Постановка задач. Как написать задачу, которую поймут правильно?

О чём речь?

Цель данной статьи - прояснить некоторые размытые понятия, связанные с формулированием задач и дать постановщикам набор приёмов и практик, при помощи которых можно ставить задачи, которые исполнитель поймёт точнее.

Формат постановки задачи - описан в статье Руководство по написанию постановки задачи (guidline).
Рекомендую всем перейти по ссылке и освежить свои знания.
Также описание подкрепляет шаблон в Trello: Шаблон постановки задачи на разработку ТРЕБУЕТ ОБСУЖДЕНИЯ / ПР...

Однако, что именно, а главное, как именно надо писать в секции “**Изменения/Требования**“?

Кроме того, помимо постановки задач для Back-а, есть ещё постановка задач для Front-а (Шаблон постановки задачи для SPA (Frontend-разработка по макету) ТРЕБУЕТ ОБСУЖДЕНИЯ / ПР...) и в ней тоже надо как-то описать что именно необходимо сделать.
В виде Изменений или Требований?

Попробуем со всем этим этим разобраться и начнём с уточнения понятий.

Требования vs. Изменения

Что такое “Требования“ и “Изменения“ в контексте наших задач? Какая между ними разница? Она вообще есть? Какой вариант использовать при постановке задачи? Почему именно его?

В Руководстве написано следующее:

- **Изменения** — детальный перечень действий и изменений которые требуется выполнить для выполнения “решения”.
- **Требования** — перечень функциональных требований, то есть какие функции и возможности должны быть реализованы в задаче для достижения цели.

То есть требования - описание функций и возможностей, а изменения - описание шагов. На первый взгляд, разница между этими двумя вариантами постановки задачи не такая очевидная.
Описывая алгоритм мы формулируем последовательность шагов. Это изменения?
Описывая новую функциональность мы делаем описание функции. Это требования?

В сердце различий два важных аспекта:

1. Объект, который постановка задачи описывает
2. Форма изложения

Давайте немного раскроем определения наших понятий с учётом этих аспектов.

Требования, это...

Требования, которые мы формулируем - функциональные и это значит, что у них есть ряд существенных признаков:

- Требования описывают систему, которую мы хотим сделать - метод, экран, пользовательскую функцию, алгоритм.
  И поэтому:

- В описании требований мы ссылаемся на объект, который создаём или модифицируем:
  "*На экране должно отображаться то-то*", "*Необходимо предоставить пользователю такую-то возможность*", "*Метод возвращает такое-то значение*"
- В описании требований также фигурирует актёр - тот, кто выполняет функцию и в функциональных требованиях это всегда программный компонент
  - Порой это просто абстрактная система. Просто потому что мы не знаем или нам не важно какой именно компонент/сервис/модуль должен выполнять описанную работу:
    "*Система должна делать то-то*", "*Система должна вызвать метод такой-то*".
    Так, нет ничего плохого в том, чтобы упоминать систему, формулируя задачи для Front-а
  - Но нередко это и что-то менее абстрактное:
    "*Метод должен проверить то-то*", "*Фронт должен нарисовать UI для ввода такого-то значения*",
    "*Подсистема логирования должна предоставить такой-то интерфейс*"
  - Иногда, для удобства восприятия актёр не называется явно, но он всегда подразумевается:
    "*При нажатии на кнопку "Далее", надо сделать то-то*" - обработка нажатия произойдёт не сама по себе - какой-то программный компонент тут точно должен поработать. При этом из соседних высказываний должно быть чётко понятно, кто тут актёр

- Требования говорят о том, какими свойствами должно обладать решение, которое мы разрабатываем. Они описывают, что должно случиться на конкретном шаге процесса и что должно получиться в итоге.
  И именно поэтому в формулировках функциональных требований всегда встречаются повелительные глаголы:
  "*Система должна*", "*Метод возвращает*", "*Экран отображает*"
- Требования описывают условия при которых должны происходить действия, из-за чего большинство требований это высказывания вида - "Если, то":
  "*Если пользователь нажал крестик, надо: отменить выбор пользователя и закрыть экран*", "*Если выбранный файл больше 20mb, необходимо выдать сообщение об ошибке*"

Итого: Требования описывают программный компонент / систему и, при помощи глаголов, указывают, что система должна делать в тех или иных условиях.

