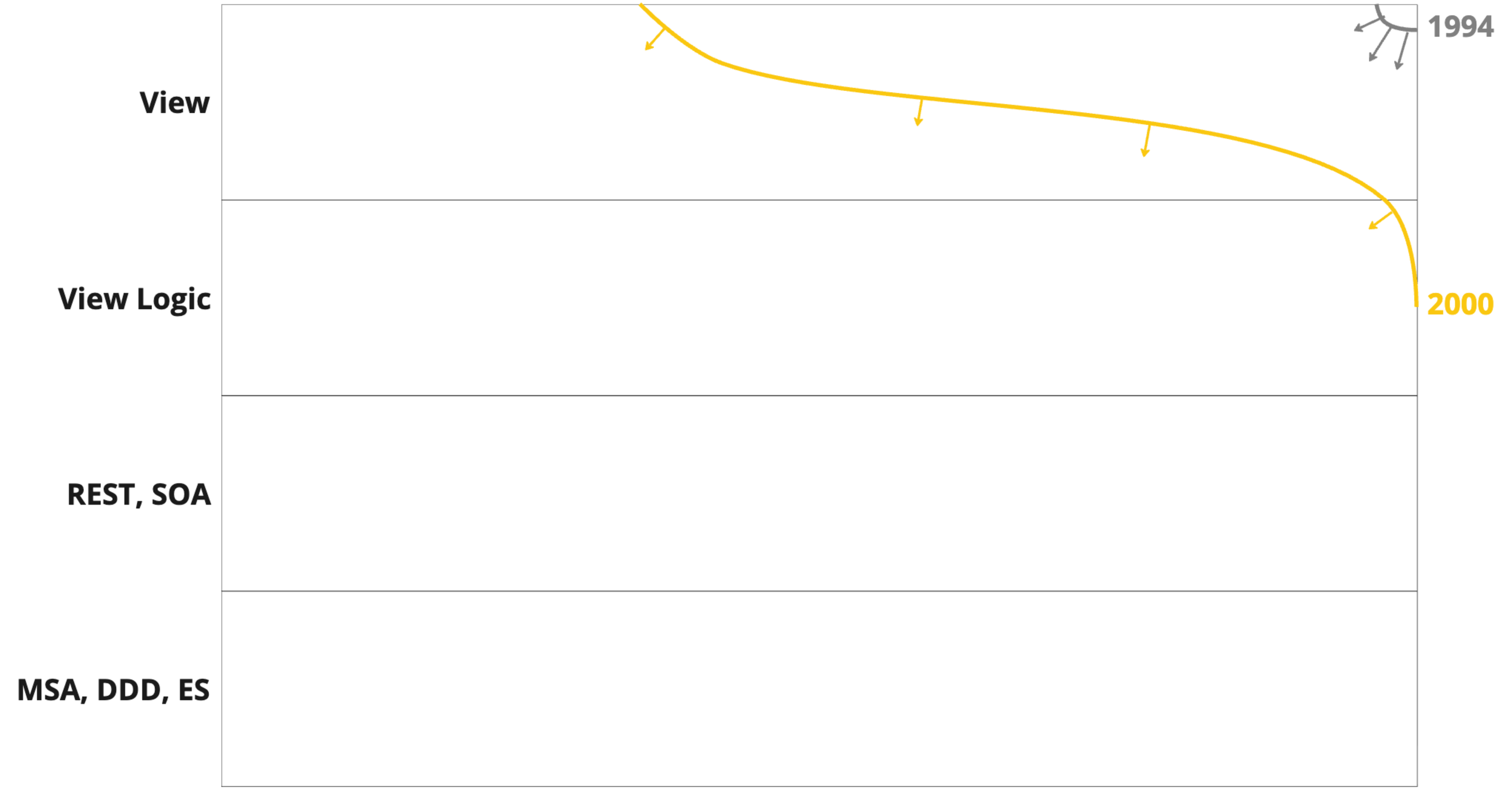
View Logic

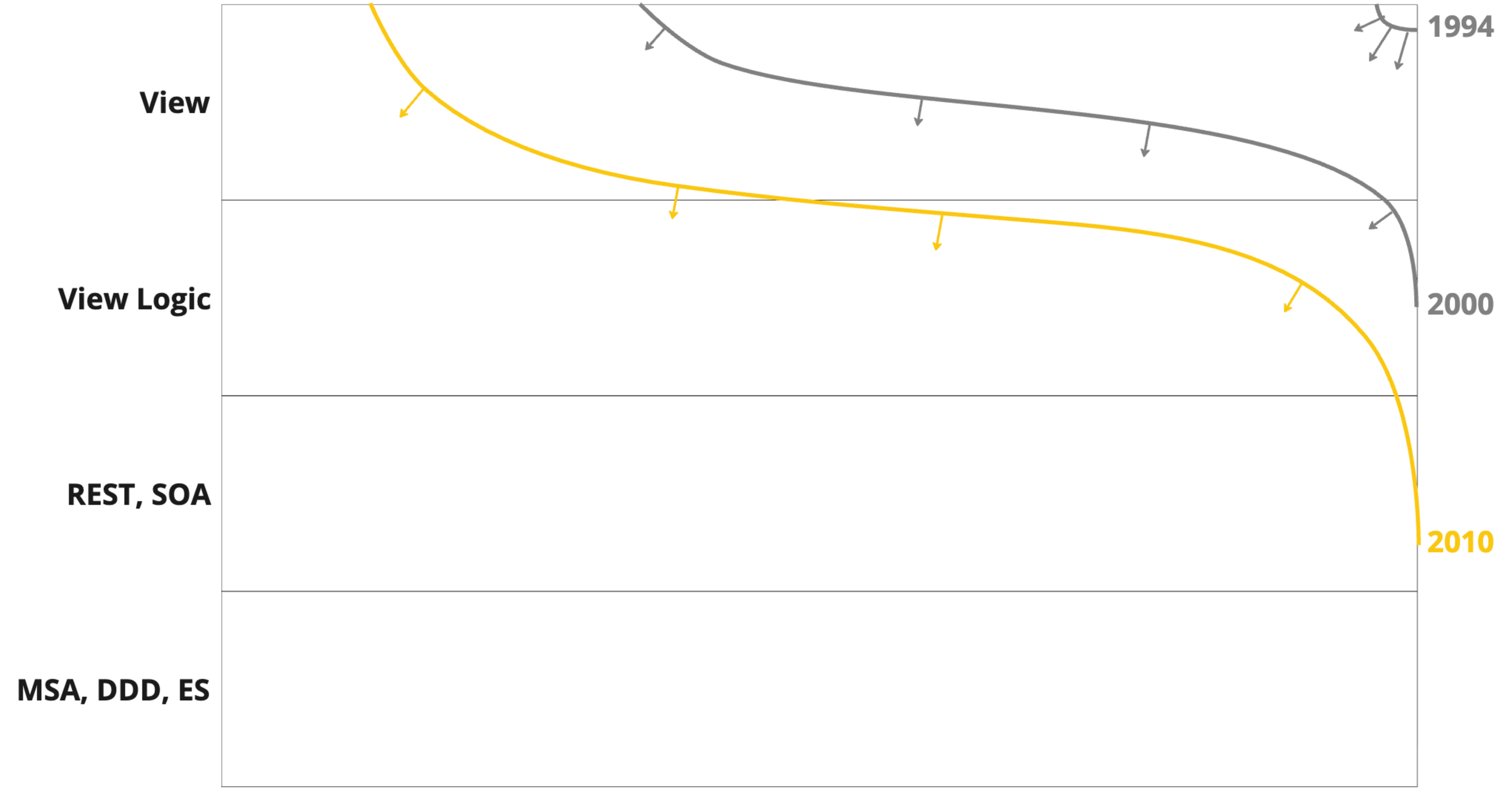
REST, SOA

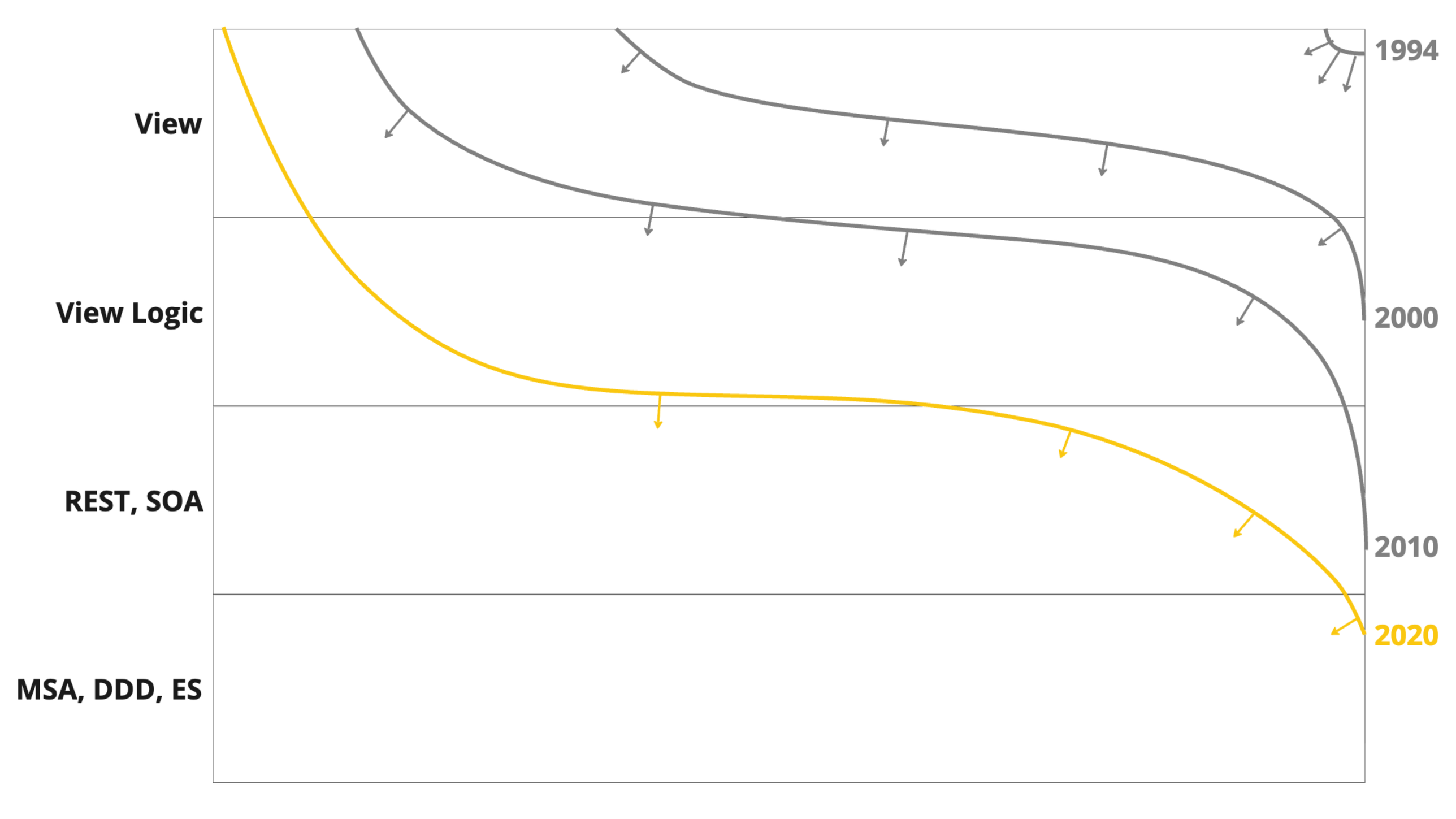
MSA, DDD, ES

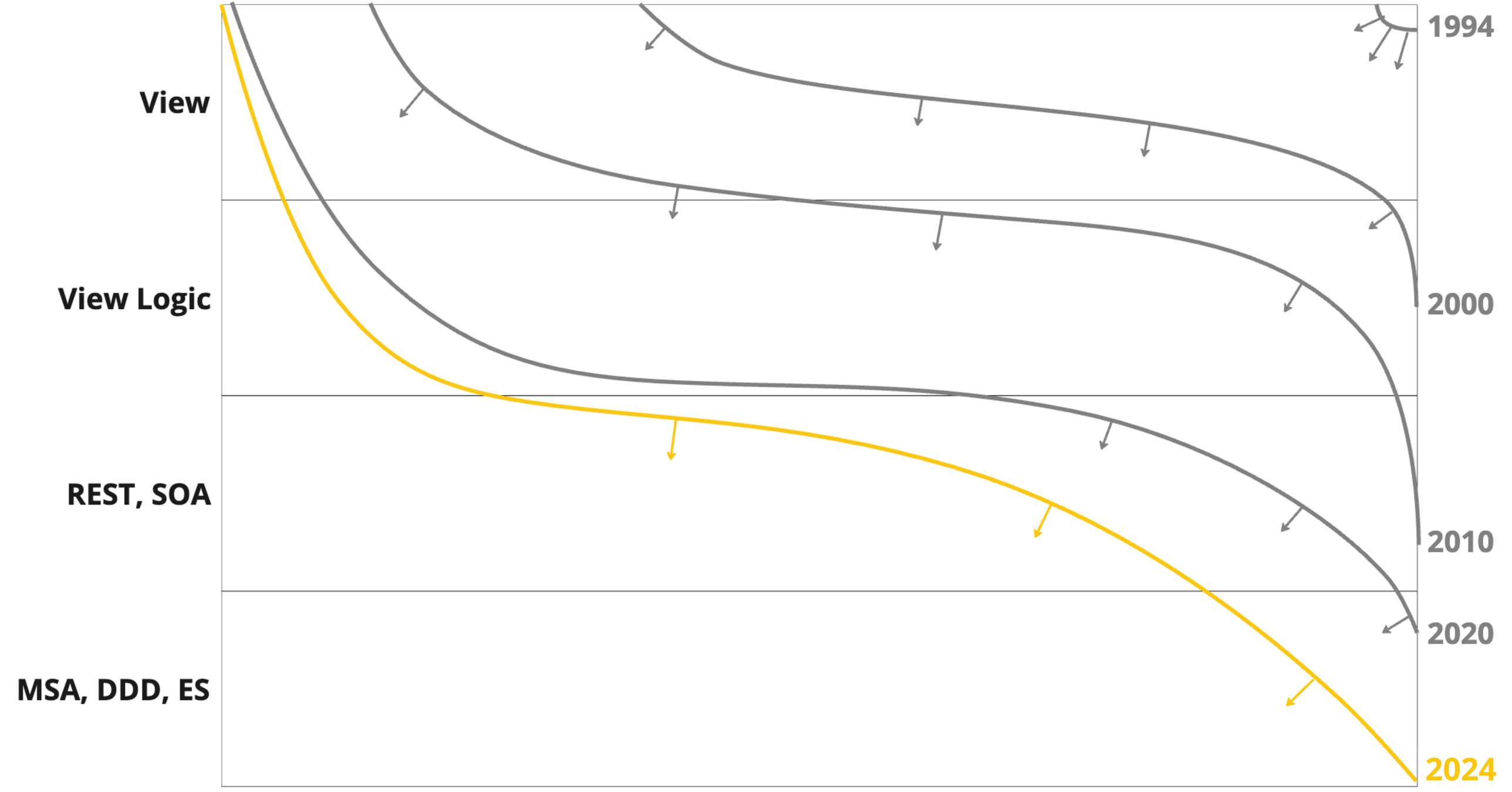
эволюция FE-Экосистемы


View
View Logic
REST, SOA
MSA, DDD, ES
1994


View
View Logic
REST, SOA
MSA, DDD, ES
1994
2000


View
View Logic
REST, SOA
MSA, DDD, ES
1994
2000
2010


View
View Logic
REST, SOA
MSA, DDD, ES
1994
2000
2010
2020


View
View Logic
REST, SOA
MSA, DDD, ES
1994
2000
2010
2020
2024

высокая потребность в архитектуре

01 устройство фреймворков

02 микрофронты

03 FrontOps

04 написание NodeJS-приложений

золотой путь
в архитектуре

сколько лет человеку, родившемуся в 1924 году?

интуитивный выбор

Плюсы: быстро, дешево

Минусы: поверхностно, на прошлом опыте

высокая потребность в архитектуре

01 Либо вы не видите альтернатив, выбирая на основе похожего опыта;

высокая потребность в архитектуре

01 Либо вы не видите альтернатив, выбирая на основе похожего опыта;

02 Либо система уже настолько сложна, что у вас не остается других вариантов.

Цена ошибки

Совершенная на ранних этапах ошибка, может вылиться провал проекта или годы рефакторинга;

принятие решений

«В начале процесса создания ПО архитектура программно-интенсивной системы представляет собой изложение видения. В конце архитектура каждой такой системы становится отражением миллиардов маленьких и больших, преднамеренных и случайных проектных решений, принятых на этом пути.»

Грейди Буч

осознанные решения

01 У одной проблемы может быть множество потенциальных решений;

02 Любое решение - компромисс с преимуществами и недостатками;

векторы
принятия решений

01 Трудозатраты на принятие решения ~ цене ошибки;

векторы
принятия решений

01 Трудозатраты на принятие решения ~ цене ошибки;

02 Решения должны быть аргументированными;

векторы
принятия решений

01 Трудозатраты на принятие решения ~ цене ошибки;

02 Решения должны быть аргументированными;

03 Решения не должны ограничивать архитектуру.

векторы
принятия решений

01 Трудозатраты на принятие решения ~ цене ошибки;

02 Решения должны быть аргументированными;

03 Решения не должны ограничивать архитектуру.

