**Этапы разработки ПО**

1. планирование,
2. [анализ требований](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7_%D1%82%D1%80%D0%B5%D0%B1%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B9),
3. [проектирование](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F),
4. [программирование](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)
5. [тестирование](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F)
6. [документирование](https://ru.wikipedia.org/wiki/%D0%94%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%86%D0%B8%D1%8F_%D0%BD%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5).

**Анализ требований**— это процесс сбора [требований к программному обеспечению (ПО)](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B5%D0%B1%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%BC%D1%83_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8E), их систематизации, документированию, анализа, выявления противоречий, неполноты, разрешения конфликтов в процессе [разработки программного обеспечения](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F). В англоязычной среде также говорят о дисциплине «инженерия требований» ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Requirements Engineering*). В процессе сбора требований важно принимать во внимание возможные противоречия требований различных заинтересованных лиц, таких как заказчики, разработчики или пользователи.

Полнота и качество анализа требований играют ключевую роль в успехе всего проекта. Требования к ПО должны быть документируемые, выполнимые, тестируемые, с уровнем детализации, достаточным для проектирования системы. Требования могут быть функциональными и нефункциональными.

Анализ требований включает три типа деятельности:

* Сбор требований: общение с клиентами и пользователями, чтобы определить, каковы их требования.
* Анализ требований: определение, являются ли собранные требования неясными, неполными, неоднозначными или противоречащими; решение этих проблем.
* Документирование требований: требования могут быть задокументированы в различных формах, таких как простое описание, [сценарии использования](https://ru.wikipedia.org/wiki/%D0%A1%D1%86%D0%B5%D0%BD%D0%B0%D1%80%D0%B8%D0%B9_%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), [пользовательские истории](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D0%BA%D0%B8%D0%B5_%D0%B8%D1%81%D1%82%D0%BE%D1%80%D0%B8%D0%B8), или спецификации процессов.

Анализ требований может быть длинным и трудным процессом, во время которого вовлечены много тонких психологических навыков. Новые системы изменяют окружающую среду и отношения между людьми, поэтому важно определить все заинтересованные лица, принять во внимание все их потребности и гарантировать, что они понимают значения новых систем. [Аналитики](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B0%D0%BB%D0%B8%D1%82%D0%B8%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F) могут использовать несколько методов, чтобы выявить следующие требования от клиента: проведение интервью, использование фокус-групп или создание списков требований. Более современные методы включают создание прототипов и сценариев использования. Где необходимо, аналитик будет использовать комбинацию этих методов, чтобы выявить точные требования заинтересованных лиц так, чтобы была создана система, которая удовлетворяет деловые потребности.

Процесс анализа требований к информационной системе включает следующие фазы:[[1]](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7_%D1%82%D1%80%D0%B5%D0%B1%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B9#cite_note-1)

* Разработка требований
  + Выявление требований
  + Анализ требований
  + Спецификация требований
  + Проверка требований
* Управление требованиями

**Выявление и анализ требований**

Определяется физическое лицо или организация формулирующая требования к [систем](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0)е или её свойства, удовлетворяющие их потребностям и ожиданиям (т.н. Стейкхолдер).

Проводятся опросы для выявления и сбора требований, при этом могут использоваться предварительно разработанные опросные листы и шаблоны параметров системы.

### Требования часто имеют сложное пересекающееся функциональное назначение, не известное отдельным стейкхолдерам. Такие требования часто упускаются или не полностью определяются во время их опросов. Для решения этой проблемы используют сеансы совместного развития требований (СРТ), где все заказчики участвуют в обсуждениях, чтобы определить требования, проанализировать их детали и выявить скрытые пересекающиеся взаимосвязи между требованиями.

## Спецификация требований

Традиционный способ документировать требования — это создание списков требований. В сложной системе такие списки требований могут занимать сотни страниц. Такие списки крайне неэффективны в современном анализе.