Изменения, это…

Изменения у нас - это инструкция для исполнителя. Отличительные черты изменения:

1. Изменение говорит исполнителю-человеку, что именно он должен сделать. Он, а не система/модуль/программный компонент:
   "*Добавьте новое поле такое-то в вон-ту таблицу*", "*Сделайте импорт таких-то данных*"
2. Изменения редко содержат условия, которые бы влияли на состав и порядок действий человека. Обычно это - линейный список подзадач без ветвлений:
   "*Сделайте миграцию таких-то данных, по таким-то правилам*"
   - В примере выше упомянуты правила, а в правилах будут ветвления и условия, но это - условия для алгоритма обработки данных, который человек должен реализовать, а не условия для человека - делать такой алгоритм или другой
   - Если в постановке подзадачи на миграцию будет детально расписан алгоритм - он будет формулироваться в терминах требований к этому алгоритму. Подробнее о таких ситуациях написано в разделе Формулируя сложные инструкции вы можете незаметно перейти к описанию требований

Итого: Изменения - линейный перечень подзадач, которые должен сделать человек.

Что и в каких случая применять?

Базовые правила простые:

Если ключевая суть задачи - произвести манипуляции с данными, создать или обновить документ, запланировать/проконтролировать работу других людей - т.е. в постановке задачи описаны действия человека - пишите **Изменения**.
Формат Изменений больше подходит для ситуаций, когда решение уже детально проработано постановщиком на уровне конкретных изменений в коде / поведении системы и фокус постановки задачи направлен на описание того, что именно надо сделать.

Если ключевая суть задачи - создание / обновление программного компонента - т.е. в постановке задачи описаны действия системы - пишите **Требования**.
Формат Требований работает лучше, когда нюансы реализации оставляются на усмотрение исполнителя, а фокус постановки задачи направлен на описание желаемого конечного результата.

Самое важное, что стоит помнить - не очень принципиально как именно будет называться раздел - ”Изменения” или “Требования“. Принципиально использовать корректные формулировки, описывая действия человека или системы.

Наилучшие практики формулирования требований

Грамотное формулирование функциональных требований - отдельная инженерная дисциплина. Ей уделяется немало внимания в процессе разработки программного обеспечения из-за того, что ошибки в требованиях случаются часто, а стоят дорого.

В то же время, всеобъемлющий документ с требованиями (Спецификация Требований) это фундаментальный труд, создание которого отнимет много сил и времени, а поддержание в актуальном состоянии будет требовать постоянной работы.
Детальный документ с требованиями отвечает на много вопросов ещё до начала разработки и затрудняет внесение изменений в функциональность, когда разработка уже идёт. Agile-подходы к управлению проектами так или иначе стараются уйти от детальных, “высеченных в камне“ требований, чтобы иметь возможность пробовать разные варианты решения и, при необходимости, вносить существенные изменения “на лету“.

Тем не менее, и в Agile-процессе задачи надо ставить, а от качества постановки зависит точность и скорость реализации. При грамотном подходе, функциональные требования, как мозаика, собираются из отдельных кусочков и наращиваются вместе с ростом разрабатываемой функциональности.

Форма требований. Приёмы формулирования

Существует много разных точек зрения на то, как должны выглядеть требования и как именно следует их оформлять.
Помимо самой формулировки отдельного требования в виде текста / высказывания, в спецификации у требования почти всегда есть уникальный идентификатор и набор атрибутов, которые хранят дополнительную информацию: кто автор требования, когда требование создавалось и обновлялось, какие программные компоненты оно описывает, на какие другие требования ссылается и так далее.