«Чем больше в системе степеней свободы,
тем система сложнее»

Элияху Голдратт

векторы
принятия решений

01 Трудозатраты на принятие решения ~ цене ошибки;

02 Решения должны быть аргументированными;

03 Ограничения, накладываемые решениями должны быть осознанными.

architecture decision record (ADR)

01 Короткое название

02 Контекст

03 Проблема

04 Решения

05 Последствия

06 Рассмотренные альтернативы

*** статус, участники, ограничения, аргументы, ссылки, приложения, заметки

матрица трассировки

Как работать с асинхронностью?

	Порог входа	Рост сложности	Бойлерплейт	Скорость работы	Документация	
Callback	5	1	3	5	5	19
Promise	4	3	4	5	5	21
Generators	1	1	4	5	4	15
RxJS	1	5	5	4	4	19

важность контекста

Как работать с асинхронностью?

	Порог входа	Рост сложности	Бойлерплейт	Скорость работы	Документация	
Callback	5	1	3	5	5	8
Promise	4	3	4	5	5	12
Generators	1	1	4	5	4	10
RxJS	1	5	5	4	4	14

противоречия и аксиомы

Отсеивание заведомо
ложных решений

все средства хороши

01 Системная аналитика (UML);

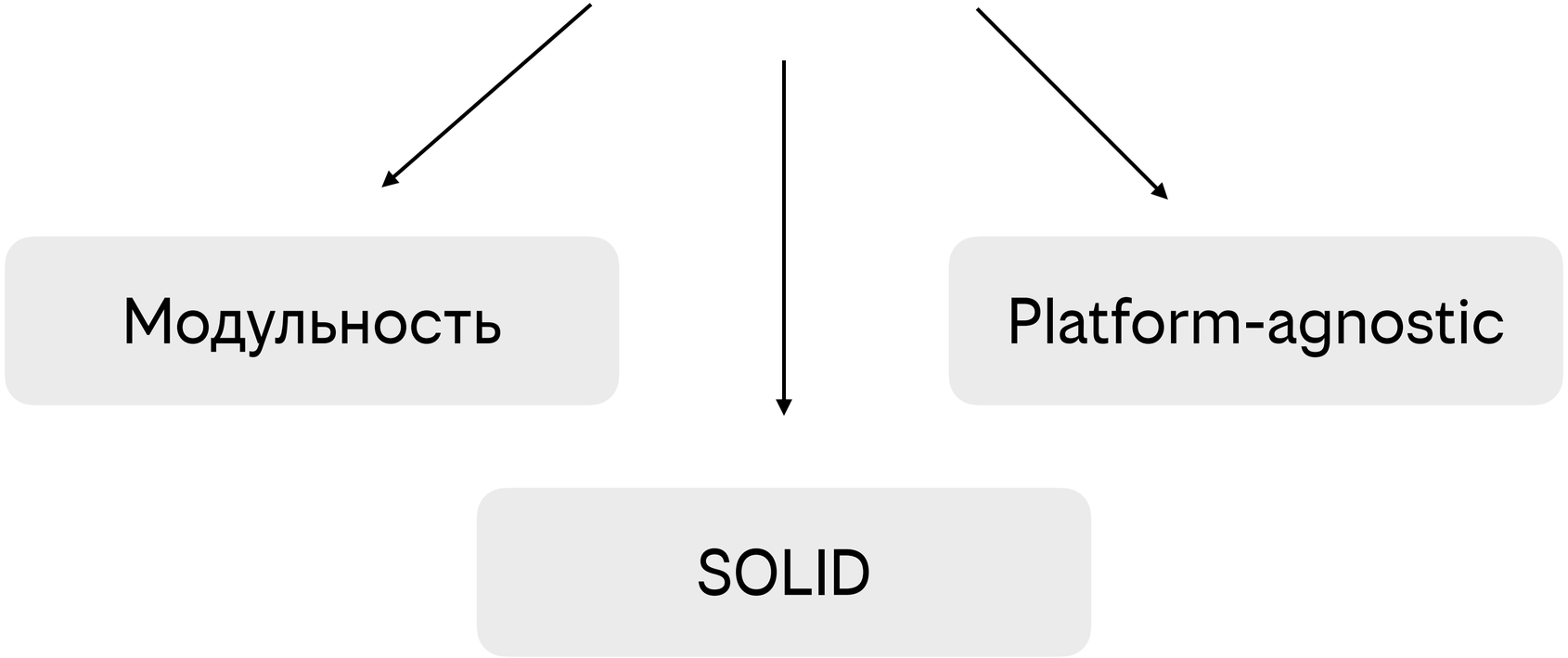
02 UX-прототипирование (CJ, Pit of Success);

design patterns

01 Готовые решения, которым можно доверять;

02 Можно и нужно адаптировать под конкретную задачу;

High Coupling / Low Cohesion



все средства хороши

01 Системная аналитика (UML);

02 UX-прототипирование (CJ, Pit of Success);

03 Бизнес-анализ (Jobs to be Done);

техники DDD

01 Близость с моделью предметной области;

техники DDD

01 Близость с моделью предметной области;

02 Core, Support, Generic - домены;

техники DDD

01 Близость с моделью предметной области;

02 Core, Support, Generic - домены;

03 Сохранение консистентности;

Event Sourcing

01 Крайне высокий порог входа;

Event Sourcing

Event Sourcing

+ Крайне высокий порог входа;

Event Sourcing

\+ Крайне высокий порог входа;

\- Сокращение роста сложности;

\- Сокращение цены ошибки.

КОГНИТИВНАЯ СЛОЖНОСТЬ

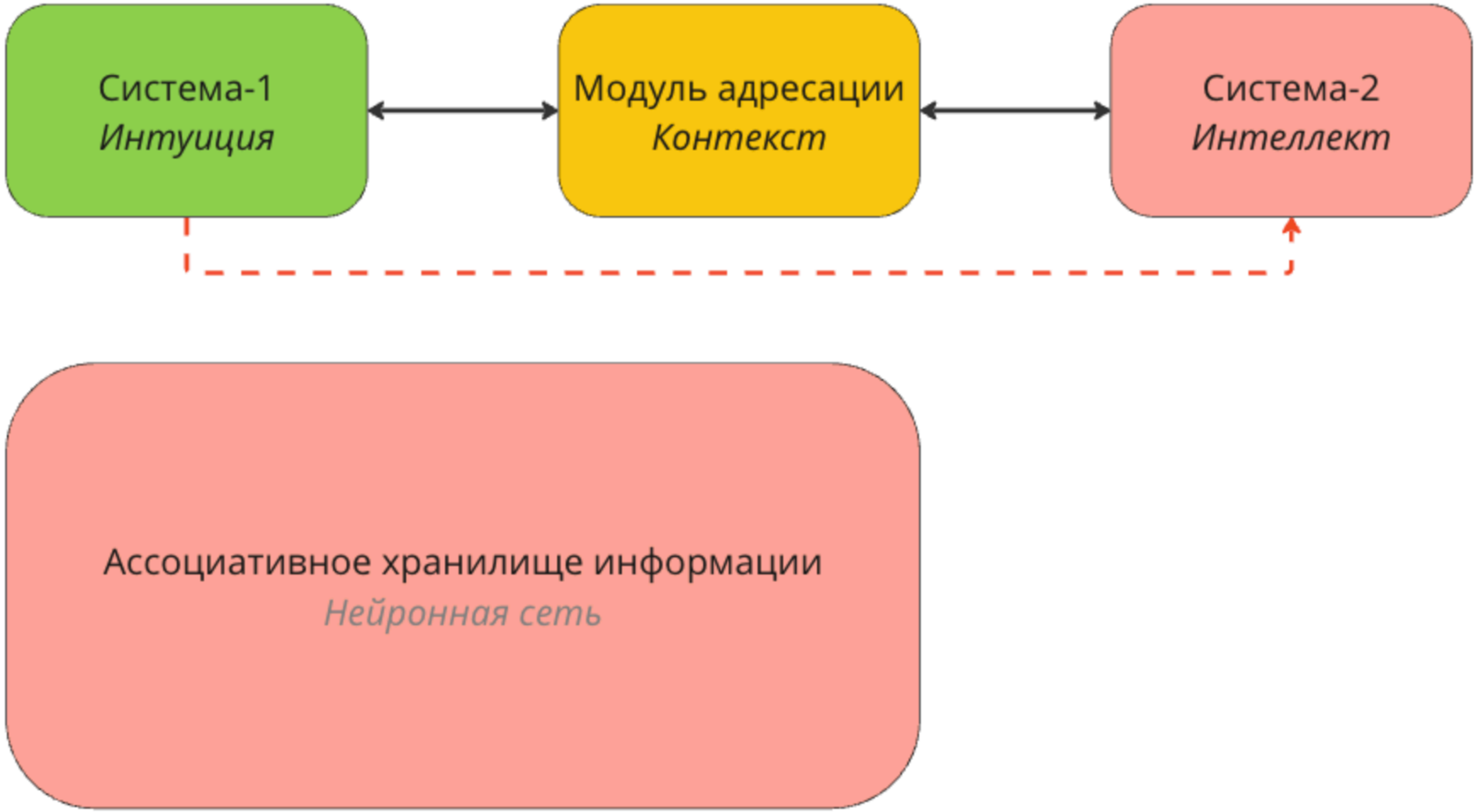
энергозатраты осознанного мышления

устройство мышления

по Дэниэлю Канеману

устройство мышления

по Дэниэлю Канеману



«Прокрастинация - это система внутреннего саботажа, когда человек откладывает задачу на потом, но при этом продолжает тратить энергию.»