#### Преимущества:

* Обеспечивает контрольный список требований.
* Обеспечивает договор между заказчиками и разработчиками.
* Для большой системы может обеспечить описание высокого уровня.

#### Недостатки

* Такие списки могут занимать сотни страниц. Фактически невозможно прочитать такие документы в целом и получить чёткое понимание системы.
* Такие списки требований перечисляют отдельные требования абстрактно, оторванно друг от друга и от контекста использования
  + Эта абстракция лишает возможности видеть, как требования связываются между собой или работают вместе.
  + Эта абстракция мешает верно расположить требования по приоритетам; в то время как список, действительно облегчает приоретизацию отдельных пунктов, удаление одного пункта из контекста может сделать весь сценарий использования или деловое требование бесполезным.
  + Эта абстракция увеличивает вероятность неверного трактования требований; поскольку чем больше число людей будет их читать, тем большее будет число (различных) интерпретаций системы.
  + Эта абстракция означает, что чрезвычайно трудно убедиться, что у вас есть все необходимые требования.
* Эти списки создают ложное чувство взаимопонимания между заинтересованными лицами и разработчиками.
* Эти списки дают заинтересованным лицам ложное чувство защищённости, что разработчики должны достигнуть определённых вещей. Однако, из-за природы этих списков, они неизбежно упускают важные требования, которые будут выявлены позже в процессе. Разработчики могут использовать новые требования для пересмотра сроков и условий в их пользу.

### Альтернативы спискам требования

Альтернативами большим, предопределенным спискам требований могут служить пользовательские истории, которые определяют требования обычным языком.

### Измеримые цели

Лучшие методы рассматривают составленный список требований просто как подсказки и постоянно спрашивают «почему?», пока не будут выявлены истинные деловые цели. После этого заинтересованные лица и разработчики могут разработать тесты, измеряющие, какой уровень каждой цели был достигнут. Такие цели изменяются медленнее, чем длинный список определённых, но неизмеримых требований. Как только маленький набор критических, измеримых целей установлен, быстрое прототипирование и короткие этапы разработки могут дать заинтересованным лицам реальную ценность ещё до окончания проекта.

### Прототипы (опытные образцы)

Прототипы — макеты системы. Макеты дают возможность пользователям представить систему, которая ещё не построена. Опытные образцы помогают пользователям представить, на что будет похожа система, и облегчают пользователям принятие проектных решений, не дожидаясь окончания постройки системы. Наибольшие улучшения во взаимопонимании между пользователями и разработчиками часто замечались с введением опытных образцов. Ранние обзоры системы приводят к меньшему количеству изменений на поздних стадиях разработки и следовательно значительно уменьшают затраты.

Опытные образцы могут быть плоскими диаграммами (часто называемые каркасами) или рабочими программами, использующими синтетические функциональные возможности. Каркасы могут быть представлены графическими документами. В случаях, где законченное программное обеспечение должно иметь графическое оформление, из каркаса удаляют цвет (то есть используют серую палитру цветов). Это помогает предотвратить недоразумения по поводу окончательного вида программы.

### Сценарии использования

[Вариант использования](https://ru.wikipedia.org/wiki/%D0%92%D0%B0%D1%80%D0%B8%D0%B0%D0%BD%D1%82_%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Use Case*) — техника для документации потенциальных требований для создания новой системы или изменения существующей. Каждый вариант описывает один или несколько способов взаимодействия системы с конечным пользователем или другой системой, для достижения определённой цели. Варианты использования обычно избегают технического жаргона, предпочитая вместо этого язык конечного пользователя или эксперта в данной области. Они часто создаются совместно специалистами по сбору требований и заинтересованными лицами.