В этой статье мы будем описывать требования в формате, пригодном для постановки наших задач. Это накладывает ряд ограничений и имеет свои особенности. Так что:

Ниже приведено изложение общепринятых практик формулирования требований, адаптированное под нужды нашего процесса.

Требования, из которых состоят наши постановки задач - это высказывания без идентификаторов и атрибутов. Это обычный текст в карточке задачи. Нередко одно высказывание связано с несколькими другими в единую структуру и вместе формирует одно требование.
Давайте разберёмся, как эти высказывания писать и как составлять из них полноценную постановку задачи.

Ключ к хорошим требованиям - грамотная формулировка, эффективная форма подачи. Требование должно быть легко читать и понимать. Оно должно ясно и однозначно передавать смысл. Совокупность требований должна описывать задачу во всей её полноте, на нужном уровне детализации. Чтобы добиться такого эффекта можно использовать описанные ниже приёмы.

Формулируйте кратко

Сжатое формулирование каждого отдельного высказывания предпочтительнее пространного изложения сложноподчинённого предложения. Если передать нужную мысль одним высказыванием не получается - лучше написать два коротких, вместо одного длинного

- Поскольку часть требований описывает условия при которых оно выполняется, а условия могут быть довольно сложными, не всегда у вас получится сделать компактную формулировку. В таких случаях стоит разбить сложное условие на отдельные части - использовать структуру (см. приём Используйте структуру)

Используйте структуру

Структурированный текст - главный приём, при помощи которого можно и нужно отделять требования друг от друга. Конечно все высказывания можно просто свалить в один мега-абзац. Но толку от такой постановки задачи будет мало.
Для того, чтобы описанием задачи, особенно сложной, было удобно пользоваться имеет смысл сделать следующее:

Разделите описание на секции

Грамотная декомпозиция постановки задачи - отдельный большой вопрос. В виде отдельных секций стоит описывать независимые части задачи. Что-то, что можно описать, как единое целое.

Например - в рамках задачи сбора обратной связи есть процесс проверки правильности заполнения формы или, скажем, проверка формата и размера прикрепляемого файла. Эти части функциональности имеет смыл описать отдельными секциями.

Полезно выделять в секции требования, которые будут ре-использоваться внутри вашей постановки задачи. Например - у вас есть правила валидации текстового поля и при этом задача описывает много таких полей в разных местах UI. В отдельной секции можно описать проверку текстового поля на соответствие правилам, а из других мест постановки задачи - просто сослаться на эту секцию.

Главный критерий для выделения секции - возможность описывать эту секцию независимо от других частей постановки. Тогда, при необходимости изменить/дополнить описание, будет достаточно изменить только эту секцию.
Кроме того, функцию, описанную в виде секции, будет проще реализовать и протестировать.

Делайте вложенные высказывания

Довольно часто одно требование состоит из нескольких высказываний, связанных по смыслу. Такие высказывания нужно “упаковать“ в нумерованный или маркированный список:

- На верхнем уровне - общее условие, ниже - последовательность действий и, если нужно, дополнительные ветвления условий
- Нумерованный список применяется, когда есть необходимость сослаться на конкретный пункт требований
- Маркированный список применяется, когда ссылка на отдельные пункты списка не нужна и достаточно просто обозначить структуру - вложенность высказываний
- Нумерованный и маркированный списки можно использовать совместно - на верхнем уровне номера, ниже - маркеры/буллеты

Разбивайте сложные условия на составные части

Начинайте каждую новую посылку / часть сложного условия с новой строки. В таком “развёрнутом“ виде, сложное условие гораздо проще понять целиком и не упустить одну из его частей. Чтобы сделать такое разбиение для высказывания, которое находится в нумерованном или маркированном списке используйте сочетание клавиш Shift + Enter.

