# ПРИКЛАДНАЯ ПРОБЛЕМОЛОГИЯ.

*В.Н. Болотовский,*
*Санкт-Петербург*

## 1. Проблемные ситуации.

ТРИЗ возникла и разрабатывалась, как инструмент для решения *Изобретательских задач,* что отражено даже в её названии. Со временем, значение ТРИЗ было переосмыслено, и в настоящий момент её можно рассматривать, как комплексный инструмент, для *решения проблем* вообще, а решение изобретательских задач, - это лишь один из этапов решения проблемы.

Ввиду того, что именно решение изобретательских задач является наиболее проработанным, развитым и формализованным инструментом, многие, кто владеют (или думают, что владеют) инструментами ТРИЗ, пытаются их применять, только на этапе решения таких задач. Реальная жизнь почти никогда не предоставляет нам изобретательские задачи в чистом виде. Г. С. Альтшуллер рассматривал так называемые *Изобретательские ситуации,* которые необходимо преобразовать в задачу. Для этого в ТРИЗ существуют определенные правила и инструменты.

Но в повседневной жизни человек сталкивается с ситуациями, которые в подавляющем большинстве случаев, совсем не являются изобретательскими. Они являются просто ***проблемными.*** Причем проблемы эти могут быть совершенно любой природы, не обязательно технические. И что интересно: даже обладая определенными навыками в ТРИЗ, человек не всегда берется за разрешение изобретательских ситуаций, но большинство людей ежедневно самоотверженно стремятся решить ситуации проблемные, не обладая никакими алгоритмами их решения. В быту это называются «я решаю свои проблемы».

Между тем, четкой, общедоступной методики решения проблем не существует. Чтоб решать проблемы методами ТРИЗ, необходимо научится переводить проблемные ситуации в изобретательские задачи. Причем не индивидуально, каждую ситуацию, а иметь общий алгоритм, который позволит свести проблемную ситуацию любой природы к виду, в котором она будет разрешима существующими инструментами ТРИЗ.

Это значительно расширит область применения ТРИЗ, поскольку проблемных ситуаций у обычного человека неизмеримо больше, чем изобретательских у специалистов.

В данной работе представлена попытка проанализировать проблемные ситуации в общем виде и представить некоторые алгоритмы приведения их к виду, разрешимому в ТРИЗ.

### *1.1. От решения изобретательских задач – к решению проблем.*

Решением изобретательской задачи является устранение технического противоречия в системе, путем разрешения противоречия физического, которое его вызывает. Но для большинства потребителей ТРИЗ-технологий, физические,

и технические противоречия «остаются за кадром», как порой и сама система.

Если рассматривать применение технологий ТРИЗ для решения проблем, то возникает закономерный вопрос: а что считать решением проблемы? И как оно связано с изобретательскими задачами?

Обычно, на интуитивном уровне, под решением проблемы понимают переход к ситуации, в которой проблема отсутствует. Это влечет за собой второй вопрос – а что же является проблемой, и какая ситуация считается проблемной? Ответив на этот вопрос, можно определить, что именно должно отсутствовать в конечной ситуации, чтобы проблему можно было считать решенной.

Тогда можно будет поставить задачу примерно следующим образом: нечто (икс) не должно быть, так как его отсутствие и есть решение проблемы, и нечто быть должно, поскольку оно есть (ведь изначально ситуация проблемная). Это уже вполне ТРИЗовская постановка задачи, и разрешается она известными инструментами.

Осталось определить элемент, который делает проблему проблемой.

### *1.2.Бытовое и формальное определение проблемы.*

В бытовом массовом сознании проблемой считается некая ситуация, которая по тем или иным причинам нас не устраивает. То есть «что-то не так». Соответственно, ситуация, когда это «что-то» будет «так» является решением проблемы.

Заменим бытовые термины абстрактными: пусть «что-то» будет «А», а «так» будет «Б». Тогда проблему можно сформулировать следующим образом:

А должно [быть] Б

Поскольку в общем случае «Б» может быть действием (глаголом), то слово «быть» забрано в скобки.