Варианты использования — наиболее важный инструмент для моделирования требований с целью спецификации функциональных возможностей разрабатываемого программного обеспечения или системы в целом. Они могут содержать дополнительное текстовое описание всех способов, которыми пользователи могут работать с программным обеспечением или системой. Такое текстовое описание называется сценарием. Как правило, варианты использования отвечают на вопрос «Что должна выполнить система для конкретного [актёра](https://ru.wikipedia.org/wiki/%D0%90%D0%BA%D1%82%D1%91%D1%80_%28UML%29) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Actor*)?», не отвечая на вопрос «Каким образом система должна это реализовать?» Текст сценария в этом случае дополняет графическое представление вариантов использования в форме описания последовательности шагов или действий, следуя которым пользователь может достичь желаемой цели при взаимодействии с системой. Полнота функциональных требований к разрабатываемой системе достигается спецификацией всех вариантов использования с соответствующими сценариями, отражающими все пожелания и потребности пользователей к разрабатываемой системе.

программы.

### Сценарии использования

[Вариант использования](https://ru.wikipedia.org/wiki/%D0%92%D0%B0%D1%80%D0%B8%D0%B0%D0%BD%D1%82_%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Use Case*) — техника для документации потенциальных требований для создания новой системы или изменения существующей. Каждый вариант описывает один или несколько способов взаимодействия системы с конечным пользователем или другой системой, для достижения определённой цели. Варианты использования обычно избегают технического жаргона, предпочитая вместо этого язык конечного пользователя или эксперта в данной области. Они часто создаются совместно специалистами по сбору требований и заинтересованными лицами.

Варианты использования — наиболее важный инструмент для моделирования требований с целью спецификации функциональных возможностей разрабатываемого программного обеспечения или системы в целом. Они могут содержать дополнительное текстовое описание всех способов, которыми пользователи могут работать с программным обеспечением или системой. Такое текстовое описание называется сценарием. Как правило, варианты использования отвечают на вопрос «Что должна выполнить система для конкретного [актёра](https://ru.wikipedia.org/wiki/%D0%90%D0%BA%D1%82%D1%91%D1%80_%28UML%29) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Actor*)?», не отвечая на вопрос «Каким образом система должна это реализовать?» Текст сценария в этом случае дополняет графическое представление вариантов использования в форме описания последовательности шагов или действий, следуя которым пользователь может достичь желаемой цели при взаимодействии с системой. Полнота функциональных требований к разрабатываемой системе достигается спецификацией всех вариантов использования с соответствующими сценариями, отражающими все пожелания и потребности пользователей к разрабатываемой системе.

### Спецификация требований программного обеспечения

[Спецификация требований программного обеспечения](https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B5%D1%86%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Software Requirements Specification, SRS*) является полным описанием поведения системы, которая будет создана. Она включает ряд сценариев использования, которые описывают все виды взаимодействия пользователей с программным обеспечением. Сценарии использования также известны как [функциональные требования](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%BF%D0%B5%D1%86%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F). В дополнении к сценариям использования, спецификация программного обеспечения также содержат нефункциональные (или дополнительные) требования. Нефункциональные требования — требования, которые налагают дополнительные ограничения на систему (такие как требования эффективности работы, стандарты качества, или проектные ограничения).

Рекомендуемые подходы для спецификации требований программного обеспечения описаны стандартом IEEE 830—1998. Этот стандарт описывает возможные структуры, желательное содержание, и качества спецификации требований программного обеспечения.

## Типы требований

Требования систематизируются несколькими способами. Ниже представлены общие классификации требований, которые касаются технического управления.

### Требования клиентов

Клиенты, это те, кто выполняет основные функции системного проектирования, со специальным акцентом на пользователе системы как ключевом клиенте. Пользовательские требования определят главную цель системы и, как минимум, ответят на следующие вопросы:

* Требования эксплуатации или развёртывания: Где система будет использоваться?
* Профиль миссии или сценарий: Как система достигнет целей миссии?
* Требования производительности: Какие параметры системы являются критическими для достижения миссии?
* Сценарии использования: Как различные компоненты системы должны использоваться?
* Требования эффективности: Насколько эффективной должна быть система для выполнения миссии?
* Эксплуатационный жизненный цикл: Как долго система будет использоваться?
* Окружающая среда: Каким окружением система должна будет эффективно управлять?