причины прокрастинации

01 Задача слишком сложная для системы-1;

причины прокрастинации

01 Задача слишком сложная для системы-1;

02 Попытки заставить систему-2 решать задачу;

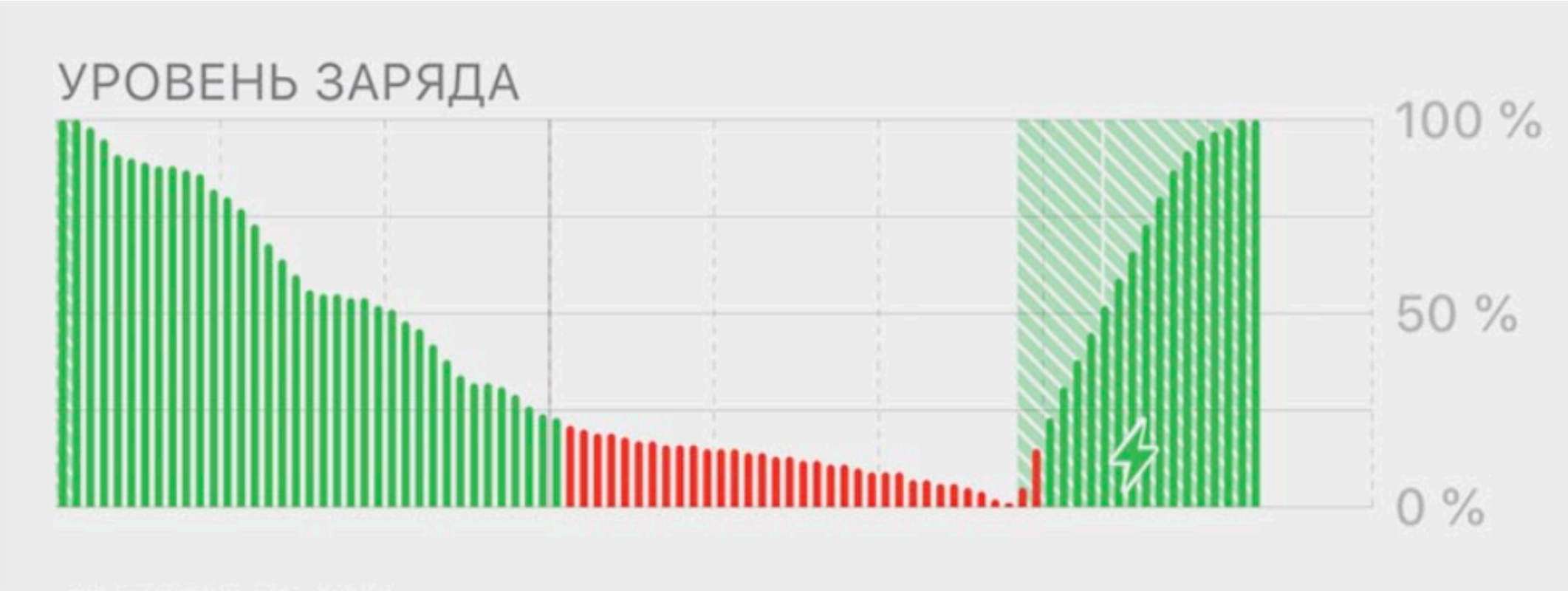
причины прокрастинации

01 Задача слишком сложная для системы-1;

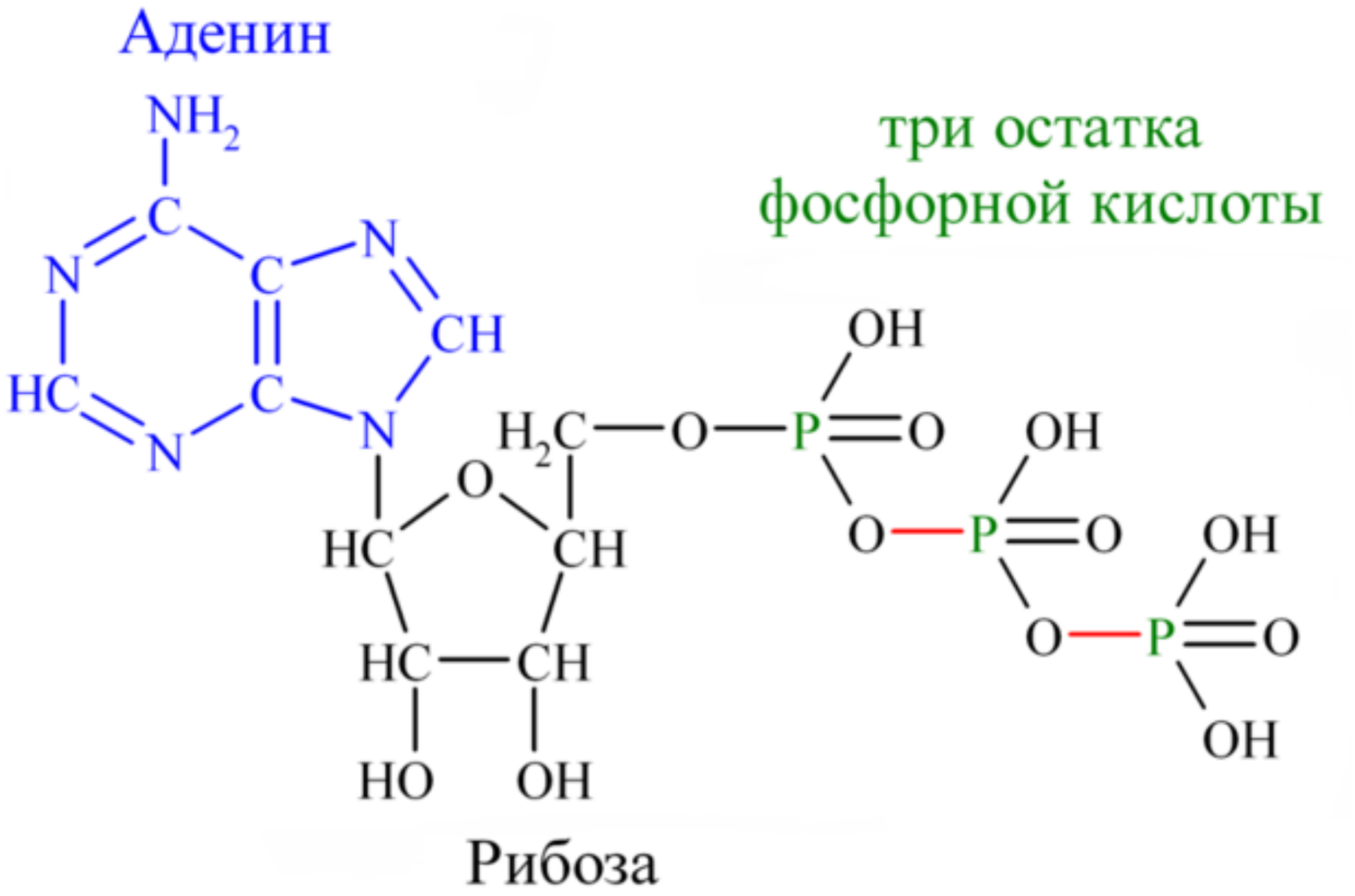
02 Попытки заставить систему-2 решать задачу;

03 Продолжительная прокрастинация;