Например: из крана течет вода, расходуясь понапрасну. Это нас не устраивает. «Вода» должна «не течь». Чтобы вода не текла, достаточно повернуть кран. Является ли это проблемой? По-видимому, нет. Значит «А должно быть Б» недостаточно для существования проблемы. Проблема в нашем примере возникнет, если кран не перекрывает воду, или до него некому дотянуться, или перекрытие крана может привести к еще худшим последствиям, и т. п. Иными словами, если А **нельзя** по каким-то причинам сделать Б.

В повседневной жизни второе условие очень часто существует неявно. И большинство людей принимают за проблему именно первую часть определения, считая, что проблема заключается именно в том, что А должно быть Б. Хотя на самом деле проблема возникает, когда выполняются оба условия:

А должно [быть] Б
А нельзя по каким-то причинам сделать Б

Приведем примеры таких «неявных» ситуаций.

«У меня проблема: я не успеваю на концерт»
«Проблема в том, что в холодильнике пусто»
«У меня нет ключа, и мне не попасть домой»
«Встать в 7 утра – для меня проблема!»
«Заставить персонал запомнить пароли – это большая проблема!»

В приведенных бытовых примерах указано только одно условие, характеризующее наличие проблемы, но авторы этих высказываний не сомневаются, что проблема описана полностью. Более того, данные проблемы многие люди тут же пытаются решать, так и не сформулировав условия. Надо ли удивляться, что как-нибудь внятно их решить не удастся?

### *1.3. Что должна содержать проблема?*

Рассмотрим приведенные примеры в полной формулировке, и посмотрим, как это будет способствовать решению.

**«У меня проблема: я не успеваю на концерт»**

Скорее всего, рядом с человеком, который это говорит, находятся другие люди. Они, видимо, тоже на этот концерт успеть не смогут, но проблемой это не считают. Почему? Потому что им там быть не нужно.

Поэтому сама проблема звучит так:
«Я должен быть на концерте в такое-то время, но я не могу быть на концерте в такое-то время».

У остальных нет первого условия, поэтому нет и самой проблемы.

**«Проблема в том, что в холодильнике пусто»**

Представьте торговый зал, где выставлены холодильники. В них тоже пусто, но никто проблемой это не считает. Значит, дело в другом. Наверное, под «пусто в холодильнике» понимается отсутствие продуктов. Причем их отсутствие не «вообще», а в квартире. В обычной жизни эта ситуация легко разрешима при помощи похода в магазин, и проблемой не является.

Если, все же данная ситуация заявлена, как проблема, то возможны такие варианты:
- прямо сейчас придут гости (или уже в прихожей)
- есть срочные дела, не дающие сходить в магазин
- нет денег
- магазин не работает (например, ночью), или магазин очень далеко (в соседнем селе).

В первых двух случаях проблему можно сформулировать так:
«Я должен пойти за продуктами, чтобы дома была еда, но
Я не должен идти за продуктами, чтобы (не оставлять гостей)/(делать срочное дело)»

В других двух случаях возможна такая формулировка:
«Я должен пойти за продуктами, чтобы дома была еда, но
Я не могу пойти в магазин, потому что нет денег/нет магазина»

**«У меня нет ключа, и мне не попасть домой»**

Опять-таки, множество людей не имеют при себе ключей (соответственно не могут попасть в дом), но проблемой это не считают. Либо им туда не нужно

сейчас, либо дома кто-то есть. Значит, проблему можно сформулировать так:
«Я должен войти в квартиру, чтобы отдохнуть (поесть, переодеться и т д), но
Я не могу войти в квартиру потому что, дверь закрыта».

**«Встать в 7 утра – для меня проблема!»**

Если оставить проблему в такой формулировке – то она имеет очевидное решение: «не надо вставать». Но ведь зачем-то человек просыпается рано утром, несмотря на все неудобства. Скорее всего, для чего-то очень важного.

Вот пример формулировки:

«Я должен спать дальше, чтобы мне было комфортно, но
Я не должен спать дальше, чтобы присутствовать на конференции».

**«Заставить персонал запомнить пароли – это большая проблема!»**

Как и в предыдущем примере, проблема легко решаема: пустые пароли или что-то типа 12345 у всех. (Так, кстати многие и поступают). Второе решение – стикеры на мониторе. Если это действительно решает проблему, то пароли используются просто потому, что «так положено», но никто не понимает зачем. Если же это серьёзная организация, желающая сохранить свою информацию, то такое решение не подходит. Соответственно и сама проблема звучит по-другому:

«Пароль должен быть простым, чтобы пользователь его запомнил, но
Пароль не должен быть простым, чтобы сохранить информацию»
Или:
«Пользователь должен всегда знать свой пароль, чтобы войти в систему, но
Пользователь не может всегда знать свой пароль потому, что пароль длинный, сложный, и часто меняется».