Пример такого условия:
Если у очередного доп-параметра нет ни предустановленного, ни выбранного пользователем значения,
Или выбранное значение есть, но оно не входит в список допустимых значений параметра,
И при этом есть больше одного допустимого значения

Пользуйтесь заголовками

Ещё один приём тесно связанный со структурированием описания. В Trello (и тем более в Confluence) есть возможность применять стили заголовков разного уровня вложенности. Использование этих стилей поможет чётко обозначить:

- Разделы постановки задачи: Цель / Проблема, Решение, Требования
- Секции внутри раздела Требования

Надо бы обновить шаблон постановки задачи, чтобы сразу выделить его разделы в виде заголовков.

Вводите определения терминов и сокращений

Часто в требованиях необходимо выделить какие-то специфические понятия. Например - разделить ключевые и дополнительные параметры, чтобы наделить их разными свойствами.

- Необходимо обязательно ввести определение понятия до того, как вы начнёте применять его в тексте требований
  - Будет полезно выделить новый термин в тексте (например - жирным шрифтом). Так читателю будет проще найти его при работе над задачей
  - Это же касается описаний структур, параметров, полей баз данных и тому подобного - вначале надо описать - потом пользоваться
  - Если логика постановки задачи требует вначале использовать понятие, а уже потом описать его целиком, при первом использовании термина - стоит сослаться на секцию / требование, где приведено определение этого термина (см. приём Делайте ссылки)
  - Если хочется использовать сокращение, даже общепринятое, будет лучше вначале упомянуть полное название с использованием Заглавных Букв (ЗБ) а после него - сокращение в скобках
- Используйте одинаковую терминологию
  Порой возникает желание применять синонимы: система, компонент, модуль. Делать этого не следует. Постановка задачи - не литературное произведение. Однозначность понимания тут гораздо важнее красоты изложения. Так что если уж взялись использовать фразу “вызвать метод такой-то“, не переходите на “отправить запрос такой-то“. Даже если очень хочется. Читатель может не понять - перед ним тот же самый объект, просто иначе названный или нечто другое

Выделяйте важное

То, на что читателю следует обратить особое внимание - не грех и подчеркнуть. Например - **выделить** жирным шрифтом.
Такое выделение помогает заметить важную часть текста: определение термина или ключевой компонент условия. Для этой же цели используются специальные стили шрифта, чтобы выделять куски кода, названия полей и тому подобное.
Выделенный текст позволяет быстро найти нужное место - фокусирует внимание. Однако, но не стоит злоупотреблять этим приёмом, иначе происходит обратный эффект - внимание не фокусируется на одном, нужном вам объекте, а рассеивается на много разных.

Дополняйте требования комментариями

Требование описывает то, что должно быть сделано системой. Но из него не всегда понятно для чего это делается. Например - вы ставите задачу для разработки нового метода API. И внутри этого метода происходит какая-то обработка данных, результаты которой будут использоваться на Front-е и влиять на пользовательский сценарий. Бывает полезно дать немного контекста к формулировке требований при помощи комментария. Комментарий - хорошее место, чтобы сделать ссылку на другую задачу или документ.

Делайте ссылки

Ссылки в требованиях - мощный инструмент для решения двух задач:

1. Когда нужно указать на другую секцию в этой постановке задачи
2. Когда надо сослаться на внешние источники:
   - Другие задачи
   - Статьи в Confluence
   - Страницы в сети

Ссылки дают возможность указать на дополнительный контекст или источники связанных данных и при этом не перегружают формулировки требований.

Добавляйте диаграммы и иллюстрации

Одна картинка стоит тысячи слов. Очень часто всё, что требуется для правильного понимания - скриншот, диаграмма, иллюстрация. Тут сложно дать общие рекомендации, но в целом, когда вы описываете процесс с большим количеством ветвлений и изменений состояния - диаграмма может очень помочь. Причём не только будущим читателям, но и вам. Потому что формирование графического описания процесса позволит взглянуть на него под другим углом и заметить упущенные моменты или противоречия.

Не допускайте противоречий

Очевидно, что противоречивые высказывания в рамках одной постановки задачи могут нанести существенный вред - читатель не сможет понять какому из высказываний надо следовать. Нередко противоречия в постановке проявляются неявно и с первого взгляда не заметны. Так, например более общая формулировка может вступать в противоречие с частным случаем, если он явно не описан, как исключение из общего правила.