### Функциональные требования

Функциональные требования объясняют, что должно быть сделано. Они идентифицируют задачи или действия, которые должны быть выполнены. Функциональные требования определяют действия, которые система должна быть способной выполнить, связь входа/выхода в поведении системы.

### Нефункциональные требования

Нефункциональные требования — требования, которые определяют критерии работы системы в целом, а не отдельные сценарии поведения. Нефункциональные требования определяют системные свойства такие как производительность, удобство сопровождения, расширяемость, надежность, средовые факторы эксплуатации.

### Производные требования

Требования, которые подразумеваются или преобразованы из высокоуровневого требования. Например, требование для большего радиуса действия или высокой скорости может привести к требованию низкого веса.

Известные модели классификации требований включают [FURPS](https://ru.wikipedia.org/wiki/FURPS) и FURPS+, разработанные в Hewlett-Packard.

## Проблемы анализа требований

* пользователи не понимают то, что они хотят, или у пользователей нет ясного представления об их требованиях;
* пользователи не соглашаются с ранее записанными требованиями;
* пользователи настаивают на новых требованиях после того, как стоимость и график работ были установлены;
* коммуникация с пользователями является медленной;
* пользователи часто не участвуют в обзорах требований или неспособны в них участвовать;
* пользователи технически не подготовлены;
* пользователи не понимают процесса разработки ПО.

Это может привести к ситуации, где пользовательские требования продолжают изменяться, даже когда система или разработка новой продукции были начаты.

### Проблемы инженеров / разработчиков

Возможные проблемы, вызванные инженерами и разработчиками во время анализа требований:

* У технического персонала и конечных пользователей могут быть различные мнения. Следовательно, они могут неправильно полагать, что они находятся во взаимопонимании, пока готовое изделие не будет отправлено.
* Инженеры и разработчики могут попытаться подкорректировать требования чтобы они соответствовали существующей системе или модели, вместо того, чтобы разработать систему, соответствующую потребностям клиента.
* Анализ может часто выполняться инженерами или программистами, а не персоналом с навыками работы с людьми и знаниями проблемной области.

### Решения проблем

Одно из решений проблемы общения состояло в том, чтобы нанять специалистов в деловом или системном анализе.

Методики, введённые в 1990-х — прототипирование, [унифицированный язык моделирования](https://ru.wikipedia.org/wiki/UML) (UML), [сценарии использования](https://ru.wikipedia.org/wiki/%D0%A1%D1%86%D0%B5%D0%BD%D0%B0%D1%80%D0%B8%D0%B9_%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) и [гибкая методология разработки](https://ru.wikipedia.org/wiki/%D0%93%D0%B8%D0%B1%D0%BA%D0%B0%D1%8F_%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8), — также предназначены для решения описанных выше проблем.

**Проектирование программного обеспечения** — процесс создания проекта программного обеспечения (ПО), а также дисциплина, изучающая [методы проектирования](https://ru.wikipedia.org/w/index.php?title=%D0%9C%D0%B5%D1%82%D0%BE%D0%B4%D1%8B_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F&action=edit&redlink=1). Проектирование ПО является частным случаем [Проектирования](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) продуктов и процессов.

Целью проектирования является определение внутренних свойств системы и детализации её внешних (видимых) свойств на основе выданных заказчиком [требований к ПО](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B5%D0%B1%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%BC%D1%83_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8E) (исходные условия задачи). Эти требования подвергаются анализу.  
Первоначально программа рассматривается как чёрный ящик. Ход процесса проектирования и его результаты зависят не только от состава требований, но и выбранной модели процесса, опыта проектировщика.