Из приведенных примеров видна следующая закономерность: каждую проблему можно сформулировать в одном из двух видов:

1) A должно [быть] B, чтобы C, но
   A не может [быть] B, потому что D.

2) A должно [быть] B, чтобы C, но
   A не должно [быть] B, чтобы D.

То есть, во-первых, каждая проблема должна иметь два «плеча». Во-вторых, первое плечо одинаково, а второе может быть двух видов.

## 2. Состав проблемы: два плеча.

В чем же принципиальное отличие этих двух видов второго плеча?

В обоих случаях, сама суть проблемы, сформулирована в первом плече. Второе плечо описывает, невозможность «лобового» решения. В первом случае, такое решение невозможно, потому, что на то есть некоторые **причины.** Во втором случае, решение невозможно, потому что оно вызовет нежелательные **следствия**.

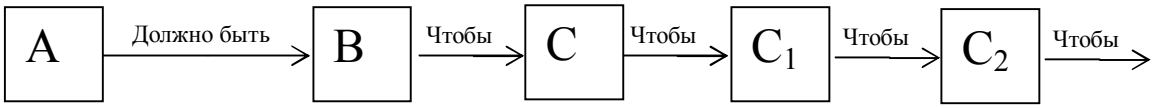
Интересно отметить, что первое плечо всегда имеет следственную природу. Рассмотрим все три ветви проблемы подробнее.

### *2.1.Конечная цель решения проблемы.*

Итак, первое плечо: А должно [быть] В, чтобы С. Все вроде бы просто. Но возникает вопрос: а зачем С?

Очевидно, С влечет за собой некое $C_1$, которое нам и необходимо. А необходимо оно потому, что в свою очередь влечет некое $C_2$, и так далее. В конце концов мы приходим к результату, ради которого, собственно А и должно быть В.

Таким образом, мы получаем следственную цепочку:




При формировании такой цепочки, необходимо помнить два момента:

Во-первых, каждое последующее звено цепочки должно непосредственно следовать из предыдущего. Например, из кипячения воды НЕ следует отсутствие в ней микробов. Из него следует повышение температуры воды, из которого следует разрушение структуры белков, из которого уже следует уничтожение микроорганизмов.

Во-вторых, необходимо «вовремя остановиться». Теоретически, следственную цепочку можно продолжать бесконечно, поскольку из каждого звена что-нибудь, да следует. Но необходимо понимать цели, которые мы преследуем, решая проблему. С другой стороны, нельзя обрывать цепочку, полагая, что дальнейшие шаги очевидны.

Типичный пример, когда цепочка обрывается на звене «чтобы было много денег», предполагая, что зачем нужно много денег, и так понятно. Хотя большинство из тех, кто хочет «много денег», очень слабо представляют, зачем именно.

Другой, более технический пример: информационная безопасность. Очень часто главной и конечной целью ИБ считают ограничение доступа определенным лицам к определенным ресурсам. Но ограничение доступа само по себе не является ни опасным, ни безопасным. И с обратной стороны, некоторые «хакеры», желая кому-то нанести вред, взламывают пароль, и получают доступ, например, к электронной почте. При этом они считают факт доступа к чужому ящику самодостаточной опасностью, хотя опасны действия, которые они могут предпринять, читая чужую почту, а не сам взлом. В почте может оказаться спам и пустая болтовня.

Обратим внимание на особенность следственной цепочки: она предназначена для достижения последнего звена (результата), путем перевода А в состояние В. Но ведь по логике, достичь результата можно из **любого звена цепочки**, а не только из первого. Если удается реализовать какое-либо $C_i$ тем или иным способом, **то цепочка сама приведет нас к результату**, причем в этом случае переводить А в В нет необходимости. Но с другой стороны, именно невозможность перевести А в В и составляет осязаемую проблему.

Итак, построив следственную цепочку, от проблемного звена до результата, мы

имеем возможность поставить задачу достижения результата, как реализацию любого из звеньев цепочки. При этом задача может измениться до неузнаваемости, и кроме того, «потерять» проблемную составляющую.