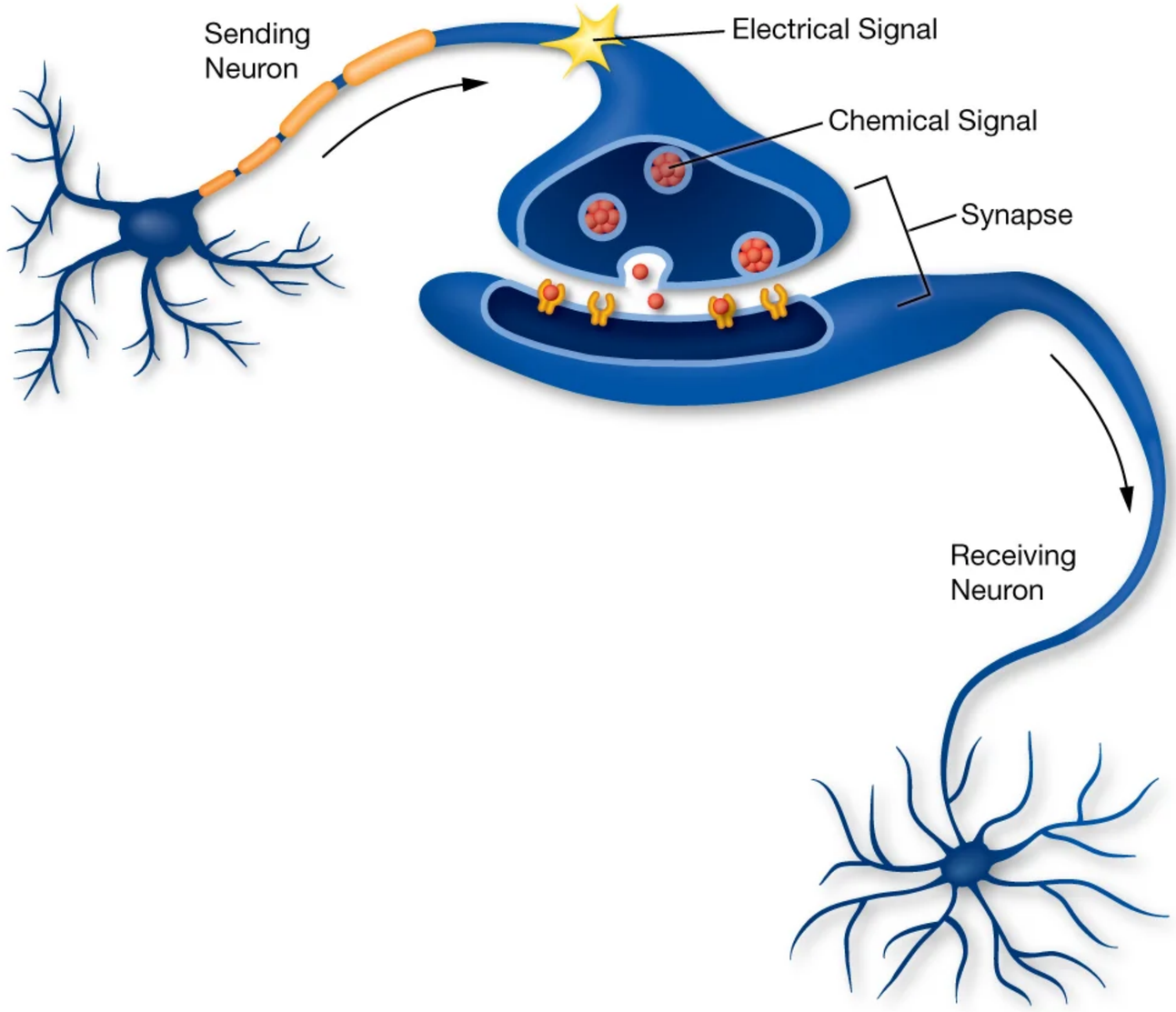
мыслетопливо



научное обоснование



научное обоснование



последствия прокрастинации

01 Сжигание времени и мыслетоплива

последствия прокрастинации

01 Сжигание времени и мыслетоплива

02 Муки совести

последствия прокрастинации

01 Сжигание времени и мыслетоплива

02 Муки совести

03 Повышение уровня кортизола

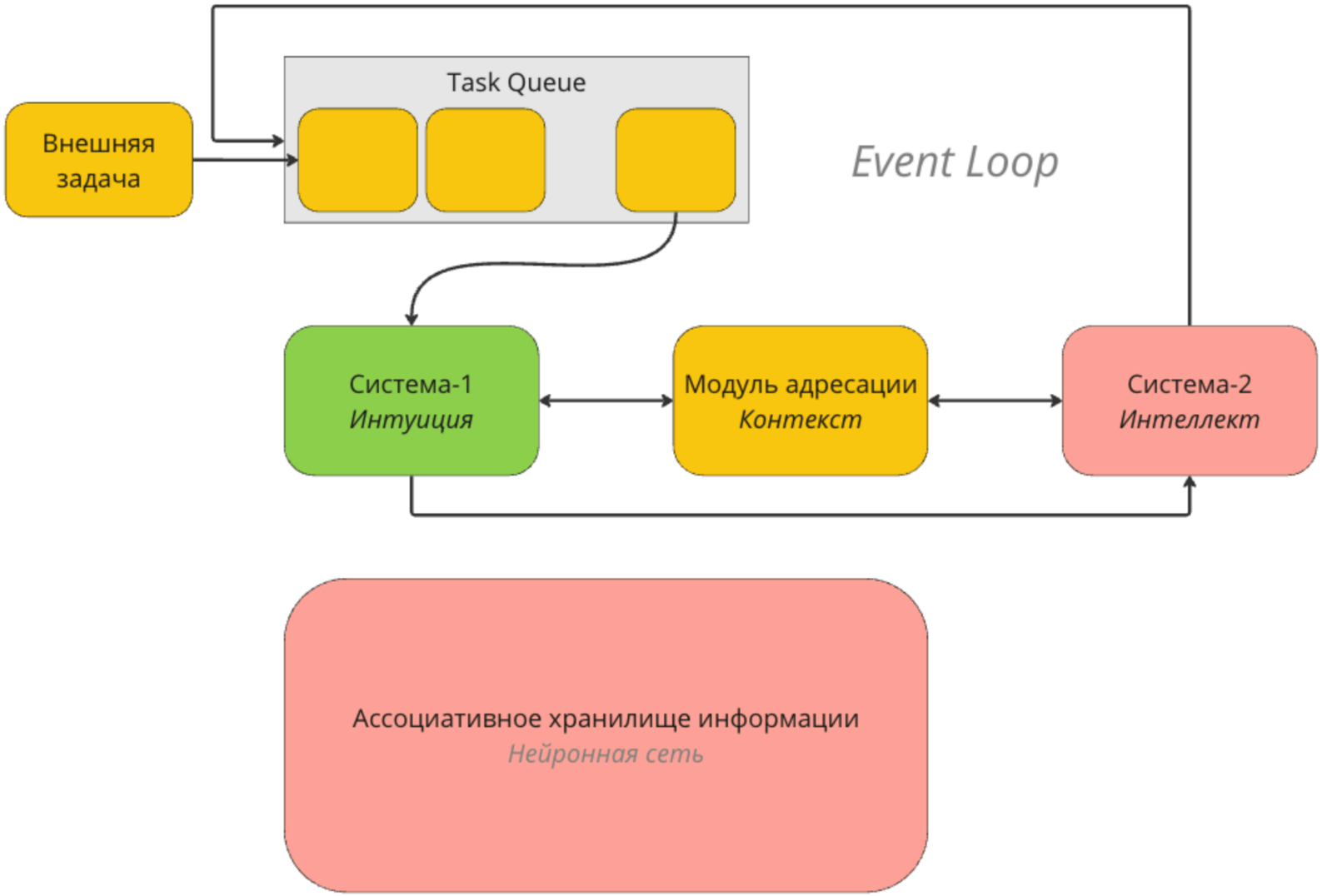
— Снижение лимита мыслетоплива

— Риск выгорания

