



Ринат Рысаев
Product Lead

Артефакт ведения бэклога по фреймворку Opportunity Solution Tree

Непрерывный поток проверки гипотез — классическая задача продакта, и фреймворков хватает. Но ни один не закрывает последнюю милю: на своих данных, в своей команде, со своими ограничениями каждый достраивает процесс сам. Ниже — проблемы, в которые мы упёрлись на практике, и решения, которые ускорили наш Discovery и сонаправили его с Delivery.

1. Discovery живёт отдельно от целей

Discovery не бьётся с целями и метриками (OKR, NSM): Double Diamond не говорит, откуда брать начало первого ромба. Гипотезы растут вразнобой, и те, что двигают разные метрики, сложно сравнить между собой.

2. Приоритизация наступает уже после потерь

Чтобы приоритизировать гипотезу, надо сперва заполнить её скор, а для этого — поресёрчить. Мы вкладываемся в каждую ещё до того, как решаем, стоит ли её вообще рассматривать. Приоритет приходит, когда затраты уже понесены.

3. Один тест хоронит гипотезу — а иногда и всю проблему

Гипотезу отвергают после первого же провального теста — ложноотрицательный вывод: провалилась реализация, а не идея. Хуже, когда одним тестом закрывают не гипотезу, а всю проблему под ней — и из беклога вылетает живая, неисчерпанная боль.

Мы взяли за основу Opportunity Solution Tree Терезы Топпес (Continuous Discovery Habits) и достроили его в трёх местах — по одному на каждую боль.

1. Дерево растёт от ключевой метрики

Корень — не абстрактный outcome, а ключевая метрика юнита. Мы декомпозируем её (условно: $\text{Оборот} = \text{Покупатели} \times \text{Частота} \times \text{Чек}$), и слагаемые становятся ветками. Гипотеза с рождения привязана к своей суб-метрике — сравнивать разное больше не нужно.

2. Приоритизация вшита в структуру, а не следует за скорингом

Приоритет задаёт форма дерева: метрика → сегменты → проблемы. Сегменты известны заранее и ранжируемы между собой (ёмкость, LTV, RFM), поэтому вес ветки виден из её адреса — до скоринга отдельных гипотез.

3. Тест, гипотеза и проблема — разные уровни, и мы их не схлопываем

Неудачный тест закрывает тест, а не гипотезу; отвергнутая гипотеза не отменяет проблему. Пока под проблемой есть непроверенные гипотезы, она живая — одна кривая реализация не хоронит ценную ставку.

Разберем на примере вымышленного продукта доставки

Шаг 1. Возьми цель, привязанную к продуктовой метрике.

Команда берёт цель юнита, выраженную не фичей, а метрикой. Одна метрика – один корень дерева.

→ «вырасти как бизнес» превращается в метрику Оборот (GMV).

Шаг 2. Задай амбицию – это контракт.

Определи, на сколько двигаешь метрику: $X + N\%$. Цифра приходит либо сверху, от потребности бизнеса, либо снизу, от возможностей и ресурса. В любом случае это контракт между «надо» и «можем» – зафиксируй его.

→ Оборот +15% за квартал.

Шаг 3. Декомпозируй метрику.

Декомпозируй так, чтобы сабметрики честно собирались обратно в метрику. Каждая сабметрика становится веткой-рычагом. На этом шаге ты ещё не знаешь, какая фича закроет цель, – и это нормально.

→ $\text{Оборот} = \text{Покупатели} \times \text{Частота} \times \text{Средний чек}$.

Шаг 4. Построй дерево вниз от рычагов.

Под каждым рычагом – сегменты, под ними проблемы, под проблемами гипотезы, под гипотезами тесты. Сегменты обычно известны заранее. Если у тебя уже есть готовый эпик или решение, которым ты планируешь двигать метрику, – не протаскивай его через все слои, а повесь на уровень того рычага, к которому он относится.

→ под Частотой – сегменты Регулярные / Остывающие / Скидочные

Шаг 5. Приоритизируй ветки.

Приоритет задаёт сама структура: сегменты ранжируемы заранее – по ёмкости, LTV, RFM, – поэтому вес ветки виден ещё до скоринга отдельных гипотез. Ты решаешь, в какую ветку спускаться, а не считаешь скор для всех подряд.

→ берём Остывающих: их ещё можно удержать, в отличие от давно ушедших.

Шаг 6. Работай с одной веткой до листа.

Спускайся по выбранной ветке: сегмент → проблема → гипотеза → тест. Тесты ставь по возрастанию стоимости: сначала дешёвое опровержение, потом проверка ценности, потом проверка реализации.

→ Долго собирать заказ → умный список → fake door → concierge → A/B.

Шаг 7. Не схлопывай уровни — это главное правило.

Заваленный тест $\neq$ заваленная гипотеза. Заваленная гипотеза $\neq$ отсутствие проблемы.

Договорись с командой заранее: сколько провалов и какими способами закрывают гипотезу, какие свидетельства закрывают проблему. Без этого договора ты будешь хоронить ценное на ложноотрицательных.

Шаг 8. Возвращай знание обратно в дерево.

После каждого теста добавляй learning в узел и обновляй данные гипотезы и проблемы — статус, уверенность, приоритет. Дерево живое: оно копит, что проверено и с каким исходом, а не просто хранит список идей.

Бонус. Держи дерево как граф.

Весь беклог удобно вести атомарными markdown-заметками, где метрики, проблемы, гипотезы и тесты — отдельные сущности со ссылками друг на друга. Тогда дерево превращается в единый граф, который можно запрашивать — и поверх которого хорошо ложится ИИ.