***Пример.***

Проблема из предыдущего раздела: «Я должен пойти за продуктами, чтобы дома была еда, но

Я не должен идти за продуктами, чтобы (не оставлять гостей)/(делать срочное дело)»

Построим первое плечо проблемы.

Я должен *пойти в магазин*, чтобы *купить продукты*, чтобы *дома были продукты*, чтобы *накрыть на стол*, чтобы *сесть праздновать с гостями*, чтобы *отдыхать/разговаривать* чтобы *остались приятные воспоминания*, чтобы *укрепились дружеские отношения*, чтобы …

На этом можно остановиться. Собственно задача похода в магазин – не поссориться в гостями. В магазин я пойти не могу, но к этому же результату приведет решение следующих задач:

- Как *купить продукты*;
- Как сделать, чтоб *дома были продукты;*
- Как *накрыть на стол;*
- Как *сесть праздновать с гостями;*
- Как *оставить приятные воспоминания;*
- Как *укрепить дружеские отношения*;

Видно, что сесть праздновать можно в ближайшем кафе, что приведет к тому же результату, но избавит от необходимости идти в магазин.

### *2.2.Последствия решения*

Зеркальным отражением рассмотренной цепочки является следственный вариант второго плеча проблемы. В этом случае, изменение, вызывающее цепочку, приводящую к желаемому результату, также вызывает другую следственную цепочку, приводящую к результату нежелательному или недопустимому.

Для лучшего осознания сути проблемы, эту цепочку необходимо построить также тщательно. Из каждого звена должно непосредственно следовать следующее, и также необходимо добраться до нежелательного результата, чтобы понять, а с чем мы собственно боремся.

Поскольку цепочка зеркальная, то строится она «со знаком минус»:

$$A \xrightarrow{\text{НЕ должно быть}} B \xrightarrow{\text{Чтобы НЕ}} D \xrightarrow{\text{Чтобы НЕ}} D_1 \xrightarrow{\text{Чтобы НЕ}} D_2$$

Рассмотрим ту же проблему: «Я должен пойти за продуктами, чтобы дома была еда, но

Я не должен идти за продуктами, чтобы (не оставлять гостей)/(делать срочное дело)»

Я не должен *идти за продуктами*, чтобы не *оставлять гостей*, чтобы *не тратить (их) время*, чтобы не *осталось мало времени на общение*, чтобы не *осталось чувство напрасно потраченного вечера*, чтобы не *сочли меня негостеприимным*, чтобы *не пропало их желание приходить ко мне в гости*, чтобы не *сидеть вечерами одному*.

Возможно, в данном примере получилось несколько утрировано. Некоторые могут остановиться на звене «чтобы не сочли меня негостеприимным». Все зависит от того, какой результат считать нежелательным. Но на этом примере в дальнейшем будут видны некоторые закономерности.

Основная особенность этого вида цепочек такова: если разрушить **любое из звеньев**, то нежелательный результат достигнут не будет. Следовательно, задачу можно сформулировать так: как не допустить реализации какого-либо звена.

В данном примере из цепочки вытекают такие формулировки:

- *Как не оставить гостей*
- *Как не потратить их время*
- *Как оставить много времени на общение*
- *Как не оставить чувство напрасно потраченного вечера*
- *Как не прослыть негостеприимным*
- *Как сохранить их желание приходить ко мне в гости*
- *Как не остаться вечерами одному*

Вот тут как раз и проявляется разница, где нужно остановиться. Если конечным нежелательным результатом считать возможное одиночество, то вопрос можно решить, заведя новых друзей. Если же дороги именно эти друзья, то такое решение не годится. Появилось оно потому, что мы «промахнулись» мимо нежелательного результата на один шаг. Если убрать последнее звено цепочки, то такое решение и не появится.

### *2.3. Возникновение проблемы.*

Третья разновидность цепочки – причинная. Предыдущие две были следственными.

Причинная разновидность второго плеча проблемы возникает тогда, когда события, предшествующие проблеме, приводят к невозможности её решения «в лоб». Иными словами перевод А в состояние В не влечет вредные последствия, а физически невозможен. Причины этой невозможности как раз и отражает причинная цепочка.

