MoSCoW − это метод расстановки приоритетов, используемый в бизнес-анализе и разработке программного обеспечения, для достижения заинтересованными сторонами взаимного понимания важности, которую они придают каждому требованию – также известному как приоритезация MoSCoW или анализ MoSCoW.

Заглавные буквы в акрониме MoSCoW означают:

* M − должен сделать это (англ. − «MUST have this.»)
* S − должен сделать это, если это вообще возможно (англ. − «SHOULD

have this if at all possible.»)

* C − мог бы сделать это, если это не повлияет отрицательно на что- то другое (англ. − «COULD have this if itdoes not affect anything else.»)
* W − не будет достаточно времени на это, но в будущем хотелось бы.

Или хочу. (англ. − WON’T have this timebut WOULD like in the future. Alternatively WANT.»)

Буквы «о» добавлены в акроним MoSCoW для удобства произношения. Часто их оставляют строчными, чтобы показать, что они ничего не означают.

Элементы, помеченные как M обязаны быть включены в продукт. Проект не может считаться успешно завершенным, если хоть один подобный элемент (сценарий/возможность/требование) не включен в продукт.

Элементы S также являются критическими для продукта, но, тем не менее, они могут быть исключены из текущей версии продукта по объективным причинам.

Элементы C являются некритичными, но способны увеличить пользовательскую удовлетворенность.

Элементы W являются наименее критичными для продукта. Они могут считаться интересными и перспективными для будущих версий продукта, но точно не будут реализованы в текущей версии.

Различают две категории представления требований: требования заказчика (первичные требования ) и требования разработчика (детальные требования). Отличаются они друг от друга степенью проработки описаний

Первичные требования документируют желания и потребности заказчика и пишутся на языке, понятном заказчику. Детальные требования документируют требования в специальной, структурированной форме, они детализированы по отношению к первичным требованиям.

Работу по созданию первичных требований будем называть сбором, или формированием требований. Проводится она на этапе подготовки жизненного цикла разработки. Работу по созданию детальных требований будем называть анализом требований.

ТРЕБОВАНИЯ ЗАКАЗЧИКА

ПО должно обеспечить средства для ввода и сохранения разнообразных данных абонента-пользователя,

ТРЕБОВАНИЯ РАЗРАБОТЧИКА

Пользователь должен иметь возможность определять тип вводимых данных.

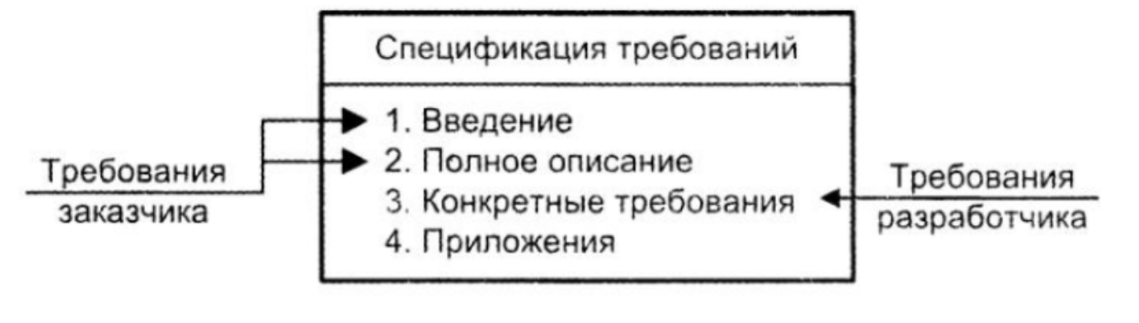
Для каждого типа данных должно иметься соответствующее средство, обеспечивающее ввод и сохранение элемента данных этого типа.

Каждый тип данных должен представляться соответствующей пиктограммой на дисплее пользователя.

Пользователю должна предлагаться пиктограмма для каждого типа данных. Кроме того, должна предлагаться возможность самостоятельного выбора пиктограммы для каждого типа данных.

При выборе пользователем пиктограммы типа данных к элементу данных должно быть применено средство, ассоциированное с указанным типом.