Необходимо очень внимательно перечитать постановку задач, после её завершения, чтобы найти и устранить противоречия.

Избегайте дублирования

Каждая часть вашей постановки должна описывать свой аспект задачи. Если в требования прокрадываются дублирующие друг друга высказывания при изменении это может привести к появлению противоречивых требований.

Кроме того, наличие дублирующий требований - косвенный признак неудачной структуры. При хорошем структурировании вероятность появления дублей существенно меньше, потому что все требования, относящиеся к конкретной функции находятся рядом и их проще проверить.

Стремитесь к полноте описания

Полнота - важное свойство функциональных требований. Вы описываете поведение программного продукта, на каком-то уровне детализации. На этом уровне (!) ваше описание должно покрывать все возможные исходы. Если какой-то вариант не будет описан - он будет сделан “как-то“, а “как-то“ - это не всегда то, что требуется.

Покрывайте ситуации использования

Чтобы добиться полноты описания надо анализировать ситуации использования, которые вы описываете.

Перед вами форма обратной связи. Что пользователь может в ней делать?

- Заполнить и отправить
  - Надо сообщить пользователю об успешной отправке?
  - А что, если отправка не успешная?
    - Может нужен механизм повторных отправок?
      - Ручной (по команде пользователя) или автоматический (в фоне)?
- Заполнить и закрыть форму
  - Надо ли при этом сохранить то, что пользователь заполнил? Чтобы при следующем открытии формы он мог продолжить?
    - Если решили сохранять, то делать это молча или показать сообщение о том, что данные сохранены?
    - Каким образом будут храниться данные? Как долго?
    - Как мы узнаем, что перед нами тот самый пользователь и у него есть не отправленные данные?
  - Или надо всё стереть?
    - Стоит об этом предупредить пользователя при попытке закрыть форму?

… Этот список вопросов можно продолжать довольно долго. И на все эти вопросы должен быть ответ в вашей постановке задачи.

Собственно, именно так и следует поступать - идентифицировать все ситуации, какие могут возникнуть и в явном виде прописывать их в постановке задачи.
Кроме того - анализ на полноту надо проводить и после того, как постановка завершена - это позволяет найти “дыры“ в описании и закрыть их до того, как задача уйдёт исполнителю.

Покрывайте все ветвления в требованиях

Редкие требования обходятся без условий. Даже требование “долива“ имеет условие - “после отстоя пены“. Функциональные требования без них не обходятся.
А там, где есть условия - есть и варианты поведения, описанные для данного условия. Эти варианты тоже надо описывать явно.

В примере с формой обратной связи можно идентифицировать такое условие:

- Если пользователь заполнил все обязательные поля и нажал кнопку “Отправить“, то - делать то-то
  - А что будет если пользователь нажал кнопку “Отправить“, не заполнив все обязательные поля?
    - Надо выводить сообщение об ошибке, подсветить незаполненные обязательные поля
  - Может кнопка “Отправить” будет отображаться, как недоступная / серая, пока все обязательные поля не заполнены?
    - Но, может при этом она всё-таки будет воспринимать нажатия и выводить сообщение об ошибке?
- А что, если заполнил и **не** отправил?
  - Тут может скрываться поведение по умолчанию - например браузер будет терпеливо ждать, пока не истечёт время сессии
  - Или для таких ситуаций была реализована специальная обработка - про неё можно рассказать в (или сослаться, если есть куда). Это будет не новая функциональность, но и разработчику и тестировщику будет полезно знать, что произойдёт при такой комбинации условий

Важно - явно идентифицировать и описать все варианты ветвления. Даже те, которые вы не собираетесь реализовывать - их надо описать комментариями. Главное - закрыть все “дыры“ в логике условных требований.