Строится она следующим образом:


Какие особенности имеет причинная цепочка?

Во-первых, устранив **любое из звеньев**, мы тем самым устраняем все последующие звенья, включая невозможность для А быть В. Таким образом,

задача по решению проблемы может быть трансформирована в задачу по устранению какого-либо звена.

Пример.

Возьмём проблему из первого раздела:

«Пользователь должен всегда знать свой пароль, чтобы войти в систему, но Пользователь не может всегда знать свой пароль, потому что пароль длинный, сложный, и часто меняется».

Построим причинную цепочку:

Пользователь не может *всегда знать свой пароль*, потому что *пароль длинный, сложный, и часто меняется* потому что *сисадмин так запрограммировал систему,* потому что *отдел ИБ составил такую инструкцию,* потому что *легкие пароли может взломать злоумышленник,* потому что *он мечтает украсть информацию,* потому что *обладая ею он может получить много денег* (оставим подробности), потому что *информация позволяет получить эти деньги...*

В отличие от следственных цепочек, в причинной нет необходимости «вовремя остановиться». Либо это происходит естественным образом (например, на фразе «потому что Земля притягивает»), либо, исходя из здравого смысла.

Какие задачи можно сформулировать на примере приведенной цепочки? Я перечислю их чисто механически, не вдаваясь в осмысленность:

- Как сделать пароль простым и легким
- Как сисадмину не программировать жесткие правила для паролей
- Как отделу ИБ не писать подобных инструкций
- Как сделать легкий пароль защищенным от взлома
- Как сделать информацию неинтересной для хакеров
- Как сделать так, чтоб даже обладая информацией, нельзя было получить много денег
- Как сделать так, чтобы информация не позволяла получить деньги

Несмотря на то, что многие формулировки кажутся одинаковыми, все это разные задачи. И, кроме того, каждая последующая является решением для предыдущей.

Вторая особенность причинной цепочки заключается в следующем.

Причинная цепочка приводит к тому, что А не может быть В. Это приведёт к тому, что не будет С, следовательно не будет $C_1$ и т. д. Таким образом, причинная цепочка является причиной не только для проблемного звена, но и для **любого звена из следственной цепочки первого плеча**.

Обратное тоже верно: следственная цепочка первого плеча является следствием для **любого плеча причинной цепочки**.

Отсюда следует один важный вывод: следственная цепочка первого плеча и причинная цепочка второго плеча, соединяясь в проблемном звене, **образуют единую причинно-следственную цепочку**, любое звено которой может быть проблемным. Иными словами, проблема совсем необязательно находится в том звене, в котором она была обнаружена. Она может «перемещаться» по причинно-следственной цепочке от истоков до результата, и быть решена в любом звене.

### *2.4.Аналитика и систематика проблемы.*

Итак, мы выяснили следующее. Во-первых, каждая проблемная ситуация, чтобы быть проблемной, должна иметь как минимум, две составляющие: что должно быть, и почему этого быть не может. Во-вторых, эти две составляющие имеют цепную природу. В-третьих, в зависимости от вида второй составляющей, проблема глобально, состоит в следующем: либо желаемый результат приведет к недопустимому результату, либо желаемый результат невозможен по объективным причинам. В-четвертых, цепная природа проблем позволяет «перемещать» проблемное звено вдоль по цепочке, что дает целый каскад задач, которые можно решать, для решения исходной проблемы.

Для подготовки проблемы к решению необходимо сделать следующие вещи.

Первое - это определить «края» проблемы: желаемый результат, недопустимый результат, исходные причины, вид второго плеча. Это означает, посмотреть на проблему системно (систематика проблемы). Ведь в изначальной формулировке фигурирует только «видимая часть айсберга».

Второе – это выявить все элементы причинно-следственных цепочек в данной системе (аналитика проблемы). Причем эти два процесса невозможно сделать отдельно друг от друга.

Третье – пройти по всем ветвям цепочек, сформулировав задачу для каждого звена, в зависимости от того, в какой цепочке оно находится.

Таким образом, получаем каскад задач, не все из которых имеют смысл, но решение каждой из которых приводит к решению исходной проблемы.

Для практического применения и получения конкретного алгоритма, необходимо внимательно рассмотреть свойства самих причинных и следственных цепочек.