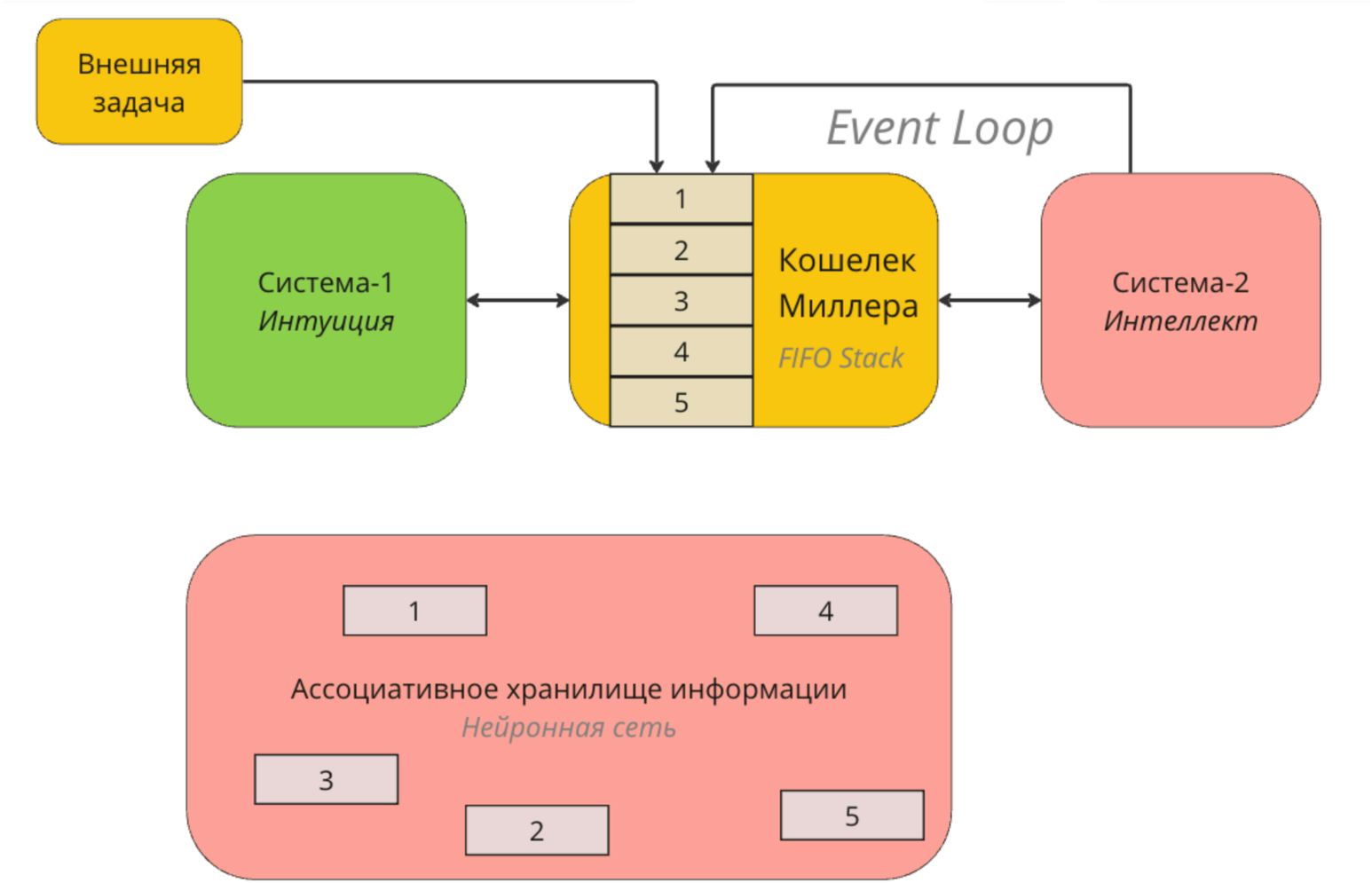
джедайские техники


В нашем мозгу есть
Event Loop

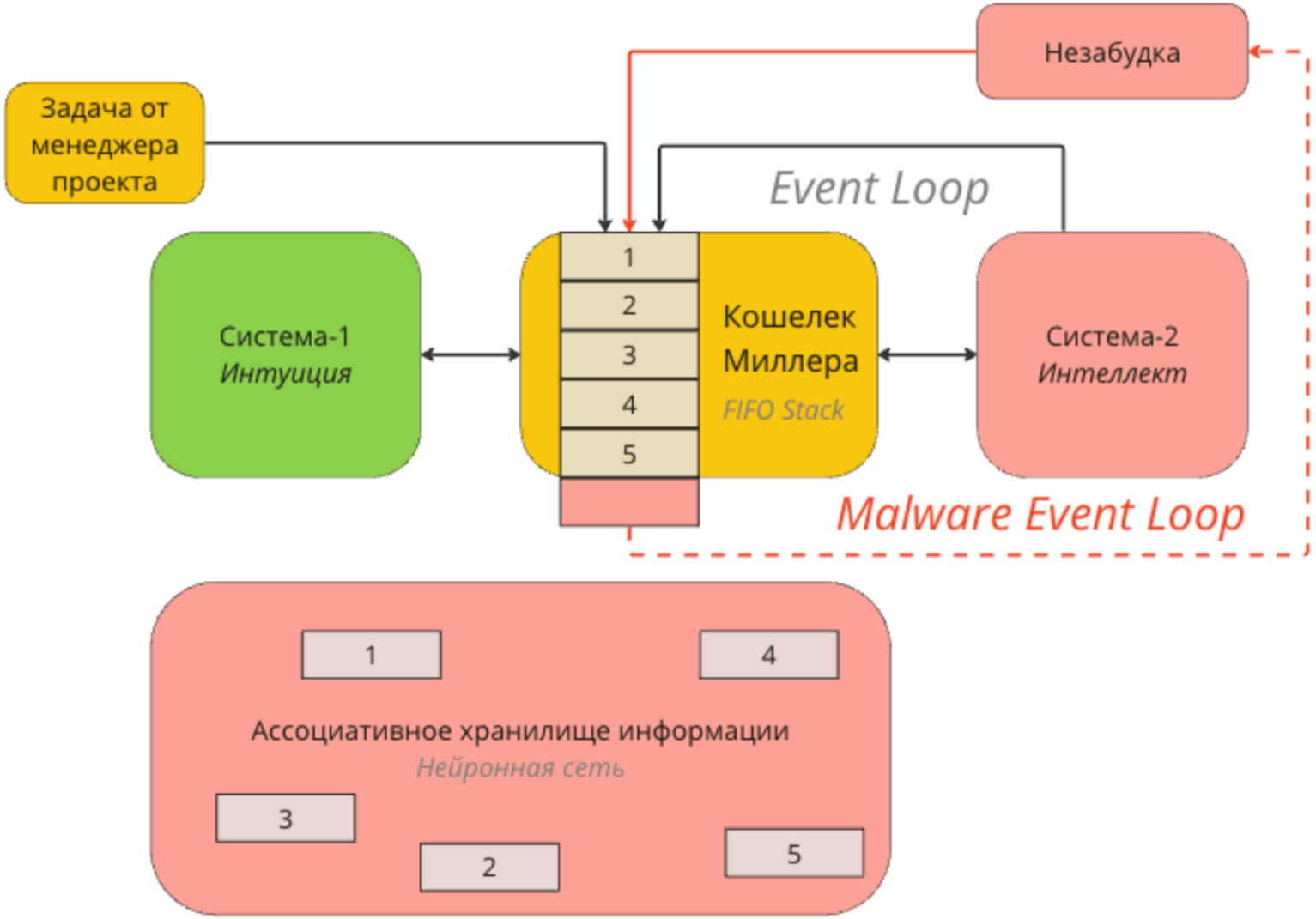
обработка задач



переключение контекста

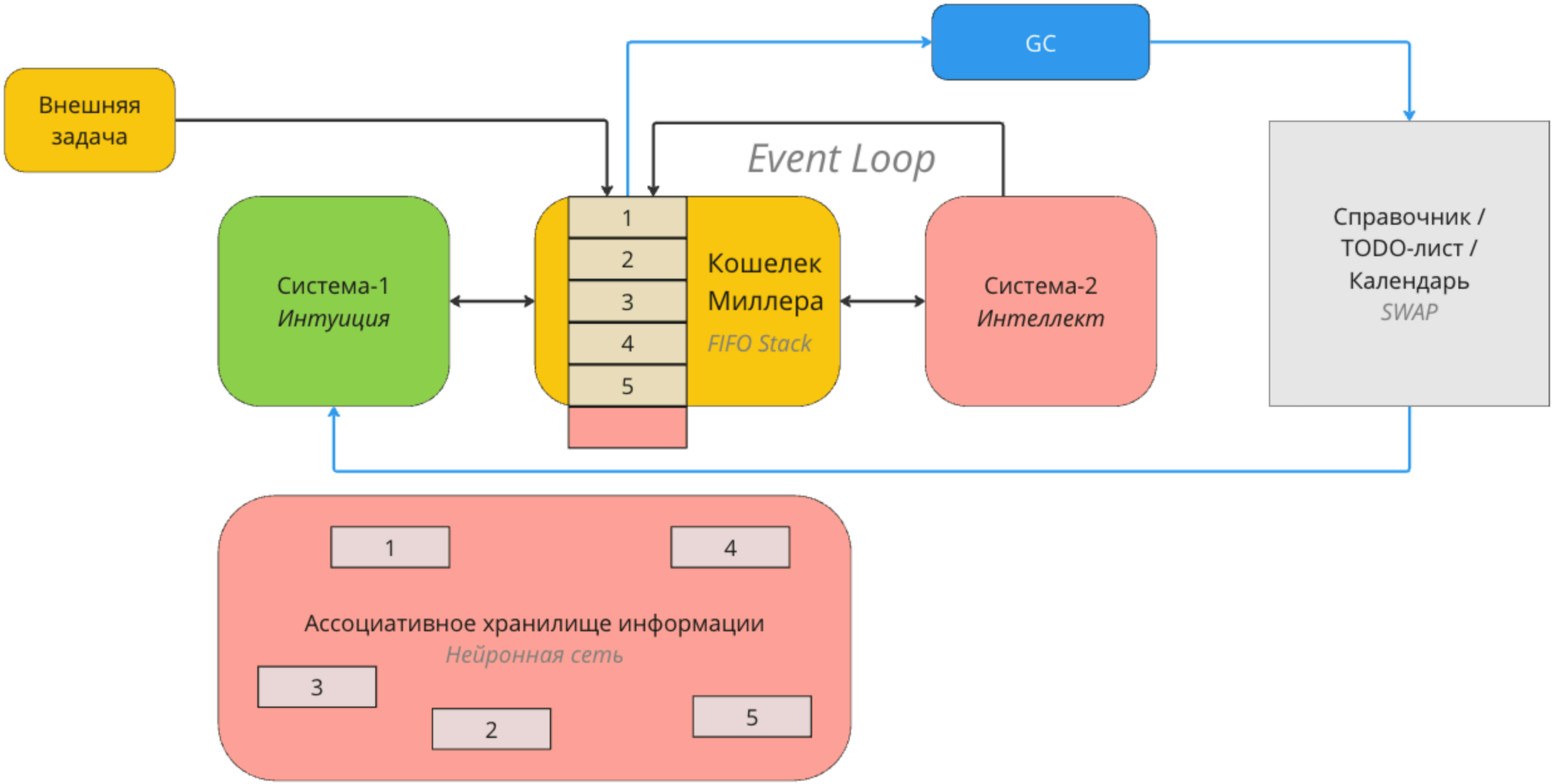


«незабудка» эффективного менеджера

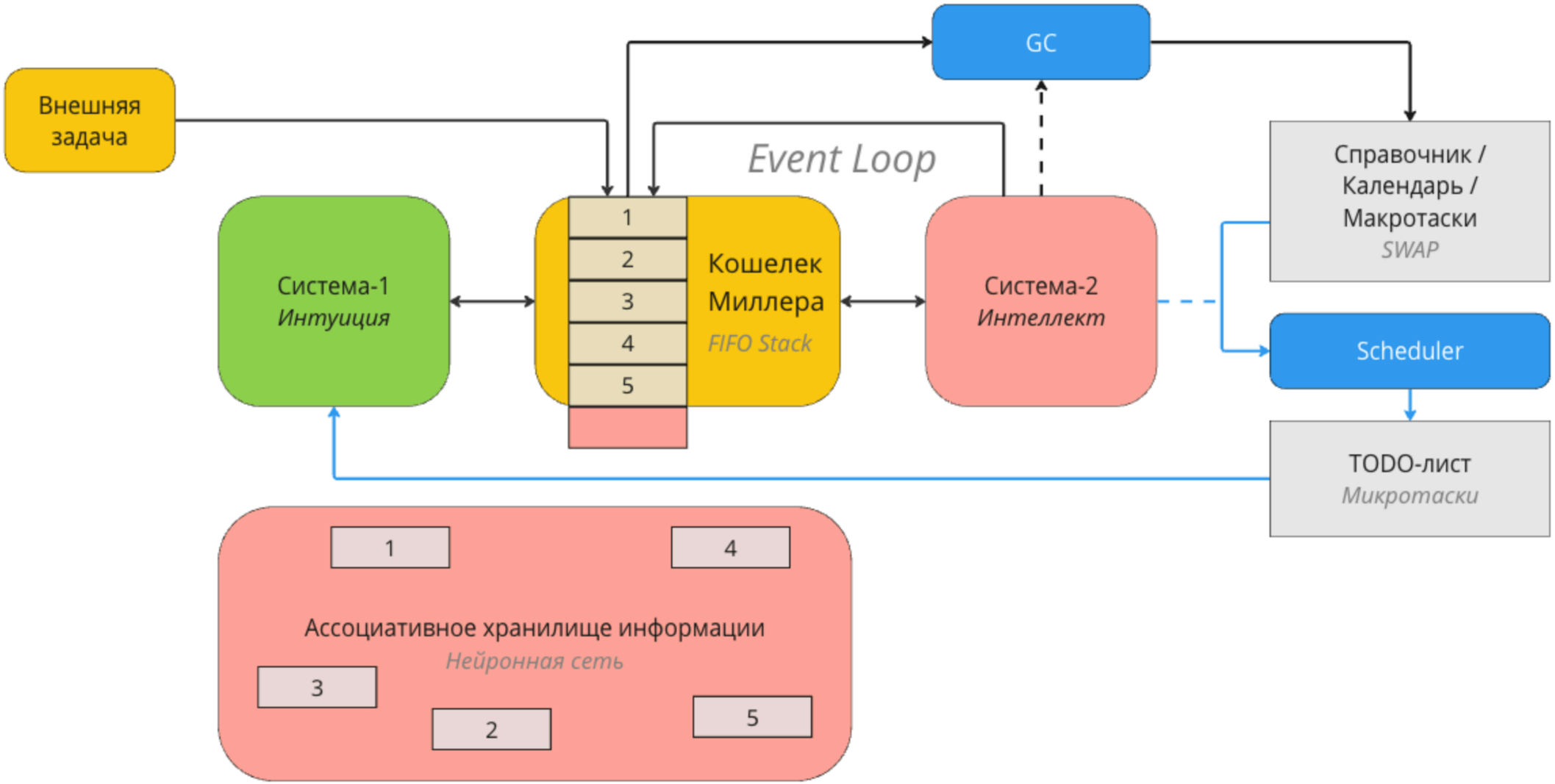


инструменты экономии мыслетоплева

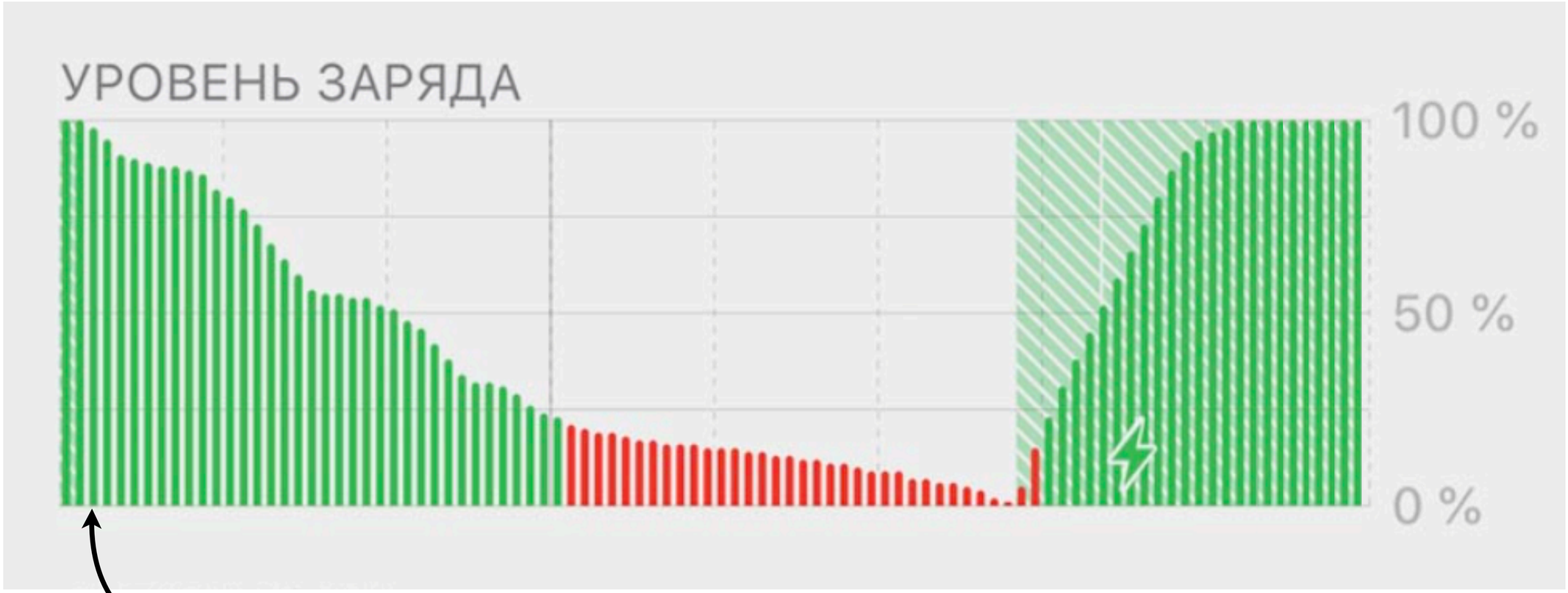
очистка контекста



краткосрочное планирование

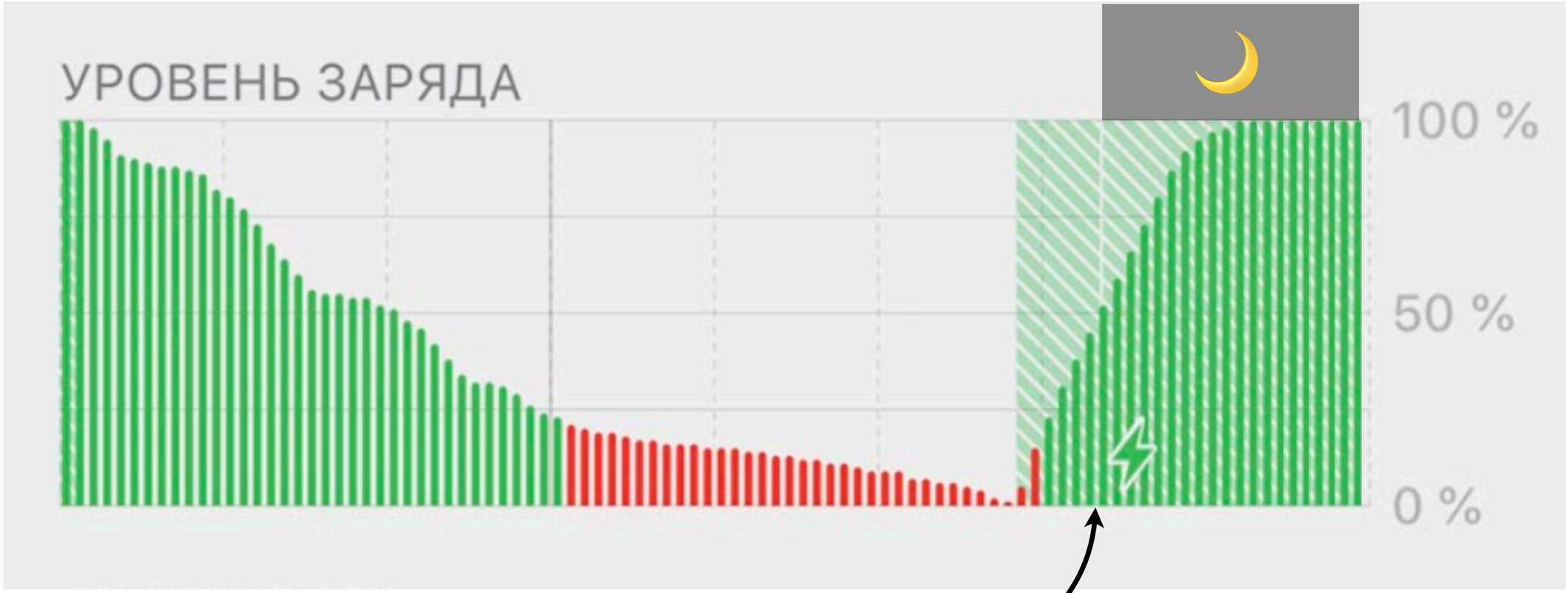


проглотить «Лягушку»