Требования заказчика являются первичным описанием на естественном языке функций, выполняемых системой, и ограничений, накладываемых на нее. Дополнительно к ним могут прикладываться поясняющие диаграммы. Требования заказчика помещаются в системную спецификацию. Требования разработчика содержат детализированное описание функций и ограничений системы, оформляемое в виде спецификации анализа. Она служит основой для заключения контракта между заказчиками ПС и разработчиками.

Весьма часто стандарты программной инженерии интегрируют обе спецификации в единый документ. 

Спецификация требований к ПО является основным документом, определяющим план

разработки ПО.

Все требования, указанные в спецификации, делятся на функциональные и нефункциональные.

Функциональные требования определяют действия, которые должна выполнять система,

без учета ограничений, связанных с ее реализацией. Тем самым функциональные

требования определяют поведение системы в процессе обработки информации.

При этом исходят из всестороннего анализа проблемной (предметной) области.

Рассматриваются разнообразные варианты поведения, определяемые различными

данными и состояниями внешней среды.

Нефункциональные требования не определяют поведение системы, но описывают атрибуты

системы или атрибуты системного окружения.

Дополнительно могут перечисляться ограничения, накладываемые на действия и функции системы, а также на условия разработки(ограничения по времени, ограничения на организацию проекта, стандарты и т.д.)

ПО для системы управления летательными аппаратами (ЛА)

1. Система должна измерять проекции скорости (по трем каналам), вычислять, отклонения от программных значений вектора скорости и радиус-вектора, а затем формировать управляющие воздействия, посылая их на исполнительные органы летательного аппарата (ЛА).

2. Система должна обеспечивать угловую стабилизацию движения ЛА: измерение углов (по трем каналам), вычисление отклонений углов от программных значений, формирование управляющих воздействий, распределение управляющих воздействий между исполнительными органами.

3. Каждая программа автопилота должна снабжаться уникальным идентификатором (Pгogram\_ID), который записывается в черный ящик вместе с параметрами полета.

Формы записи требований могут быть самыми разными.

Однако требования должны быть:

* ясными (не до пускать двоякого толкования, приводящего к искажению смысла пожеланий заказчика);
* согласованными (не содержать противоречивых и взаимоисключающих утверждений);
* полными (определять всю требуемую функциональность системы)

Нефункциональные требования не связаны непосредственно с функциями системы. Многие

нефункциональные требования относятся к системе в целом, а не к отдельным ее элементам.

Это означает, что они могут быть более критичными, чем единичные функциональные требования.

Ошибка в функциональном требовании может понизить функциональные возможности системы, а ошибка в нефункциональном требовании может привести к отказу всей системы.

Существует подход, когда выделяют три группы нефункциональных требований:

* Требования к программной системе. Описывают свойства и характеристики системы. Сюда относятся требования к скорости работы, производительности, емкости необходимой памяти, надежности, переносимости системы на разные платформы и удобству эксплуатации.
* Организационные требования. Отображают вопросы работы и организации взаимодействия заказчика и разработчика. Они включают стандарты разработки программной системы, требования к методам и средствам разработки, указывают сроки создания н набор документации.
* Внешние требования. Учитывают факторы внешней среды. Они определяют требования по взаимодействию данной системы с внешним окружением, юридические обязательства, а также этические требования,гарантирующие приемлемость системы для пользователей.

Первичные требования должны определять только внешнее поведение ПО, без детализации структурной организации системы. Они записываются на естественном языке с использованием простых таблиц, а также наглядных диаграмм, рисунков.

Естественный язык, по существу, является неоднозначным, поэтому простота формы приводит к проблемам содержания , например:

* витиеватость стиля изложения . Иногда нелегко выразить какую- то мысль на человеческом языке ясно и недвусмысленно, не сделав при этом текст многословным и трудночитаемым.
* смешение и объединение требований. В требованиях моrут быть размыты границы между функциональными и нефункциональными требованиями. Несколько различных требований могут описываться как единое требование заказчика, и разработчик может сосредоточиться только на одном из них, «потеряв» другое.
* Шаг 1. Определение представителей заказчика
* Шаг 2. Проведение опроса представителей заказчика
* Шаг З. Документирование результатов опроса
* Шаг 4. Проверка требований

Исследования показали: исправлять ошибки требований в конце разработки системы в 100 раз дороже, чем в ходе формирования этих требований.