Несложно увидеть, что первый и второй подход, по сути очень близки - и там и там вы анализируете пространство возможных вариантов. Разница в том, что в первом случае вы идентифицируете ситуации, а во втором - убеждаетесь в логической полноте их описания.

Добивайтесь однозначности формулировок

Хорошая формулировка - такая, которую все понимают одинаково. В ней нет пространства для интерпретаций. Добиться этого помогает несколько полезных практик:

- Следование формату “Если-то“.
  В начале высказывания - посылка / условие, потом - заключение / действие
- Использование повелительных глаголов.
  ”Система вызывает метод такой-то”, “Метод такой-то проверяет то-то“, “Если в результате проверки обнаружено то-то, метод такой-то возвращает вон-то“
- Ссылка на актёра.
  Недаром в примере выше упомянут именно “Метод такой-то“
- Следование ранее упомянутым приёмам:
  Использование описанных терминов, Структурирование описания, Краткость и непротиворечивость

Стоит ещё раз подчеркнуть, что постановка задачи при помощи требований - не литературное произведение. Тут лучше работает принцип “безобразно, но единообразно“. Хотя намеренно писать коряво тоже в общем-то не стоит.

Думайте о читателе!

Возможно самый абстрактный, но, не исключено, что самый полезный приём. Когда вы пишите свою постановку задачи - обязательно посмотрите на неё глазами читателя. Не боясь повторения и дублирования (всё-таки перед документ про требования, а не сами требования) перечислю несколько важных моментов:

Обязательно описывайте контекст

Читатель - не знает того, что знаете вы. У него свой контекст, своё содержимое головы. Делая постановку задачи вам нужно погрузить читателя в контекст задачи - дать ему необходимые ориентиры:

- Для чего всё это делается и какая от этого польза
- С какими другими задачами связана эта - может есть “большая картинка“ в рамках которой ваша задача - лишь один из “кусочков пазла”
- С какими подсистемами общается описанный вами модуль, для решения единой пользовательской задачи

Все перечисленные вопросы так или иначе касаются одного и того же контекста.
Описание контекста будет полезно не только читателям/исполнителям. Если вы вернётесь к своему собственному описанию через полгода/год вы будете благодарны себе за приложенные усилия. За это время ваш контекст поменяется и вы можете не вспомнить для достижения какой цели все это делалось и в рамках какой общей системы. Кроме того, за прошедшее время многое могло поменяться. Описание контекста поможет понять на каких основаниях была построена ваша задача, что, в свою очередь, поможет вам актуализировать её постановку под изменившиеся обстоятельства.

Для описания контекста полезно использовать ссылки и комментарии.

Принимайте во внимание то, как постановку задачи будут читать

Ещё раз (надеюсь в последний), постановка задачи - не литературное произведение. Это скорее справочная информация.
Задумайтесь над тем, как люди будут пользоваться вашим описанием задач.

Вначале просто прочитают сверху вниз. Потом, начнут с ней работать - реализовывать или тестировать отдельные части. Будут читать по частям и с середины - искать нужные в данный момент детали.

Вот почему так важна хорошая структура и прочие “хитрости“ - так читатель сможет сконцентрироваться на отдельной, нужной ему в данный момент части, без необходимости бегать вверх/вниз по всей “портянке“ в Trello, чтобы собрать описание одной функции в единое целое. Ведь в таком случае повышается вероятность, что читатель соберёт это описание неправильно.

Прочитайте готовую постановку задачи с точки зрения исполнителя

Это не всегда бывает просто, но рефлексия - способность ставить себя на место другого человека, чтобы увидеть ситуацию с его точки зрения - это свойство человеческого разума. Так что вы справитесь.

Абстрагируйтесь от написания и просто прочитайте то, что у вас получилось. Внимательно и от начала до конца. Вам точно всё понятно? Вы всё это поняли из текста или что-то взяли из головы? Если для понимания вы использовали свои знания о предметной области (например - знания архитектуры программного компонента) - это знание общеизвестно в команде или для вашего читателя нормально чего-то не знать?
В общем - проанализируйте понятность, полноту, контекст, да и сам текст с позиции другого человека. Очень полезно примерить на себя разные роли читателей: понятно ли это разработчику / тестировщику / архитектору? Будет ли у них одинаковое понимание? Будет ли им удобно это читать? Понятны ли отдельные секции сами по себе?