Самое продуктивное время

план на день



Тоже довольно удачное время

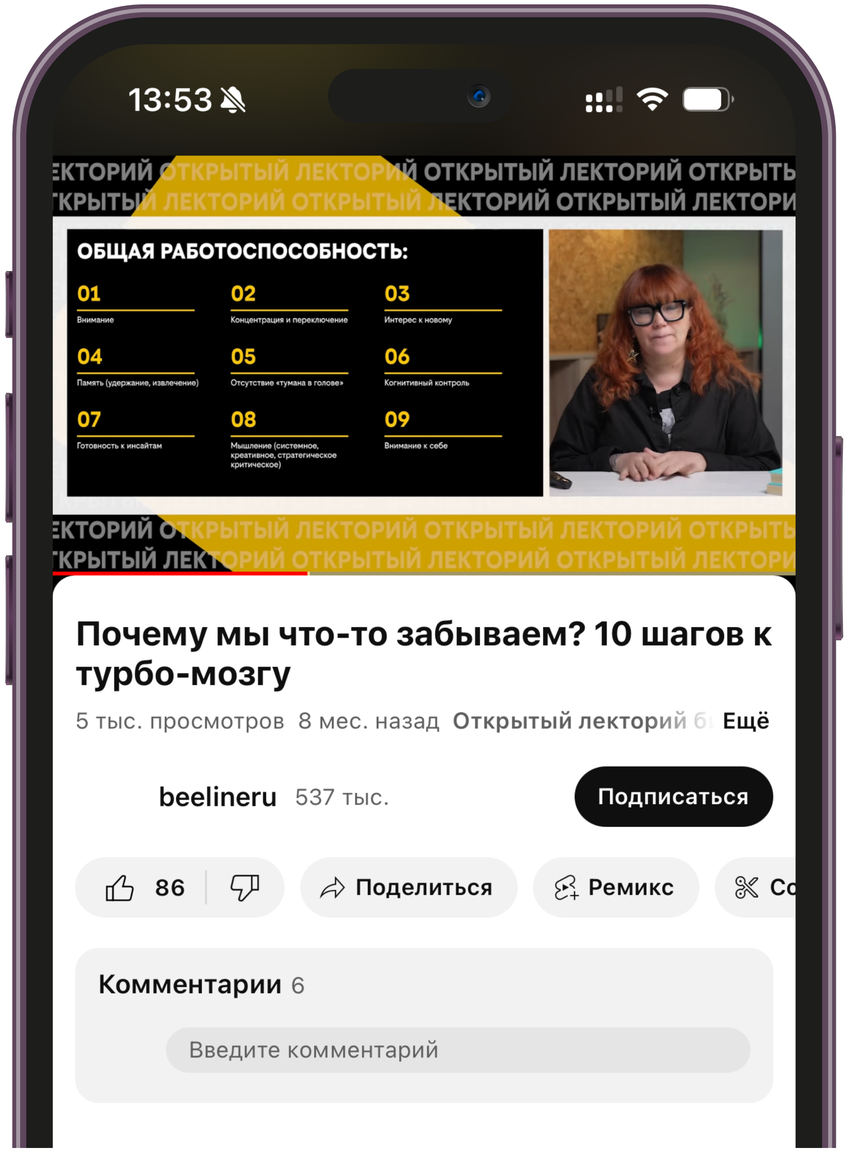
прочие инструменты эффективного расхода мыслетоплива

01 Долгосрочное планирование (фокусы)

02 Ежедневное обновление TODO-листов

03 Ультрадианные ритмы

04 Метаболизм



билайн
университет

Почему мы что-то забываем? 10 шагов к турбо-мозгу

визуализация архитектуры

зачем оно нужно?

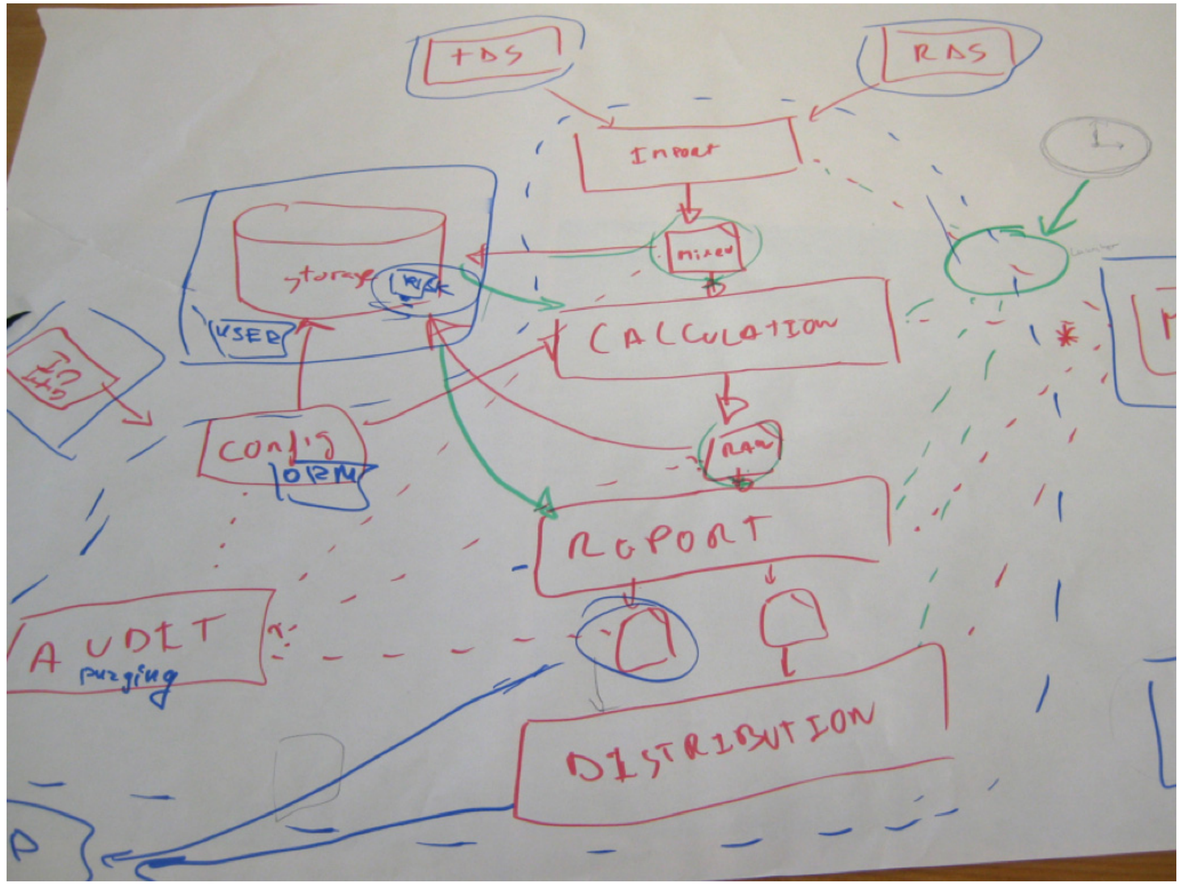
Компьютер работает на данных,
мозг на образах;

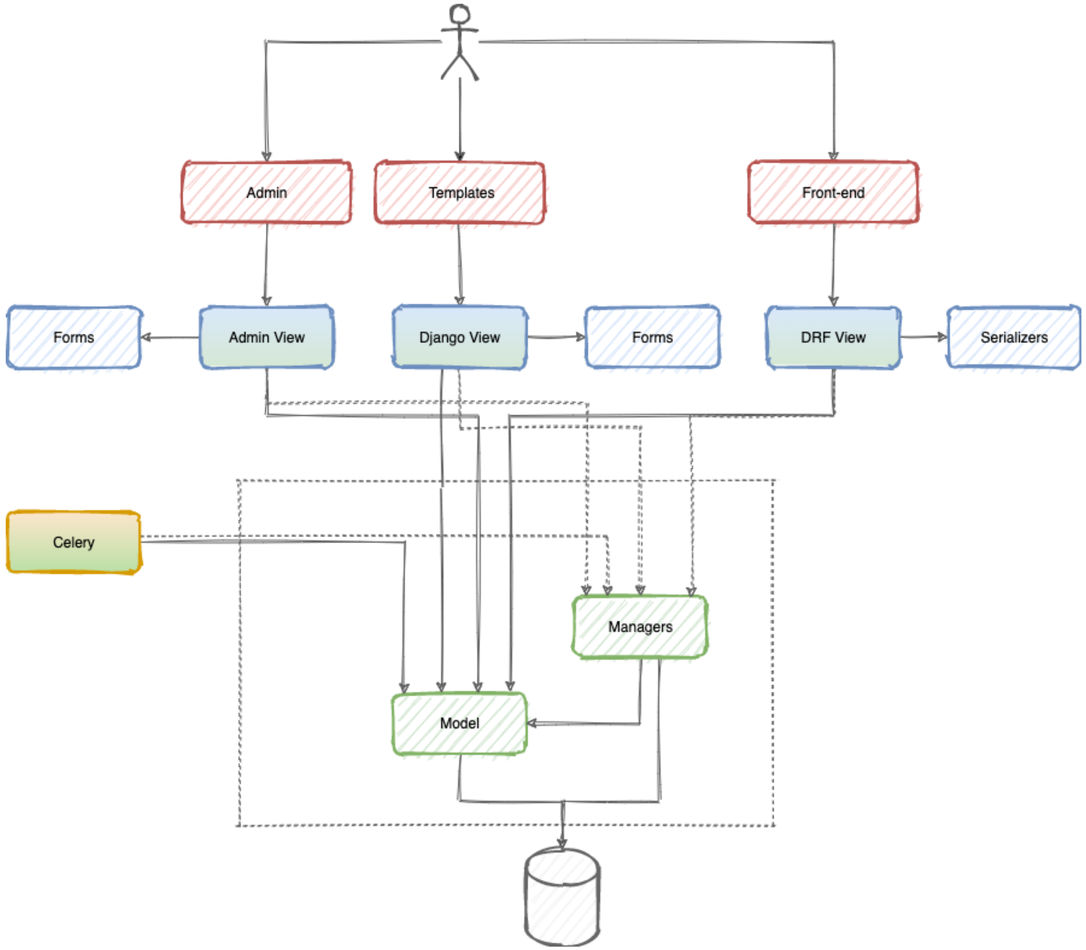
зачем оно нужно?

Компьютер работает на данных,
мозг на образах;

Нужно «сгружать» рабочую память
и быстро загружать;