Многократно подтвержден следующий постулат: любые усилия, затраченные на выявление ошибок в спецификации требований, сэкономят реальные время и деньги.

Проверка требований выполняется заказчиком и разработчиком совместно, она удостоверяет:

* предметная область проекта описана корректно;
* разработчик и заказчик имеют одинаковые представления о целях системы;
* анализ внешней среды и риска разработки подтверждает возможность создания системы;
* спецификация требований верно описывает желаемую функциональность и системы, которые соответствуют потребностям заказчика и других заинтересованных лиц;
* требования полные и качественные;
* все требования согласованы друг с другом, не содержат противоречий;
* требования обеспечивают реальную возможность разработки системы.

Формирование требований является лишь начальной фазой работы с требованиями.

Анализ требований рассматривает требования заказчика как исходные данные, на выходе анализа − требования разработчика, которые справедливо называют детальнымu требованиями.

Анализ требований служит мостом между подготовкой-планированием и проектированием ПС.

Происходит переход из мира заказчика в мир разработчика.

Меняется язык записи требований. Теперь это формальный язык моделей. С помощью этих моделей можно добиться более точного отображения множества деталей, присущих ПС.

Эти модели тоже отвечают принципам построения требований: показывать, что надо делать, а не как это делается. С другой стороны, для полного описания системы требуется такая детализация, которая должна включать информацию об организации системы на уровне архитектуры (эскизе структурной организации ПС).

Типичные шаги анализа требований:

Шаг 1. Организация первичных требований

1. По режиму
2.  По категориям пользователей
3.  По объектам
4.  По свойствам
5.  По стимулам
6.  По откликам
7.  По иерархии функций

Шаг 2. Преобразование первичных требований в детальные требования

Обычно одно требование заказчика преобразуется в несколько детальных требований.

 Первичная оценка

 Обеспечение прослеживаемости требования

 Обеспечение тестируемусти требования

 Анализ однозначности толкования требования

 Назначение приоритета требования

 Проверка полноты требования

 Проверка согласованности требования с другими требованиями

 Требование заносится в спецификацию

 Описание и обоснование желаемых характеристик детального

требования

Шаг 3. Аттестация детальных требований

Аттестация (валидация) должна подтвердить, что требования действительно определяют ту систему, которая нужна заказчику.

Содержательно в состав аттестации входят:

 проверка правильности требований

 проверка на непротиворечивость

 проверка на полноту

 проверка на выполнимость

В ходе аттестации применяют следующие методы:

 Совместные проверки требований

 Макетирование

 Генерация тестов

 Автоматизированная проверка непротиворечивости

В жизненном цикле разработки программного обеспечения требования к ПС меняются. Этот

процесс объективен.

 После создания начальной версии требований приходит более глубокое понимание области ПО,

прорисовываются новые детали, не заметные ранее. Постоянство требований − это скорее исключение из общего правила.

 Чтобы изменения требований не похоронили проект раньше времени, процессом изменения требований надо управлять.

В ходе управления требованиями надо решить ряд вопросов:

 Распознавание и учет требований. Каждое требование должно быть индивидуально учтено, поскольку оно может пересекаться с другими требованиями и использоваться в оценках трассировки.

 Управление внесением изменений. Должна предусматриваться последовательность защитных действий для оценки воздействия изменения и стоимости изменения.

 Стратегия трассировки. Существуют зависимости между требованиями, а также между требованиями и проектными решениями ПС. Трассировка должна обнаруживать зависимые требования, запоминать эти зависимости и отслеживать влияние требований друг на друга и на проектные решения.

