2.2.5. Свойства качественных требований

В процессе тестирования требований проверяется их соответствие определённому набору свойств (рисунок 2.2.f).

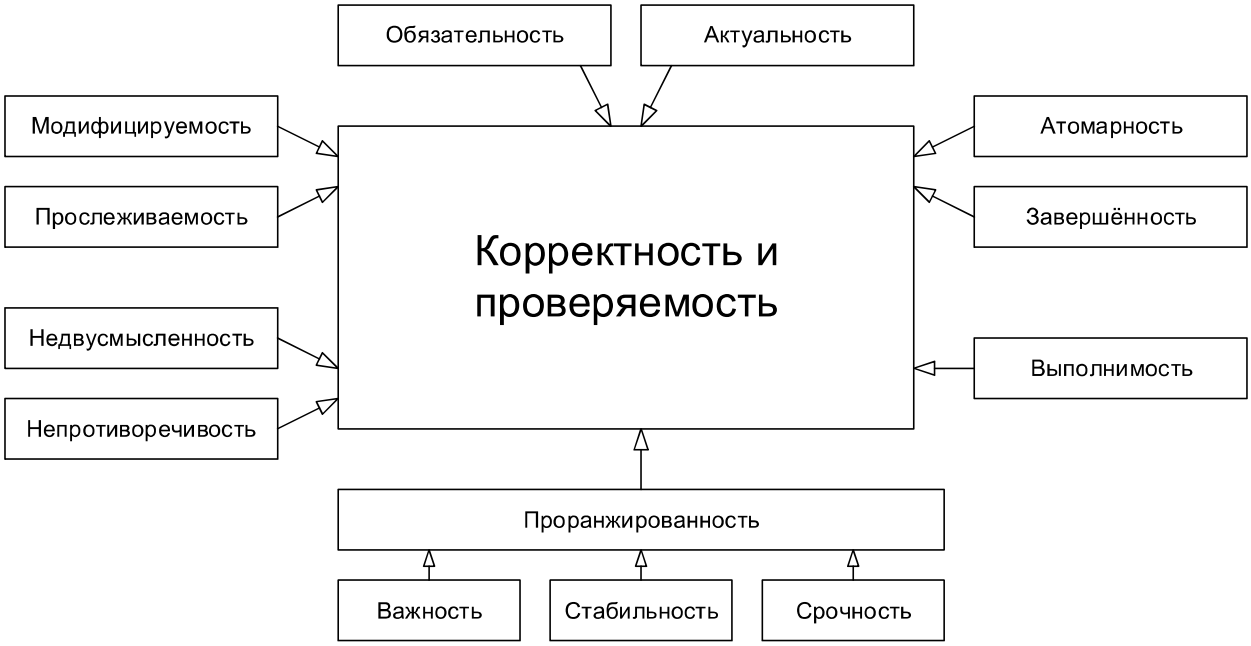

Рисунок 2.2.f — Свойства качественного требования

Завершённость (completeness[87]). Требование является полным и законченным с точки зрения представления в нём всей необходимой информации, ничто не пропущено по соображениям «это и так всем понятно».

Типичные проблемы с завершённостью:

- Отсутствуют нефункциональные составляющие требования или ссылки на соответствующие нефункциональные требования (например: «*пароли должны храниться в зашифрованном виде*» — каков алгоритм шифрования?).
- Указана лишь часть некоторого перечисления (например: «*экспорт осуществляется в форматы PDF, PNG и т.д.*» — что мы должны понимать под «и т.д.»?).
- Приведённые ссылки неоднозначны (например: «*см. выше*» вместо «*см. раздел 123.45.b*»).

Способы обнаружения проблем	Способы устранения проблем
Применимы почти все техники тестирования требований{51}, но лучше всего помогает задавание вопросов и использование графического представления разрабатываемой системы. Также очень помогает глубокое знание предметной области, позволяющее замечать пропущенные фрагменты информации.	Как только было выяснено, что чего-то не хватает, нужно получить недостающую информацию и дописать её в требования. Возможно, потребуется небольшая переработка требований.

[87] Each requirement must contain all the information necessary for the reader to understand it. In the case of functional requirements, this means providing the information the developer needs to be able to implement it correctly. No requirement or necessary information should be absent. [«Software Requirements (3rd edition)», Karl Wiegers and Joy Beatty]

Атомарность, единичность (atomicity[88]). Требование является атомарным, если его нельзя разбить на отдельные требования без потери завершённости и оно описывает одну и только одну ситуацию.