[Модель](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C) предметной области накладывает ограничения на [бизнес-логику](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B7%D0%BD%D0%B5%D1%81-%D0%BB%D0%BE%D0%B3%D0%B8%D0%BA%D0%B0) и [структуры данных](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85).

В зависимости от класса создаваемого ПО, процесс проектирования может обеспечиваться как «ручным» проектированием, так и различными средствами его автоматизации. В процессе проектирования ПО для выражения его характеристик используются различные нотации — [блок-схемы](https://ru.wikipedia.org/wiki/%D0%91%D0%BB%D0%BE%D0%BA-%D1%81%D1%85%D0%B5%D0%BC%D0%B0), [ER-диаграммы](https://ru.wikipedia.org/wiki/ER-%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), [UML](https://ru.wikipedia.org/wiki/UML)-диаграммы, [DFD](https://ru.wikipedia.org/wiki/DFD)-диаграммы, а также макеты.

Проектированию обычно подлежат:

* [Архитектура ПО](https://ru.wikipedia.org/wiki/%D0%90%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F);
* Устройство компонентов ПО;
* [Пользовательские интерфейсы](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F).

В российской практике проектирование ведется поэтапно в соответствии со стадиями, регламентированными ГОСТ 2.103-68: Техническое задание, Техническое предложение, Эскизный проект, Технический проект, Рабочий проект. На каждом из этапов формируется свой комплект документов, называемый проектом (проектной документацией).  
В зарубежной практике регламентирующими документами, например, являются Software Architecture Document, Software Design Document.

**Программи́рование** — процесс создания [компьютерных программ](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0).

В узком смысле (так называемое **кодирование**) под программированием понимается написание инструкций (программ) на конкретном языке программирования (часто по уже имеющемуся алгоритму — плану, методу решения поставленной задачи). Соответственно, люди, которые этим занимаются, называются [программистами](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%81%D1%82) (на профессиональном жаргоне — кодерами), а те, кто разрабатывают [алгоритмы](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC) — алгоритмистами, специалистами предметной области, [математиками](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA).

В более широком смысле под программированием понимают весь спектр деятельности, связанный с созданием и поддержанием в рабочем состоянии программ — программного обеспечения ЭВМ. Иначе это называется «[программная инженерия](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%B0%D1%8F_%D0%B8%D0%BD%D0%B6%D0%B5%D0%BD%D0%B5%D1%80%D0%B8%D1%8F)» («инженерия ПО»). Сюда входят анализ и постановка задачи, проектирование программы, построение алгоритмов, разработка структур данных, написание текстов программ, отладка и тестирование программы (испытания программы), документирование, настройка (конфигурирование), доработка и сопровождение.

Программирование для ЭВМ основывается на использовании языков программирования, на которых записывается программа. Чтобы программа могла быть понята и исполнена ЭВМ, требуется специальный инструмент — [транслятор](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D1%81%D0%BB%D1%8F%D1%82%D0%BE%D1%80).

В настоящее время активно используются [интегрированные среды разработки](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8), включающие в свой состав также редактор для ввода и редактирования текстов программ, отладчики для поиска и устранения ошибок, трансляторы с различных языков программирования, компоновщики для сборки программы из нескольких модулей и другие служебные модули.

Текстовый редактор среды программирования может иметь специфичную функциональность, такую как индексация имен, отображение документации, средства визуального создания пользовательского интерфейса. С помощью текстового редактора программист производит набор и редактирования текста создаваемой программы, который называют [исходным кодом](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4). [Язык программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) определяет [синтаксис](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%29) и изначальную [семантику](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D0%BC%D0%B0%D0%BD%D1%82%D0%B8%D0%BA%D0%B0_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%29) исходного кода. [Компилятор](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%82%D0%BE%D1%80) преобразует текст программы в машинный код, непосредственно исполняемый электронными компонентами [компьютера](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80). Интерпретатор создаёт [виртуальную машину](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D1%80%D1%82%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%B0) для выполнения программы, которая полностью или частично берёт на себя функции исполнения программ.