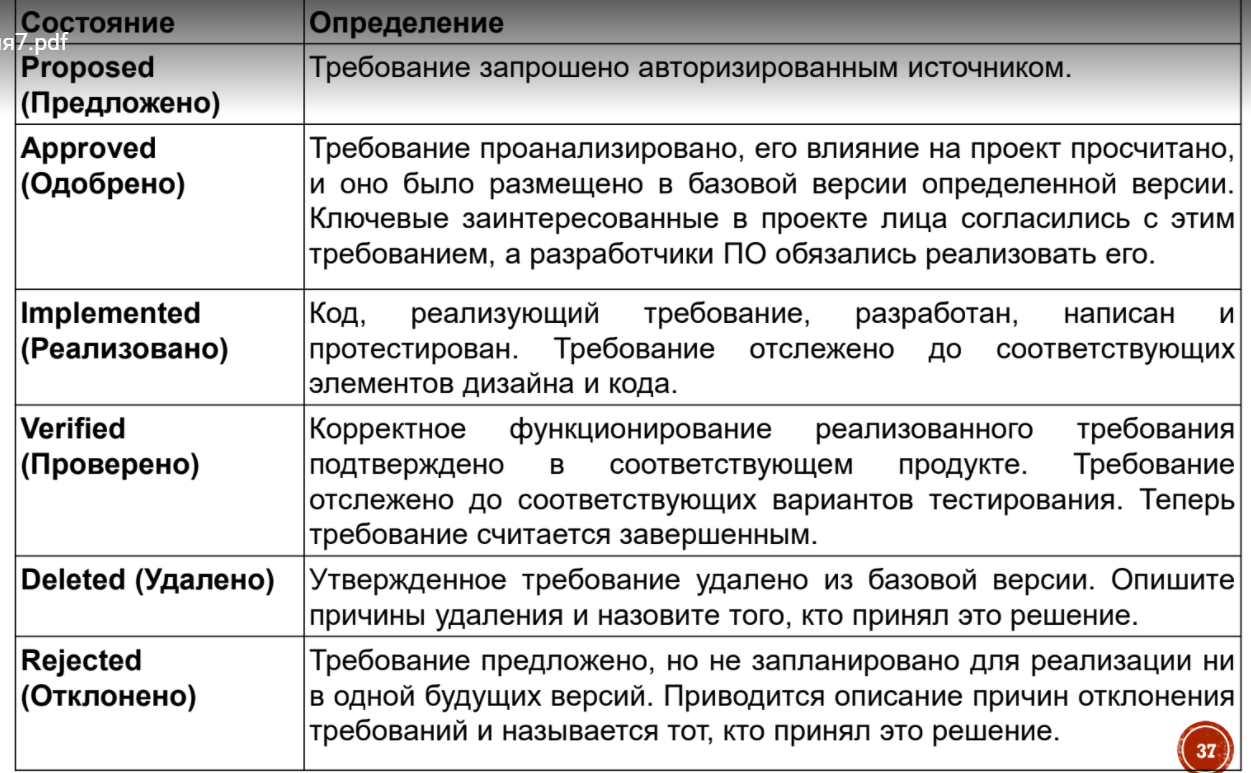
Шаги процесса управления изменениями:

Шаг 1. Распознавание проблемы. Фиксируется проблема в требованиях или прямой запрос на внесение изменения.

Проверяется обоснованность проблемы или запроса. Если обоснованность подтверждена, переходят к следующему шагу. В противном случае процесс прекращается.

 Шаг 2. Анализ изменения. При определении возможности изменения исходят из информации трассировки и общих представлений о требованиях к системе. Стоимость изменения определяется двумя параметрами: стоимостью изменения спецификации и (если это необходимо) стоимостью изменения проектного решения системы и программного кода. По окончании анализа принимается решение об изменении требования.

 Шаг 3. Выполнение изменения. Вносится изменение в спецификацию требований и, если необходимо, в проектное решение и программный код. Спецификация требований должна быть организована так, чтобы внесение изменения носило ло- кальный характер и не потребовало реорганизации всего документа. Как и в случае программ, изменяемость документов достигается минимизацией внешних ссылок и обеспечением модульности разделов. Это означает, что разделы могут изменяться без влияния на остальные части документа.

В разработке программного обеспечения при управлении требованиями рекомендуется оперировать не процентом, а статусом. 

Ошибки при документировании:

1) Описание возможных решений вместо требований.

2) Нечеткие требования, которые не допускают однозначную проверку, оставляют недосказанности, имеют оттенок советов, обсуждений, рекомендаций: "Возможно, что имеет смысл реализовать также.....", "и т.д.".