Типичные проблемы с атомарностью:

- В одном требовании, фактически, содержится несколько независимых (например: «*кнопка "Restart" не должна отображаться при остановленном сервисе, окно "Log" должно вмещать не менее 20-ти записей о последних действиях пользователя*» — здесь зачем-то в одном предложении описаны совершенно разные элементы интерфейса в совершенно разных контекстах).
- Требование допускает разночтение в силу грамматических особенностей языка (например: «*если пользователь подтверждает заказ и редактирует заказ или откладывает заказ, должен выдаваться запрос на оплату*» — здесь описаны три разных случая, и это требование стоит разбить на три отдельных во избежание путаницы). Такое нарушение атомарности часто влечёт за собой возникновение противоречивости.
- В одном требовании объединено описание нескольких независимых ситуаций (например: «*когда пользователь входит в систему, ему должно отображаться приветствие; когда пользователь вошёл в систему, должно отображаться имя пользователя; когда пользователь выходит из системы, должно отображаться прощание*» — все эти три ситуации заслуживают того, чтобы быть описанными отдельными и куда более детальными требованиями).

Способы обнаружения проблем	Способы устранения проблем
Обдумывание, обсуждение с коллегами и здравый смысл: если мы считаем, что некий раздел требований перегружен и требует декомпозиции, скорее всего, так и есть.	Переработка и структурирование требований: разбиение их на разделы, подразделы, пункты, подпункты и т.д.

Непротиворечивость, последовательность (consistency[89]). Требование не должно содержать внутренних противоречий и противоречий другим требованиям и документам.

Типичные проблемы с непротиворечивостью:

- Противоречия внутри одного требования (например: «*после успешного входа в систему пользователя, не имеющего права входить в систему…*» — тогда как он успешно вошёл в систему, если не имел такого права?)
- Противоречия между двумя и более требованиями, между таблицей и текстом, рисунком и текстом, требованием и прототипом и т.д. (например: «*712.a Кнопка "Close" всегда должна быть красной*» и «*36452.x Кнопка "Close" всегда должна быть синей*» — так всё же красной или синей?)
- Использование неверной терминологии или использование разных терминов для обозначения одного и того же объекта или явления (например: «*в случае, если разрешение окна составляет менее 800x600…*» — разрешение есть у экрана, у окна есть размер).

[88] Each requirement you write represents a single market need that you either satisfy or fail to satisfy. A well written requirement is independently deliverable and represents an incremental increase in the value of your software. [«Writing Good Requirements — The Big Ten Rules», Tyner Blain: http://tynerblain.com/blog/2006/05/25/writing-good-requirements-the-big-ten-rules/]

[89] Consistent requirements don't conflict with other requirements of the same type or with higher-level business, user, or system requirements. [«Software Requirements (3rd edition)», Karl Wiegers and Joy Beatty]

Способы обнаружения проблем	Способы устранения проблем
Лучше всего обнаружить противоречивость помогает хорошая память ☺, но даже при её наличии незаменимым инструментом является графическое представление разрабатываемой системы, позволяющее представить всю ключевую информацию в виде единой согласованной схемы (на которой противоречия очень заметны).	После обнаружения противоречия нужно прояснить ситуацию с заказчиком и внести необходимые правки в требования.

Недвусмысленность (unambiguousness[90], clearness). Требование должно быть описано без использования жаргона, неочевидных аббревиатур и расплывчатых формулировок, должно допускать только однозначное объективное понимание и быть атомарным в плане невозможности различной трактовки сочетания отдельных фраз.

Типичные проблемы с недвусмысленностью:

- Использование терминов или фраз, допускающих субъективное толкование (например: «*приложение должно поддерживать передачу больших объёмов данных*» — насколько «больших»?) Вот лишь небольшой перечень слов и выражений, которые можно считать верными признаками двусмысленности: адекватно (adequate), быть способным (be able to), легко (easy), обеспечивать (provide for), как минимум (as a minimum), быть способным (be capable of), эффективно (effectively), своевременно (timely), применимо (as applicable), если возможно (if possible), будет определено позже (to be determined, TBD), по мере необходимости (as appropriate), если это целесообразно (if practical), но не ограничиваясь (but not limited to), быть способно (capability of), иметь возможность (capability to), нормально (normal), минимизировать (minimize), максимизировать (maximize), оптимизировать (optimize), быстро (rapid), удобно (user-friendly), просто (simple), часто (often), обычно (usual), большой (large), гибкий (flexible), устойчивый (robust), по последнему слову техники (state-of-the-art), улучшенный (improved), результативно (efficient). Вот утрированный пример требования, звучащего очень красиво, но совершенно нереализуемого и непонятного: «*В случае необходимости оптимизации передачи больших файлов система должна эффективно использовать минимум оперативной памяти, если это возможно*».
- Использование неочевидных или двусмысленных аббревиатур без расшифровки (например: «*доступ к ФС осуществляется посредством системы прозрачного шифрования*» и «*ФС предоставляет возможность фиксировать сообщения в их текущем состоянии с хранением истории всех изменений*» — ФС здесь обозначает файловую систему? Точно? А не какой-нибудь «Фиксатор Сообщений»?)
- Формулировка требований из соображений, что нечто должно быть всем очевидно (например: «*Система конвертирует входной файл из формата PDF в выходной файл формата PNG*» — и при этом автор считает совершенно очевидным, что имена файлов система получает из командной строки, а многостраничный PDF конвертируется в несколько PNG-файлов, к именам которых добавляется «page-1», «page-2» и т.д.). Эта проблема перекликается с нарушением корректности.

[90] Natural language is prone to two types of ambiguity. One type I can spot myself, when I can think of more than one way to interpret a given requirement. The other type of ambiguity is harder to catch. That's when different people read the requirement and come up with different interpretations of it. [«Software Requirements (3rd edition)», Karl Wiegers and Joy Beatty]

Способы обнаружения проблем	Способы устранения проблем
Увидеть в требованиях двусмысленность хорошо помогают перечисленные выше слова-индикаторы. Столь же эффективным является продумывание проверок: очень тяжело придумать объективную проверку для требования, допускающего разночтение.	Самый страшный враг двусмысленности – числа и формулы: если что-то можно выразить в формульном или числовом виде (вместо словесного описания), обязательно стоит это сделать. Если это невозможно, стоит хотя бы использовать максимально точные технические термины, отсылки к стандартам и т.п.

Выполнимость (feasibility[91]). Требование должно быть технологически выполнимым и реализуемым в рамках бюджета и сроков разработки проекта.

Типичные проблемы с выполнимостью:

- Так называемое «озолочение» (gold plating) — требования, которые крайне долго и/или дорого реализуются и при этом практически бесполезны для конечных пользователей (например: «*настройка параметров для подключения к базе данных должна поддерживать распознавание символов из жестов, полученных с устройств трёхмерного ввода*»).
- Технически нереализуемые на современном уровне развития технологий требования (например: «*анализ договоров должен выполняться с применением искусственного интеллекта, который будет выносить однозначное корректное заключение о степени выгоды от заключения договора*»).
- В принципе нереализуемые требования (например: «*система поиска должна заранее предусматривать все возможные варианты поисковых запросов и кэшировать их результаты*»).

Способы обнаружения проблем	Способы устранения проблем
Увы, здесь есть только один путь: максимально нарабатывать опыт и исходить из него. Невозможно понять, что некоторое требование «стоит» слишком много или вовсе невыполнимо, если нет понимания процесса разработки ПО, понимания предметной области и иных сопутствующих знаний.	При обнаружении невыполнимости требования не остаётся ничего другого, как подробно обсудить ситуацию с заказчиком и/или изменить требование (возможно – отказаться от него), или пересмотреть условия выполнения проекта (сделав выполнение данного требования возможным).

Обязательность, нужность (obligatoriness[92]) и **актуальность** (up-to-date). Если требование не является обязательным к реализации, оно должно быть просто исключено из набора требований. Если требование нужное, но «не очень важное», для указания этого факта используется указание приоритета (см. «проранжированность по…»). Также исключены (или переработаны) должны быть требования, утратившие актуальность.

Типичные проблемы с обязательностью и актуальностью:

- Требование было добавлено «на всякий случай», хотя реальной потребности в нём не было и нет.

[91] It must be possible to implement each requirement within the known capabilities and limitations of the system and its operating environment, as well as within project constraints of time, budget, and staff. [«Software Requirements (3rd edition)», Karl Wiegers and Joy Beatty]

[92] Each requirement should describe a capability that provides stakeholders with the anticipated business value, differentiates the product in the marketplace, or is required for conformance to an external standard, policy, or regulation. Every requirement should originate from a source that has the authority to provide requirements. [«Software Requirements (3rd edition)», Karl Wiegers and Joy Beatty]

- Требованию выставлены неверные значения приоритета по критериям важности и/или срочности.
- Требование устарело, но не было переработано или удалено.