## [Языки программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)

Единственный язык, напрямую выполняемый ЭВМ — это *машинный язык* (также называемый *машинным кодом* и *языком машинных команд*). Изначально все программы писались в машинном коде, но сейчас этого практически уже не делается. Вместо этого программисты пишут [исходный код](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) на том или ином языке программирования, затем, используя [компилятор](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%82%D0%BE%D1%80), транслируют его в один или несколько этапов в машинный код, готовый к исполнению на целевом процессоре, или в промежуточное представление, которое может быть исполнено специальным [интерпретатором](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BF%D1%80%D0%B5%D1%82%D0%B0%D1%82%D0%BE%D1%80) — виртуальной машиной. Но это справедливо только для [языков высокого уровня](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%81%D0%BE%D0%BA%D0%BE%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F). Если требуется полный низкоуровневый контроль над системой на уровне машинных команд и отдельных ячеек памяти, программы пишут на [языке ассемблера](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%B0%D1%81%D1%81%D0%B5%D0%BC%D0%B1%D0%BB%D0%B5%D1%80%D0%B0), мнемонические инструкции которого преобразуются один к одному в соответствующие инструкции машинного языка целевого процессора ЭВМ. (По этой причине трансляторы с языков ассемблера получаются алгоритмически простейшими трансляторами.)

В некоторых языках вместо машинного кода генерируется интерпретируемый двоичный код «виртуальной машины», также называемый [*байт-кодом*](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B9%D1%82-%D0%BA%D0%BE%D0%B4) (*byte-code*). Такой подход применяется в [Lisp](https://ru.wikipedia.org/wiki/Lisp), [Java](https://ru.wikipedia.org/wiki/Java), [Perl](https://ru.wikipedia.org/wiki/Perl), языках для [.NET Framework](https://ru.wikipedia.org/wiki/.NET_Framework)

**Тести́рование програ́ммного обеспе́че́ния** — процесс исследования, испытания [программного продукта](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), имеющий две различные цели:

* продемонстрировать разработчикам и заказчикам, что программа соответствует требованиям;
* выявить ситуации, в которых поведение программы является неправильным, нежелательным или не соответствующим спецификации.

Существует множество подходов к решению задачи тестирования и верификации программного обеспечения, но эффективное тестирование сложных программных продуктов — это процесс в высшей степени творческий, не сводящийся к следованию строгим и чётким процедурам или созданию таковых.

[Качество программного обеспечения](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%87%D0%B5%D1%81%D1%82%D0%B2%D0%BE_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F) можно определить как совокупную характеристику исследуемого ПО с учётом следующих составляющих:

* [надёжность](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D0%B4%D1%91%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D1%8C)
* [сопровождаемость](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%BF%D1%80%D0%BE%D0%B2%D0%BE%D0%B6%D0%B4%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F),
* практичность,
* эффективность,
* мобильность,
* функциональность.

Состав и содержание документации, сопутствующей процессу тестирования, определяется стандартом [IEEE 829](https://ru.wikipedia.org/w/index.php?title=IEEE_829&action=edit&redlink=1)-1998.

## Тестирование программного обеспечения

Существует несколько признаков, по которым принято производить классификацию видов тестирования. Обычно выделяют следующие:

По объекту тестирования

* [Функциональное тестирование](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)
* [Тестирование производительности](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%B4%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D0%B8)
  + [Нагрузочное тестирование](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D0%B3%D1%80%D1%83%D0%B7%D0%BE%D1%87%D0%BD%D0%BE%D0%B5_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)
  + [Стресс-тестирование](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D0%B5%D1%81%D1%81-%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F)
  + [Тестирование стабильности](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D1%81%D1%82%D0%B0%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D0%B8)
  + [Конфигурационное тестирование](https://ru.wikipedia.org/w/index.php?title=%D0%9A%D0%BE%D0%BD%D1%84%D0%B8%D0%B3%D1%83%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%BE%D0%B5_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit&redlink=1)
* [Юзабилити-тестирование](https://ru.wikipedia.org/wiki/%D0%AE%D0%B7%D0%B0%D0%B1%D0%B8%D0%BB%D0%B8%D1%82%D0%B8-%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)
* [Тестирование интерфейса пользователя](https://ru.wikipedia.org/w/index.php?title=%D0%A2%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81%D0%B0_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F&action=edit&redlink=1)
* [Тестирование безопасности](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%B1%D0%B5%D0%B7%D0%BE%D0%BF%D0%B0%D1%81%D0%BD%D0%BE%D1%81%D1%82%D0%B8)
* [Тестирование локализации](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BB%D0%BE%D0%BA%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%B8)
* [Тестирование совместимости](https://ru.wikipedia.org/w/index.php?title=%D0%A2%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D1%81%D0%BE%D0%B2%D0%BC%D0%B5%D1%81%D1%82%D0%B8%D0%BC%D0%BE%D1%81%D1%82%D0%B8&action=edit&redlink=1)

По знанию системы

* [Тестирование чёрного ящика](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BF%D0%BE_%D1%81%D1%82%D1%80%D0%B0%D1%82%D0%B5%D0%B3%D0%B8%D0%B8_%D1%87%D1%91%D1%80%D0%BD%D0%BE%D0%B3%D0%BE_%D1%8F%D1%89%D0%B8%D0%BA%D0%B0)
* [Тестирование белого ящика](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D0%B0%D1%82%D0%B5%D0%B3%D0%B8%D1%8F_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BF%D0%BE_%D0%BF%D1%80%D0%B8%D0%BD%D1%86%D0%B8%D0%BF%D1%83_%C2%AB%D0%91%D0%B5%D0%BB%D0%BE%D0%B3%D0%BE_%D1%8F%D1%89%D0%B8%D0%BA%D0%B0%C2%BB)
* [Тестирование серого ящика](https://ru.wikipedia.org/w/index.php?title=%D0%A2%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D1%81%D0%B5%D1%80%D0%BE%D0%B3%D0%BE_%D1%8F%D1%89%D0%B8%D0%BA%D0%B0&action=edit&redlink=1)

По степени автоматизации

* [Ручное тестирование](https://ru.wikipedia.org/w/index.php?title=%D0%A0%D1%83%D1%87%D0%BD%D0%BE%D0%B5_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit&redlink=1)
* [Автоматическое тестирование](https://ru.wikipedia.org/wiki/%D0%90%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)
* [Полуавтоматизированное тестирование](https://ru.wikipedia.org/w/index.php?title=%D0%9F%D0%BE%D0%BB%D1%83%D0%B0%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%B8%D0%B7%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit&redlink=1)

По степени изолированности компонентов

* [Модульное тестирование](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)
* [Интеграционное тестирование](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%BE%D0%B5_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)
* [Системное тестирование](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%BD%D0%BE%D0%B5_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)

По времени проведения тестирования[[*источник не указан 32 дня*]](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%BA%D0%B8%D0%BF%D0%B5%D0%B4%D0%B8%D1%8F:%D0%A1%D1%81%D1%8B%D0%BB%D0%BA%D0%B8_%D0%BD%D0%B0_%D0%B8%D1%81%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D0%BA%D0%B8)