3) Игнорирование аудитории, для которой предназначено представление требований. Например, если спецификацию составляет инженер заказчика, то часто встречается переизбыток информации об оборудовании, с которым должна работать программная система, отсутствует глоссарий терминов и определений основных понятий, используются многочисленные синонимы и т.д. Или допущен слишком большой уклон в сторону программирования, что делает данную спецификацию непонятной всем непрограммистам.

4) Пропуск важных аспектов, связанных с нефункциональными требованиями, в частности, информации об окружении системы, о сроках готовности других систем, с которыми должна взаимодействовать данная. Последнее случается, например, когда данная программная система

является частью более крупного проекта. Типичны проблемы при создании программно- аппаратных систем, когда аппаратура не успевает вовремя и ПО невозможно тестировать, а в сроках и требованиях это не предусмотрено.

Управление требованиями, как и всякий другой процесс, требует ресурсов. Контроль усилий также позволяет выяснить, выполняют ли разработчики предполагаемые задачи для управления

требованиями.

Основные трудозатраты по управлению требованиями:

 предложение изменения требований и новых требований;

 оценка предложенных изменений, включая оценку влияния изменения;

 изменение работы;

 обновление документации требований или базы данных;

 сообщение об изменениях требований заинтересованным группам и отдельным лицам;

 контроль и отчет о состоянии требования;

 сбор информации о трассируемости требований.

Стандарты документации:

Обратимся сначала к ГОСТ-ам. Они расстраивают датами разработки (впрочем, похоже, за эти годы в документировании не многое изменилось).

ГОСТ Р ИСО 9127-94 «Документация пользователя и информация на упаковке для потребительских программных пакетов». Довольно кратко (весь документ – около 20 страниц)

указаны основные требования к составу и содержанию документации пользователя.

ГОСТ Р ИСО/МЭК 15910-2002 «Процесс создания документации пользователя программного средства» − стандарт больше отвечает не на вопрос «Что» должно быть в документе, а «Как» должен создаваться документ. Дополнительно есть подробное описание стиля документа с примером. ГОСТ-ы серии 19.хх – серия ЕСПД, древняя (в среднем, документы созданы в 78-м году), но зато лаконичные требования ко многим видам документов.

ГОСТ 34.602-89 «Техническое задание на создание автоматизированной системы» — стандарт на ТЗ.

IEEE Std 1063-2001 «IEEE Standard for Software User Documentation» − в документе обозначены требования к структуре, содержимому и формату инструкций пользователя.

 IEEE Std 1016-1998 «IEEE Recommended Practice for Software Design Descriptions» − рекомендации к документам, описывающим архитектуру программного обеспечения, то бишь к техническому

описанию.

 ISO/IEC FDIS 18019:2004 «Guidelines for the design and preparation of user documentation for application software» − рекомендации по созданию документации пользователя. Так сказать,

советы «хозяйке на заметку». Довольно приятное руководство с большим количеством примеров (имхо, больше подходит для чтения до или в самом начале создания документации, так как подходит к процессу основательно, от самого планирования). Также в приложениях есть чеклисты «что не забыть сделать в процессе разработки документации» и «что должно быть в документе»

 ISO/IEC 26514:2008 «Requirements for designers and developers of user documentation» − довольно свежий и, судя по содержанию, полезный документ

 ТР 2008/009/BY устанавливает требования к информационной совместимости программных и программно-технических средств платежной системы при совершении банковских операций.

 ТР 2013/027/BY - Информационные технологии. Средства защиты информации. Информационная безопасность, дата введения - 01.01.2014 Техническим регламентом устанавливаются требования к средствам защиты информации в целях защиты жизни и здоровья человека, имущества, а также предупреждения действий, вводящих в заблуждение потребителей (пользователей) относительно назначения, информационной безопасности и качества средств защиты информации. Технические кодексы установившейся практики (ТКП) устанавливают требования к процессам формирования электронных платежных документов, интерфейса пользователя и выходных документов.

 ТКП 061-2012 (07040) - Банковская деятельность. Информационные технологии. Процессы создания и воспроизведения на бумажном носителе электронных документов, используемых для совершения банковских операций. Общие требования. Требования к качеству программных продуктов определены в определены в СТБ ИСО/МЭК 12119-2003