Абстрактные требования редко бывают полными - они не дают полного и недвусмысленного контекста для разработки или тестирования. При работе с абстрактными утверждениями трудно определить функциональные пробелы и несоответствия, пока требования не будут выражены конкретно в коде. Когда во время разработки обнаруживаются функциональные пробелы и двусмысленности, проект всегда переполняется.

Самое раннее задокументированное использование реалистичных примеров в качестве единого источника правды, требований и автоматических тестов по проектам программного обеспечения - проект WyCash +, описанный [Уордом Каннингемом](https://en.wikipedia.org/wiki/Ward_Cunningham) в статье «Образец языка конкурентного развития»  в 1996 году.

Сам термин «Спецификация по примеру» была придумана [Мартином Фаулером](https://en.wikipedia.org/wiki/Martin_Fowler) в 2004 году. [[8]](https://en.wikipedia.org/wiki/Specification_by_example#cite_note-8)

Спецификация по примеру - это эволюция экспериментальной практики « [Экстремальное программирование»](https://en.wikipedia.org/wiki/Extreme_Programming) предложенная в 1999 году, и идея « Вездесущий язык»  с использованием идеи тестов на «черный ящик» в качестве требований, описанных Вайнбергом и Гаузе  в 1989 году.

Итак, представим что мы команда и нам хотелось бы собрать достаточно много репрезентативных примеров, чтобы пройтись по всем вариантам и важным случаям для отдельно взятой возможности системы в рамках текущей итерации. Можем начать с того, что возьмем описание некоторой будущей функциональности системы, например, use-case сценарий, пользовательскую историю или любую другую «единицу» разработки, которую используем для планирования, и зададим заказчику вопрос: «как нам удостовериться, что эта функциональность реализована полностью и корректно?». Иной вопрос, который часто применяют, звучит так: «представьте, что эта будущая функциональность чудесным образом уже реализована; тогда как именно вы бы ее тестировали?». Вот эти примеры нам и нужно будет записать и детально обсудить.

Затем нам следует определить «крайние случаи», негативные сценарии, а также сценарии, в которых не все идет согласно плану, и включить их в обсуждение. Дональд Гоз (Donald Gause) и Джеральд Винберг (Gerald Weinberg) в книге «Exploring Requirements: Quality Before Design», выделяют следующие три категории, которые должны учитываться при создании тестов для отдельного требования (в гибком приемочном тестировании они соответствуют примерам):  
  
1. нормальное использование функциональности  
  
2. непредусмотренное, но разумное использование  
  
3. непредусмотренное и нерациональное использование

Члены команды часто знакомы с крайними случаями и потенциальными проблемами и знают даже, как их можно бы решить, но им еще как-то необходимо их правильно донести. Если бы команда в свое время сообщила о своих проблемах и опасениях то можно было бы избежать многих проблем. Чтобы в нашем программном обеспечении избежать дорогостоящих дефектов подобного рода, нам всем следует сконцентрироваться на рассмотрении как можно большего числа именно таких «нерациональных» случаев. В обязанности тестировщиков как раз входит находить примеры, которые рано выявляли бы проблемные ситуации, являющиеся обычно предметом проверки на более поздней стадии.

Нужно быть внимательным по поводу различий, выявленных в похожих или связанных друг с другом примерах. Обязательно стоит проверить, не нужно ли рассматривать специальные случаи для этих связанных примеров. Крайне вероятно, что это поможет обнаружить пробелы в функциональности. И это гораздо проще сделать, когда примеры выписаны на доске или стенке, поскольку тогда вы сможете различать группы связанных примеров всей командой. Нужно постараться установить общую структуру примеров и записать связанные примеры в одном и том же формате близко друг к другу, чтобы можно было легко выявить несоответствия. Если примеры отличаются всего одной-двумя переменными, стоит выписать их в виде списка или таблицы. Незаполненные ячейки или недостающие в таблице комбинации ячеек могут указывать на еще большее число интересных случаев. Далее рассмотрим реальный пример.