Способы обнаружения проблем	Способы устранения проблем
Постоянный (периодический) пересмотр требований (желательно – с участием заказчика) позволяет заметить фрагменты, потерявшие актуальность или ставшие низкоприоритетными.	Переработка требований (с устранением фрагментов, потерявших актуальность) и переработка фрагментов, у которых изменился приоритет (часто изменение приоритета ведёт и к изменению формулировки требования).

Прослеживаемость (traceability[93, 94]). Прослеживаемость бывает вертикальной (vertical traceability[95]) и горизонтальной (horizontal traceability[96]). Вертикальная позволяет соотносить между собой требования на различных уровнях требований, горизонтальная позволяет соотносить требование с тест-планом, тест-кейсами, архитектурными решениями и т.д.

Для обеспечения прослеживаемости часто используются специальные инструменты по управлению требованиями (requirements management tool[97]) и/или матрицы прослеживаемости (traceability matrix[98]).

Типичные проблемы с прослеживаемостью:

- Требования не пронумерованы, не структурированы, не имеют оглавления, не имеют работающих перекрёстных ссылок.
- При разработке требований не были использованы инструменты и техники управления требованиями.
- Набор требований неполный, носит обрывочный характер с явными «пробелами».

Способы обнаружения проблем	Способы устранения проблем
Нарушения прослеживаемости становятся заметны в процессе работы с требованиями, как только у нас возникают **остающиеся без ответа** вопросы вида «откуда взялось это требование?», «где описаны сопутствующие (связанные) требования?», «на что это влияет?».	Переработка требований. Возможно, придётся даже менять структуру набора требований, но всё точно начнётся с расстановки множества перекрёстных ссылок, позволяющих осуществлять быструю и прозрачную навигацию по набору требований.

[93] **Traceability.** The ability to identify related items in documentation and software, such as requirements with associated tests. [ISTQB Glossary]

[94] A traceable requirement can be linked both backward to its origin and forward to derived requirements, design elements, code that implements it, and tests that verify its implementation. [«Software Requirements (3rd edition)», Karl Wiegers and Joy Beatty]

[95] **Vertical traceability.** The tracing of requirements through the layers of development documentation to components. [ISTQB Glossary]

[96] **Horizontal traceability.** The tracing of requirements for a test level through the layers of test documentation (e.g. test plan, test design specification, test case specification and test procedure specification or test script). [ISTQB Glossary]

[97] **Requirements management tool.** A tool that supports the recording of requirements, requirements attributes (e.g. priority, knowledge responsible) and annotation, and facilitates traceability through layers of requirements and requirements change management. Some requirements management tools also provide facilities for static analysis, such as consistency checking and violations to predefined requirements rules. [ISTQB Glossary]

[98] **Traceability matrix.** A two-dimensional table, which correlates two entities (e.g., requirements and test cases). The table allows tracing back and forth the links of one entity to the other, thus enabling the determination of coverage achieved and the assessment of impact of proposed changes. [ISTQB Glossary]

Модифицируемость (modifiability[99]). Это свойство характеризует простоту внесения изменений в отдельные требования и в набор требований. Можно говорить о наличии модифицируемости в том случае, если при доработке требований искомую информацию легко найти, а её изменение не приводит к нарушению иных описанных в этом перечне свойств.

Типичные проблемы с модифицируемостью:

- Требования неатомарны (см. «атомарность») и непрослеживаемы (см. «прослеживаемость»), а потому их изменение с высокой вероятностью порождает противоречивость (см. «непротиворечивость»).
- Требования изначально противоречивы (см. «непротиворечивость»). В такой ситуации внесение изменений (не связанных с устранением противоречивости) только усугубляет ситуацию, увеличивая противоречивость и снижая прослеживаемость.
- Требования представлены в неудобной для обработки форме (например, не использованы инструменты управления требованиями, и в итоге команде приходится работать с десятками огромных текстовых документов).

Способы обнаружения проблем	Способы устранения проблем
Если при внесении изменений в набор требований, мы сталкиваемся с проблемами, характерными для ситуации потери прослеживаемости, значит – мы обнаружили проблему с модифицируемостью. Также модифицируемость ухудшается при наличии практически любой из рассмотренных в данном разделе проблем с требованиями.	Переработка требований с первостепенной целью повысить их прослеживаемость. Параллельно можно устранять иные обнаруженные проблемы.