> Приёмы и подходы, описанные в разделе Форма требований. Приёмы формулирования - не истина в последней инстанции и ни один из них не гарантирует написания хороших требований. Но они определённо могут помочь вам начать мыслить в нужном направлении и учитывать моменты, улучшающие качество постановки задачи.

Особенности написания Изменений

Многие вещи, раскрытые в разделе Наилучшие практики формулирования требований на самом деле применимы и к написанию Изменений. Тут точно так же важна структура и непротиворечивость. Нужен контекст и ссылки. Требуется краткость и однозначность понимания. Всё это можно и нужно использовать при использовании формата постановки задачи в виде описания изменений.

“Но есть нюансы”

Вспомним, что формат изложения типа “Изменения” - это инструкция для человека. Объект вашей постановки - сам исполнитель. К нему и следует обращаться.

Инструкции должны быть конкретными и детальными

Человеку должно быть понятно, что именно он должен сделать и описано это должно быть **с высокой степенью детализации**:

- Добавить такое-то поле в такую-то таблицу
- Провести такую-вот миграцию данных
- Реализовать метод с такими входными и выходными параметрами
- Проверить то-то и сделать то-то

> Это ответственность постановщика, чтобы, после выполнения всех описанных действий, получился ожидаемый и полезный (!) результат. Исполнитель - просто следует инструкциям и пространство для манёвра у него четко ограничено описанными шагами.

Высокие требования к детальности и конкретности делают применение формата постановки задач в виде изменений “в чистом виде” пригодным только для относительно несложных ситуаций, где проявлять “творчество“ не желательно или даже вредно.

Инструкции должны быть выполнимы

Задание, которое вы формулируете не должно относиться к задачам типа отделения рисовых зёрнышек, от гречневых. Если только вы не ставите задачу автоматизации этого безрадостного процесса. Однако в этом случае вам нужно будет формулировать задачу в виде требований.

Очевидно, что задача сформулированная как инструкция для человека, должна быть в пределах сил человеческих.

Инструкции должны быть конечны

Еще одно очевидное свойство, которое тем не менее важно упомянуть - формулируя задачу в виде инструкций надо избегать “зацикливания“. У задачи должно быть начало и конец. Задача у которой нет явно обозначенного окончания - это регулярная активность - то есть совсем другой вид деятельности.

Тут стоит ещё раз напомнить, что по окончании всех перечисленных шагов, задача, описанная в формате изменений обязана привести к желаемому результату.

Формулируя сложные инструкции вы можете незаметно перейти к описанию требований

Можно поставить задачу в формате перечисления изменений и, в процессе перейти в формат требований, а потом вернуться обратно. На самом деле это случается довольно часто.

Вы формулируете изменения - Добавить в API новый метод с такими-то входными и выходными параметрами.
А потом описываете то, как метод обрабатывает вход, чтобы получить выход. Делаете вы это при помощи формулирования требований к функциональности метода. И при этом должны применять все приёмы, относящиеся к написанию требований.
После этого вы добавляете в постановку новую инструкцию для человека - обновить базу данных и подготовить скрипт для миграции. И снова углубляетесь в описание работы скрипта.

Описанный процесс - совершенно нормальный.
Как при этом называть раздел постановки задачи - в данном случае не очень принципиально, но правильнее всё-таки озаглавить его как “Требования”. Это больше соответствует определению приведённому в данной статье и соответствует стратегии из секции Что и в каких случая применять?

В чистом виде Изменения - это набор простых подзадач, которые не требуют дополнительной детализации. Во всех остальных случаях стоит сразу писать требования в состав которых, по мере необходимости, будут входить и инструкции для человека-исполнителя.