отрисовка схем

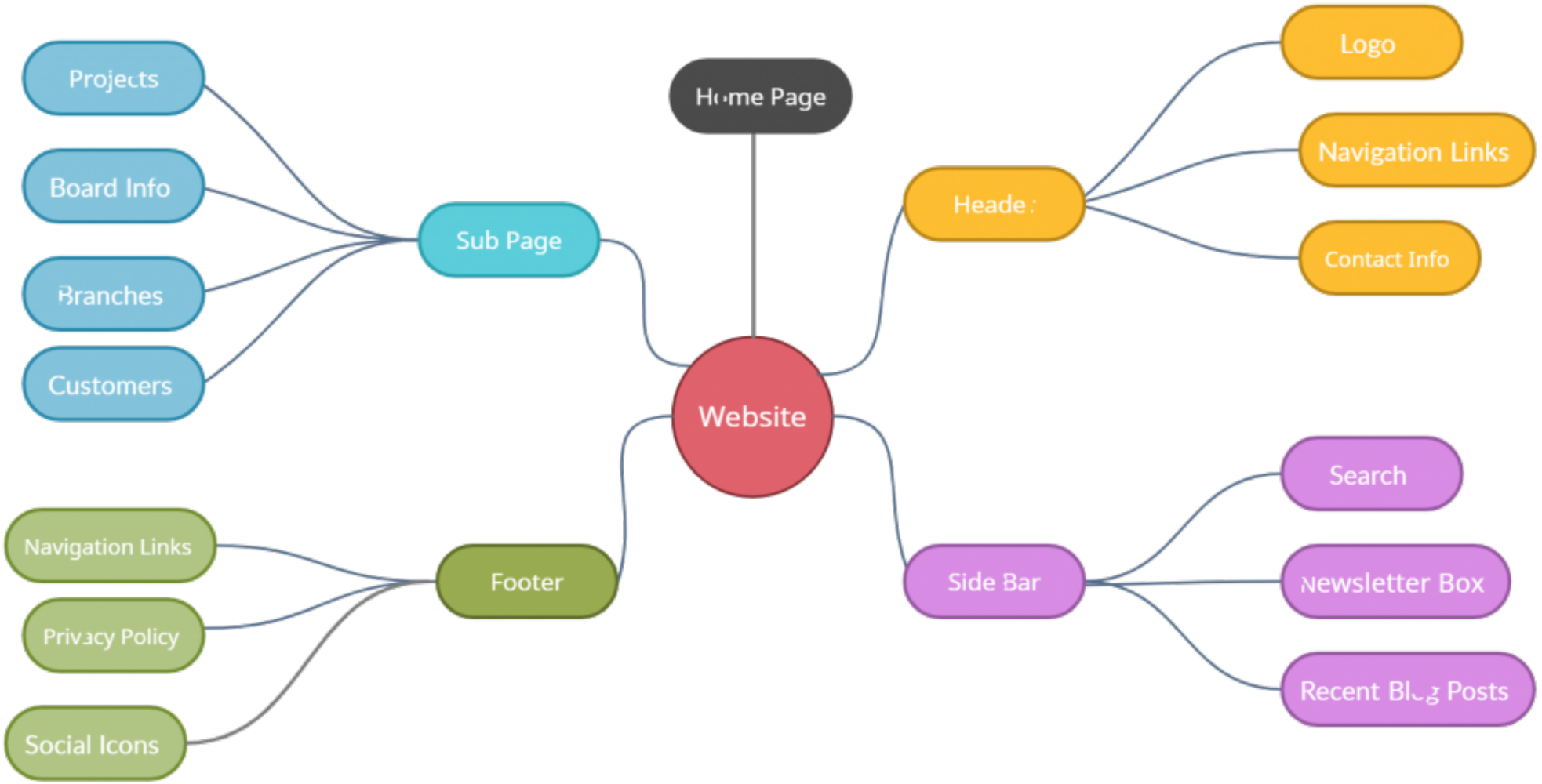




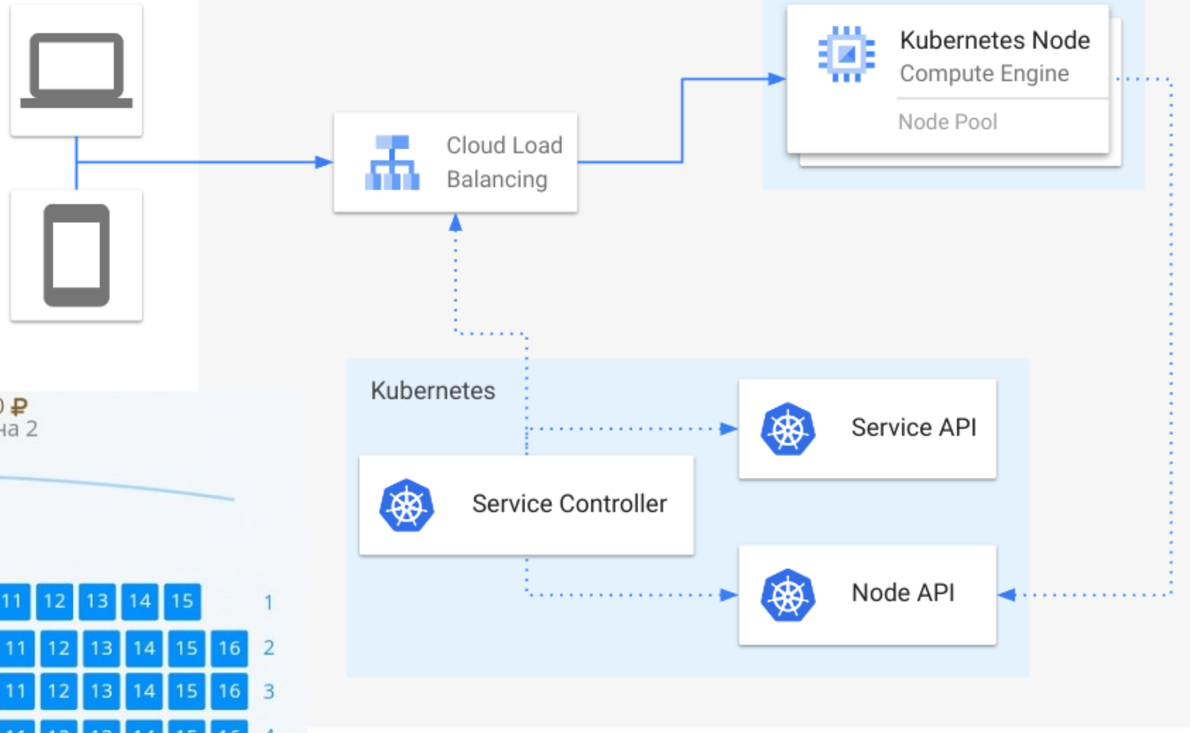
связь между схемой и принятием решений?

- Можно выделить минимальный набор пар “проблема - решение”;

Mind Map?


Projects
Board Info
Branches
Customers
Sub Page
Home Page
Header
Logo
Navigation Links
Contact Info
Website
Navigation Links
Privacy Policy
Social Icons
Footer
Side Bar
Search
Newsletter Box
Recent Blog Posts

проекции проблемных областей



Blackbox & Whitebox

01 Скрываем «самодостаточную часть схемы», оставляя только вход и выход;

Blackbox & Whitebox

01 Скрываем «самодостаточную часть схемы», оставляя только вход и выход;

02 В родительской схеме работаем с требованиями под BlackBox;

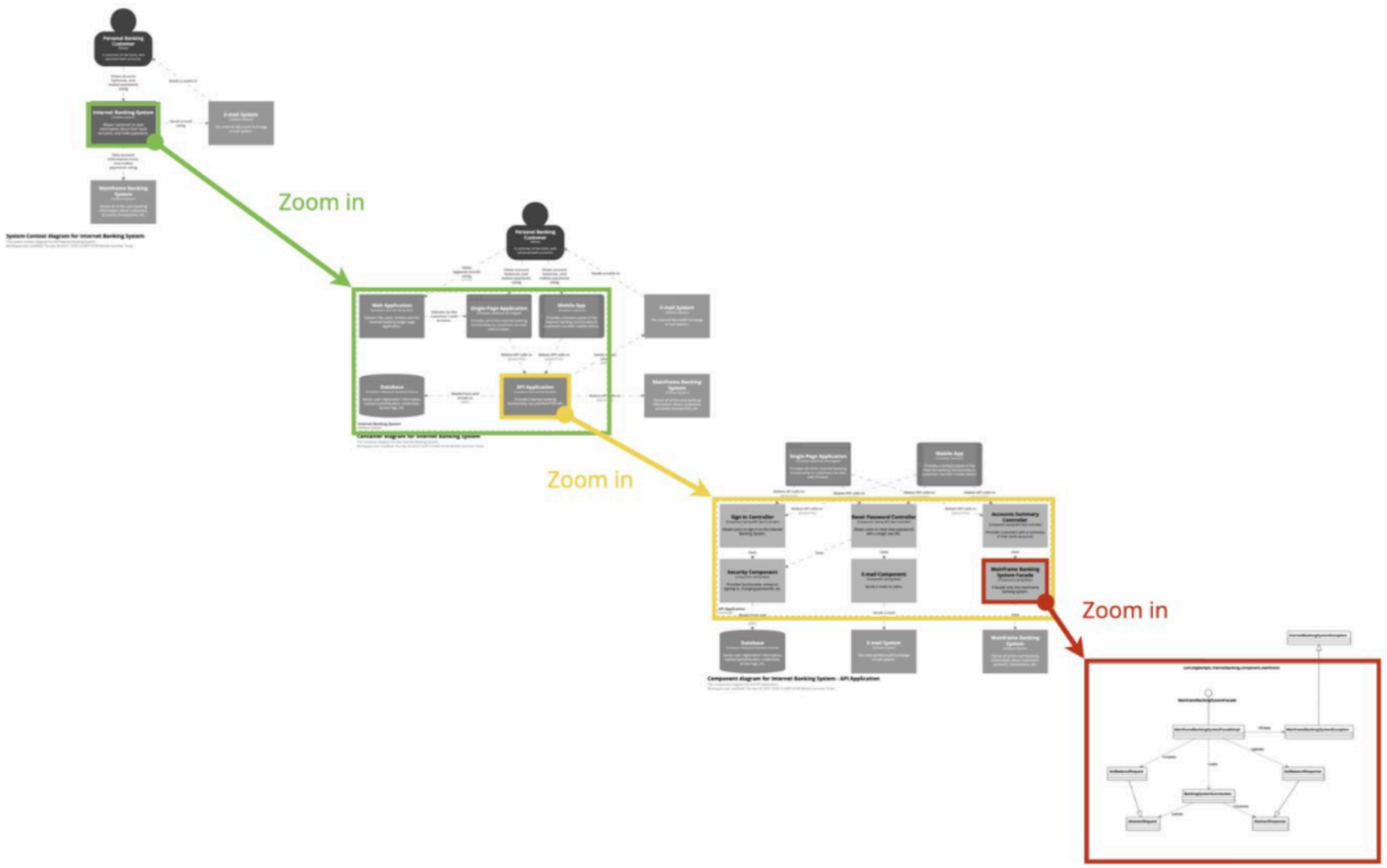
Blackbox & Whitebox

01 Скрываем «самодостаточную часть схемы», оставляя только вход и выход;

02 В родительской схеме работаем с требованиями под BlackBox;

03 В дочерней схеме исполняем требования как WhiteBox.

модель визуализации C4



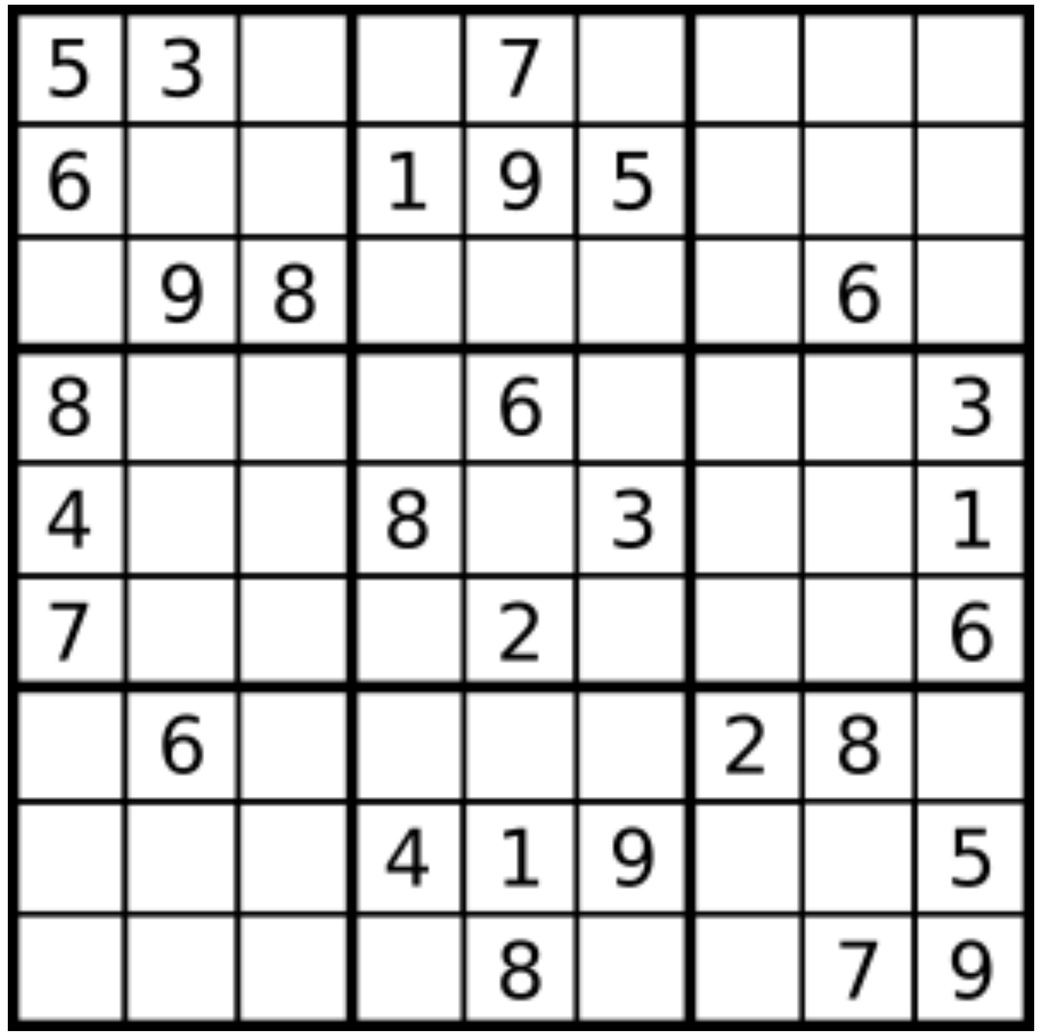
Level 1
Context

Level 2
Containers

Level 3
Components

Level 4
Code

архитектура как судоку



аналогии как архитектурный инструмент

- Переиспользование имеющихся нейронных связей

стратегическое планирование

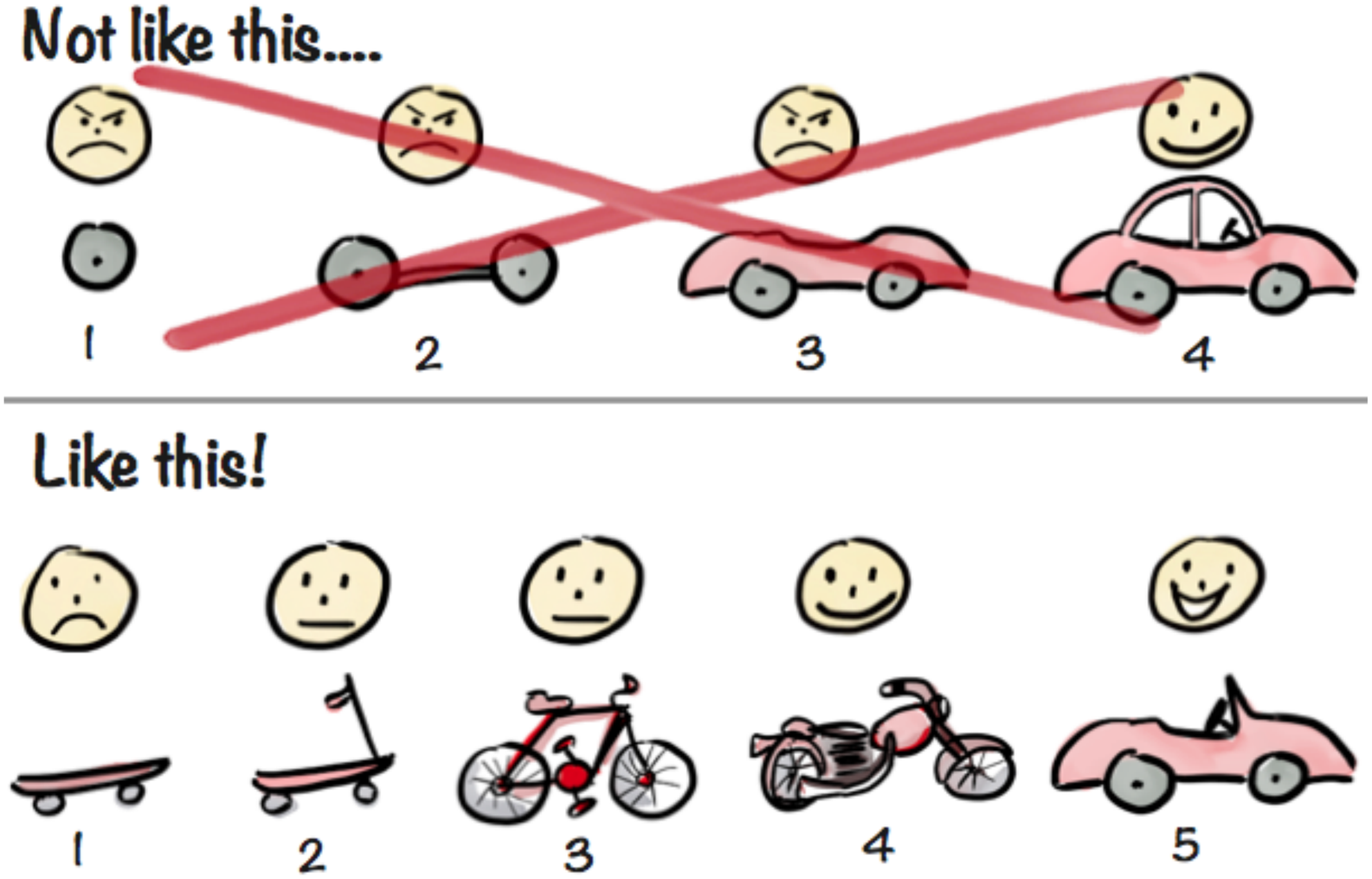
стратегическое планирование

01 Баланс архитектуры и бизнеса;