Обратите внимание на примеры для интернет-магазина цветов, предлагающего многоразовые ваучеры на бесплатную доставку для VIP-клиентов, если те делают разовую покупку на сумму более чем $50:  
  
• Марк является VIP-клиентом. Он формирует заказ в корзине на $50 и переходит к оплате. На этапе оплаты ему предлагается бесплатная доставка, поскольку он проживает в США.  
• У Марка осталось неиспользованное предложение на бесплатную доставку. Он формирует свой заказ на $30 и переходит к оплате. На этапе оплаты ему предлагается бесплатная доставка.  
• У Марка нет предыдущих неиспользованных предложений по бесплатной доставке. Он формирует заказ на $30 и переходит к оплате. На этапе оплаты ему предлагается только доставка на общих условиях.  
• Люси – VIP-клиент из Великобритании. Она формирует заказ на $50 и переходит к оплате, но получает только доставку на общих условиях, поскольку проживает в Великобритании. Мы не предлагаем бесплатной доставки за пределами США. Вместо этого мы даем ей бесплатный подарочный ваучер.  
• Том не является VIP-клиентом. Он формирует заказ на $50 и переходит к оплате. На этапе оплаты ему предлагается только доставка на общих условиях.  
  
Сама по себе последовательность выполняемых действий и имена пользователей на самом деле несущественны для этих бизнес-правил. Вместо того, чтобы записывать каждый сценарий как отдельный пример, мы можем использовать единственный список, определяющий, например, следующее: !(указать на слайд)

Если выписать все подобным образом, то сразу становится проще заметить другие случаи, интересные для обсуждения. Очевидно, мы могли бы записать пример для VIP-клиента из Великобритании, сформировавшего заказ на $30. Но здесь присутствуют и такие пробелы в спецификации, которые заметить будет посложнее; но наша таблица помогает выявить также и их. К примеру, что мы делаем, если у кого-то есть неиспользованный ваучер на бесплатную доставку, но он также является VIP-клиентом и делает заказ на более чем $50? Переносим ли мы ваучер по доставке на следующий раз? А что если клиент в этот раз решит не пользоваться бесплатной доставкой – позволим ли мы ему перенести на будущее целых два ваучера или только один? Что будет, если вдруг каким-то образом окажется, что клиент из Великобритании обладает таким ваучером? Как такое вообще может произойти (возможно, система позволяет пользователю изменить адрес, но при этом оставить за собой ваучер), и стоит ли нам блокировать такую возможность? В идеальном случае нам хотелось бы успеть пробежаться по этим важным примерам, имея в распоряжении время экспертов в бизнес-области.

Табличное представление информации весьма полезно в подобных случаях. Сначала определяется структура таблицы, так, чтобы ее колонки подходили под всевозможные случаи, а потом концентрируем усилия на заполнении отдельных строк таблицы. Задача может растянуться на недели или даже месяцы; применение табличного формата помогает удостовериться, что мы не пропустили ни единого важного случая.

Вывод:

Человеческий мозг, как правило, не так хорош в понимании абстракций или новых идей / концепций, когда они впервые подвергаются воздействию, но они действительно хороши в получении абстракций или концепций, если им дано достаточно конкретных примеров.Чем больше примеров мы даем, тем больше вероятность того, что мы правильно поймем намеченный смысл. Кроме того, используя конкретные примеры, они становятся более знакомыми и сопоставимыми с чем-то похожим на наш прошлый опыт, что в целом облегчает их понимание.

Успешное применение спецификации на примере используется для сокращения циклов обратной связи при разработке программного обеспечения, что приводит к меньшей переделке, более высокому качеству продукта, более быстрому изменению по времени программного обеспечения и лучшей согласованности действий различных ролей, участвующих в разработке программного обеспечения, таких как тестеры , аналитиков и разработчиков.