Проранжированность по важности, стабильности, срочности (ranked[100] for importance, stability, priority). Важность характеризует зависимость успеха проекта от успеха реализации требования. Стабильность характеризует вероятность того, что в обозримом будущем в требование не будет внесено никаких изменений. Срочность определяет распределение во времени усилий проектной команды по реализации того или иного требования.

Типичные проблемы с проранжированностью состоят в её отсутствии или неверной реализации и приводят к следующим последствиям.

- Проблемы с проранжированностью по важности повышают риск неверного распределения усилий проектной команды, направления усилий на второстепенные задачи и конечного провала проекта из-за неспособности продукта выполнять ключевые задачи с соблюдением ключевых условий.
- Проблемы с проранжированностью по стабильности повышают риск выполнения бессмысленной работы по совершенствованию, реализации и тестированию требований, которые в самое ближайшее время могут претерпеть кардинальные изменения (вплоть до полной утраты актуальности).

[99] To facilitate modifiability, avoid stating requirements redundantly. Repeating a requirement in multiple places where it logically belongs makes the document easier to read but harder to maintain. The multiple instances of the requirement all have to be modified at the same time to avoid generating inconsistencies. Cross-reference related items in the SRS to help keep them synchronized when making changes. [«Software Requirements (3rd edition)», Karl Wiegers and Joy Beatty]

[100] Prioritize business requirements according to which are most important to achieving the desired value. Assign an implementation priority to each functional requirement, user requirement, use case flow, or feature to indicate how essential it is to a particular product release. [«Software Requirements (3rd edition)», Karl Wiegers and Joy Beatty]

- Проблемы с проранжированностью по срочности повышают риск нарушения желаемой заказчиком последовательности реализации функциональности и ввода этой функциональности в эксплуатацию.

Способы обнаружения проблем	Способы устранения проблем
Как и в случае с актуальностью и обязательностью требований, здесь лучшим способом обнаружения недоработок является постоянный (периодический) пересмотр требований (желательно – с участием заказчика), в процессе которого можно обнаружить неверные значения показателей срочности, важности и стабильности обсуждаемых требований.	Прямо в процессе обсуждения требований с заказчиком (во время пересмотра требований) стоит вносить правки в значения показателей срочности, важности и стабильности обсуждаемых требований.

Корректность (correctness[101]) и **проверяемость** (verifiability[102]). Фактически эти свойства вытекают из соблюдения всех вышеперечисленных (или можно сказать, что они не выполняются, если нарушено хотя бы одно из вышеперечисленных). В дополнение можно отметить, что проверяемость подразумевает возможность создания объективного тест-кейса (тест-кейсов), однозначно показывающего, что требование реализовано верно и поведение приложения в точности соответствует требованию.

К типичным проблемам с корректностью также можно отнести:

- опечатки (особенно опасны опечатки в аббревиатурах, превращающие одну осмысленную аббревиатуру в другую также осмысленную, но не имеющую отношения к некоему контексту; такие опечатки крайне сложно заметить);
- наличие неаргументированных требований к дизайну и архитектуре;
- плохое оформление текста и сопутствующей графической информации, грамматические, пунктуационные и иные ошибки в тексте;
- неверный уровень детализации (например, слишком глубокая детализация требования на уровне бизнес-требований или недостаточная детализация на уровне требований к продукту);
- требования к пользователю, а не к приложению (например: «пользователь должен быть в состоянии отправить сообщение» — увы, мы не можем влиять на состояние пользователя).

Способы обнаружения проблем	Способы устранения проблем
Поскольку здесь мы имеем дело с «интегральной» проблемой, обнаруживается она с использованием ранее описанных способов. Отдельных уникальных методик здесь нет.	Внесение в требования необходимых изменений – от элементарной правки обнаруженной опечатки, до глобальной переработки всего набора требований.

Хорошее краткое руководство по написанию качественных требований представлено в статье «Writing Good Requirements — The Big Ten Rules»[103].

[101] Each requirement must accurately describe a capability that will meet some stakeholder's need and must clearly describe the functionality to be built. [«Software Requirements (3rd edition)», Karl Wiegers and Joy Beatty]

[102] If a requirement isn't verifiable, deciding whether it was correctly implemented becomes a matter of opinion, not objective analysis. Requirements that are incomplete, inconsistent, infeasible, or ambiguous are also unverifiable. [«Software Requirements (3rd edition)», Karl Wiegers and Joy Beatty]

[103] «Writing Good Requirements — The Big Ten Rules», Tyner Blain [http://tynerblain.com/blog/2006/05/25/writing-good-requirements-the-big-ten-rules/]