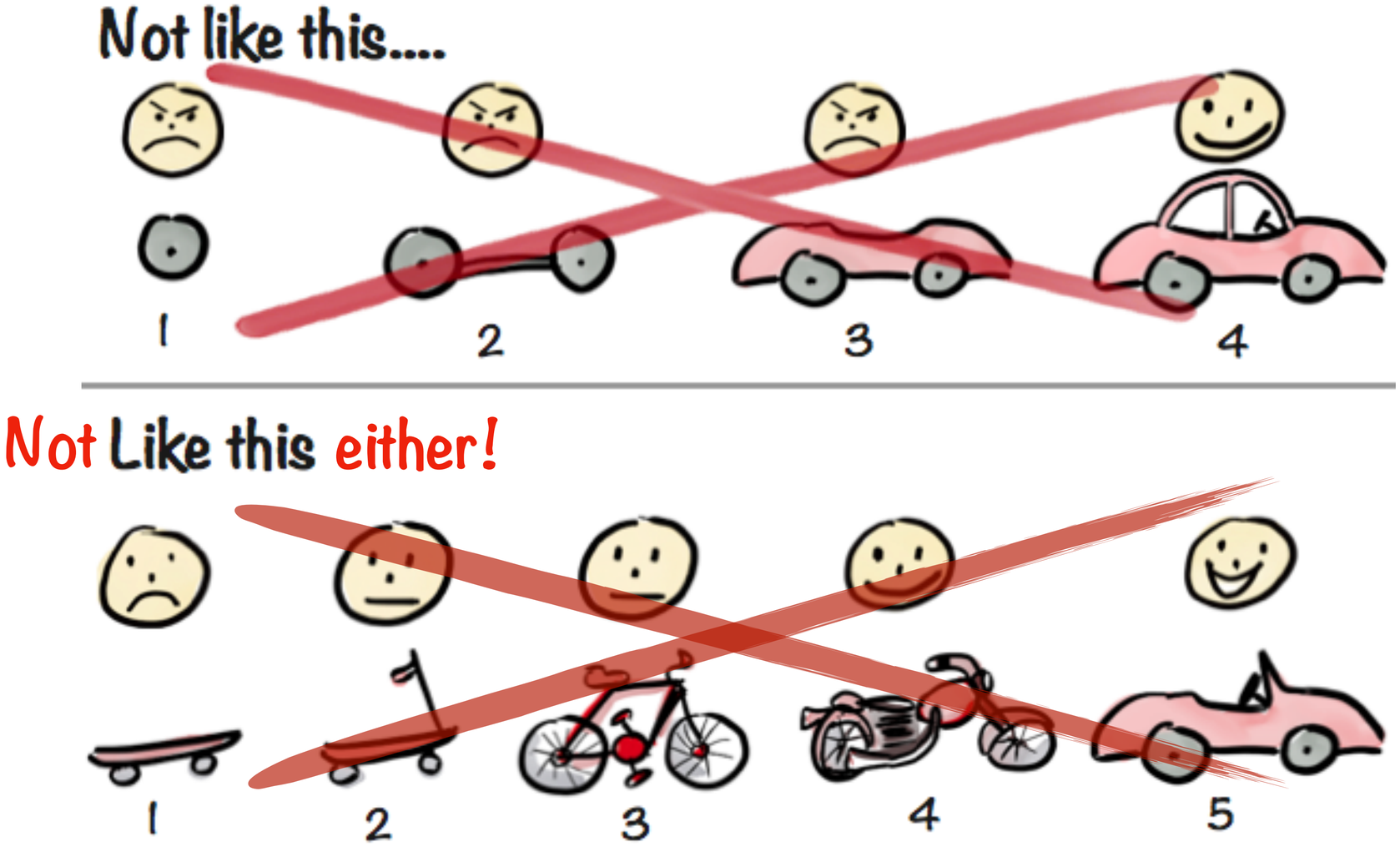
02 Добавляем в формулу время и деньги;

03 Приземляем на текущую архитектуру;

распространенная ошибка



распространенная ошибка



стратегическое планирование

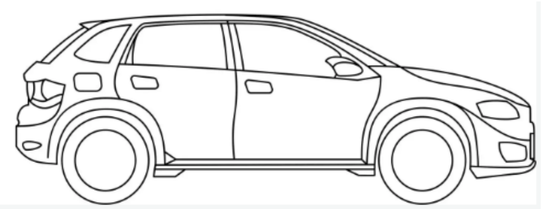
01 Реализуем MVP-приложение для одной задачи;

02 Обобщаем до универсальной платформы;

03 Profit;

горизонт планирования

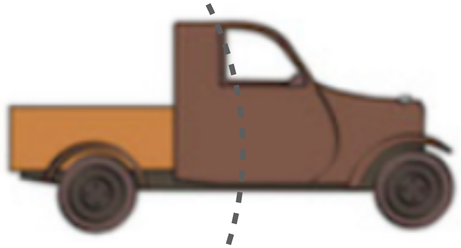
MVP



TO BE

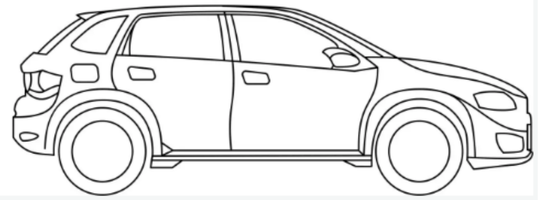
горизонт планирования

MVP



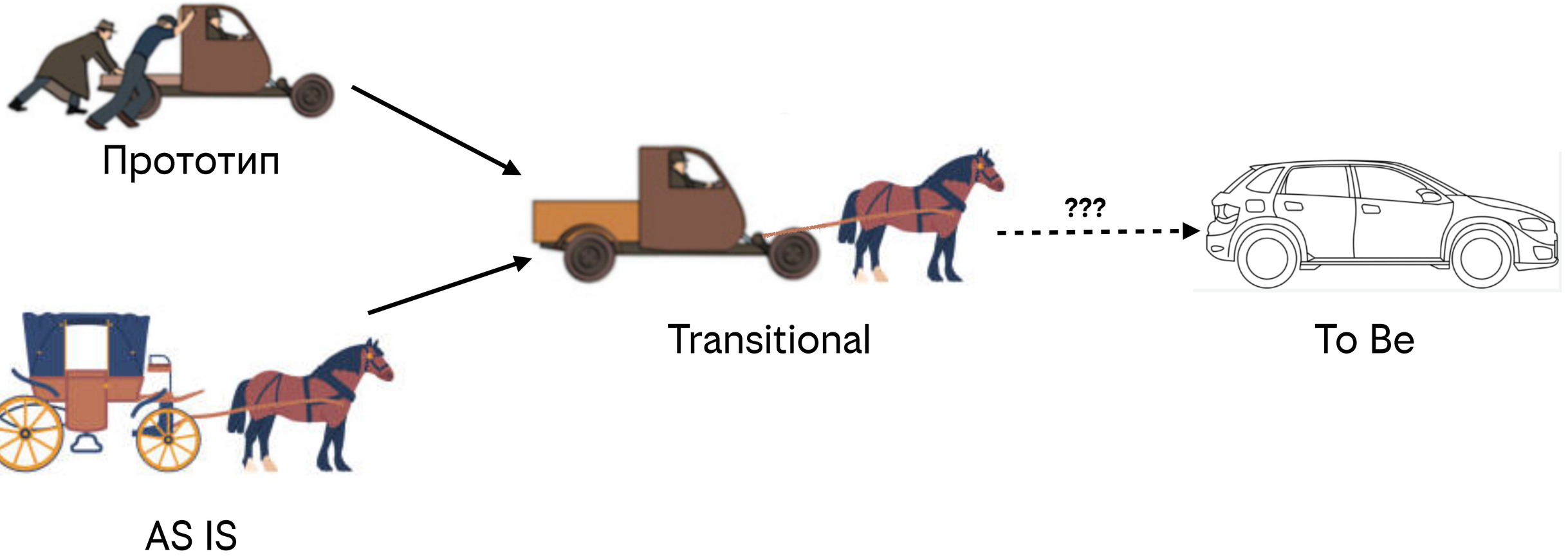
Increment

???



TO BE

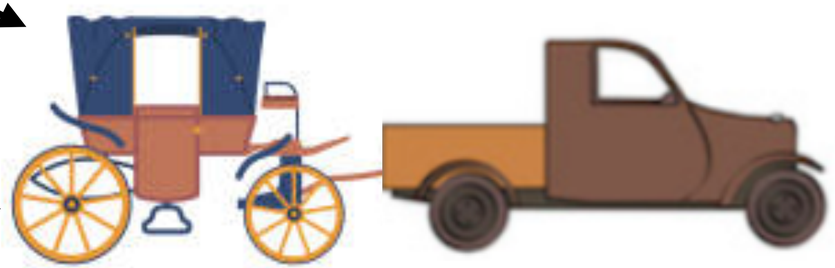
приземляем
на существующее решение



вытягиваем Legacy

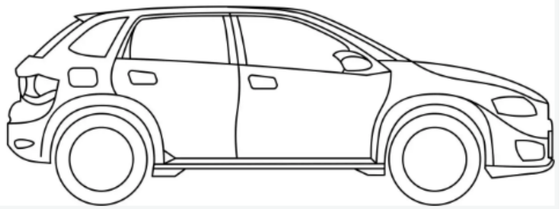
Прототип

AS IS



Transitional

???



To Be

стратегическое планирование

01 Все решения должны приближать нас к “To Be”;

стратегическое планирование

01 Все решения должны приближать нас к “To Be”;

02 Если требуется «крюк», его стоимость не должна быть высокой;

стратегическое планирование

01 Все решения должны приближать нас к “To Be”;

02 Если требуется «крюк», его стоимость не должна быть высокой;

03 Нельзя забывать про стоимость внедрения и сопровождения.

заключение

Выводы

01 Не обязательно идти в менеджмент и бросать код;

Выводы

01 Не обязательно идти в менеджмент и бросать код;

02 Любое решение в архитектуре - компромисс;

Выводы

01 Не обязательно идти в менеджмент и бросать код;

02 Любое решение в архитектуре - компромисс;

03 Для решения сложных задач важна личная эффективность;

Выводы

01 Не обязательно идти в менеджмент и бросать код;

02 Любое решение в архитектуре - компромисс;

03 Для решения сложных задач важна личная эффективность;

04 Чем больше паттернов и методологий знаешь
- тем больше у тебя возможностей;

Выводы

01 Не обязательно идти в менеджмент и бросать код;

02 Любое решение в архитектуре - компромисс;

03 Для решения сложных задач важна личная эффективность;

04 Чем больше паттернов и методологий знаешь
- тем больше у тебя возможностей;

05 Закладывайте архитектуру идеального решения на годы вперед, но планируйте разработку на короткий срок;

Выводы

01 Не обязательно идти в менеджмент и бросать код;

02 Любое решение в архитектуре - компромисс;

03 Для решения сложных задач важна личная эффективность;

04 Чем больше паттернов и методологий знаешь
- тем больше у тебя возможностей;

05 Закладывайте архитектуру идеального решения на годы
вперед, но планируйте разработку на короткий срок;

06 Принимать архитектурные вызовы
и постоянно развиваться - безумно интересно;

Спасибо за внимание!

Иван
Малюгин
Билайн