* [Альфа-тестирование](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D1%8C%D1%84%D0%B0-%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)
  + [Дымовое тестирование](https://ru.wikipedia.org/wiki/Smoke_test) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *smoke testing*)
  + [Тестирование новой функции](https://ru.wikipedia.org/w/index.php?title=%D0%A2%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BD%D0%BE%D0%B2%D0%BE%D0%B9_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B8&action=edit&redlink=1) (*new feature testing*)
  + [Подтверждающее тестирование](https://ru.wikipedia.org/w/index.php?title=%D0%9F%D0%BE%D0%B4%D1%82%D0%B2%D0%B5%D1%80%D0%B6%D0%B4%D0%B0%D1%8E%D1%89%D0%B5%D0%B5_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit&redlink=1)
  + [Регрессионное тестирование](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D1%80%D0%B5%D1%81%D1%81%D0%B8%D0%BE%D0%BD%D0%BD%D0%BE%D0%B5_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)
  + [Приёмочное тестирование](https://ru.wikipedia.org/w/index.php?title=%D0%9F%D1%80%D0%B8%D1%91%D0%BC%D0%BE%D1%87%D0%BD%D0%BE%D0%B5_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit&redlink=1)
* [Бета-тестирование](https://ru.wikipedia.org/wiki/%D0%91%D0%B5%D1%82%D0%B0-%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)

По признаку позитивности сценариев

* [Позитивное тестирование](https://ru.wikipedia.org/w/index.php?title=%D0%9F%D0%BE%D0%B7%D0%B8%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B5_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit&redlink=1)
* [Негативное тестирование](https://ru.wikipedia.org/w/index.php?title=%D0%9D%D0%B5%D0%B3%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B5_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit&redlink=1)

По степени подготовленности к тестированию

* Тестирование по документации (формальное тестирование)
* Интуитивное тестирование ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *ad hoc testing*)

## Уровни тестирования

* [Модульное тестирование](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) — тестируется минимально возможный для тестирования компонент, например, отдельный класс или функция. Часто модульное тестирование осуществляется [разработчиками](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%81%D1%82) программного обеспечения.
* [Интеграционное тестирование](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%BE%D0%B5_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) — тестируются интерфейсы между компонентами, подсистемами или системами. При наличии резерва времени на данной стадии тестирование ведётся итерационно, с постепенным подключением последующих подсистем.
* [Системное тестирование](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%BD%D0%BE%D0%B5_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) — тестируется интегрированная система на её соответствие [требованиям](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B5%D0%B1%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%BC%D1%83_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8E).
  + **Альфа-тестирование** — имитация реальной работы с системой штатными [разработчиками](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%81%D1%82), либо реальная работа с системой потенциальными пользователями/заказчиком. Чаще всего альфа-тестирование проводится на ранней стадии разработки продукта, но в некоторых случаях может применяться для законченного продукта в качестве внутреннего приёмочного тестирования. Иногда альфа-тестирование выполняется под отладчиком или с использованием окружения, которое помогает быстро выявлять найденные ошибки. Обнаруженные ошибки могут быть переданы тестировщикам для дополнительного исследования в окружении, подобном тому, в котором будет использоваться программа.
  + [**Бета-тестирование**](https://ru.wikipedia.org/wiki/%D0%91%D0%B5%D1%82%D0%B0-%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) — в некоторых случаях выполняется распространение предварительной версии (в случае проприетарного программного обеспечения иногда с ограничениями по функциональности или времени работы) для некоторой большей группы лиц с тем, чтобы убедиться, что продукт содержит достаточно мало ошибок. Иногда бета-тестирование выполняется для того, чтобы получить обратную связь о продукте от его будущих пользователей.

Часто для свободного и открытого программного обеспечения стадия [альфа-тестирования](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D1%8C%D1%84%D0%B0-%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) характеризует функциональное наполнение кода, а [бета-тестирования](https://ru.wikipedia.org/wiki/%D0%91%D0%B5%D1%82%D0%B0-%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) — стадию исправления ошибок. При этом как правило на каждом этапе разработки промежуточные результаты работы доступны конечным пользователям.

## Статическое и динамическое тестирование

Описанные ниже техники — тестирование белого ящика и тестирование чёрного ящика — предполагают, что код исполняется, и разница состоит лишь в той информации, которой владеет тестировщик. В обоих случаях это [*динамическое тестирование*](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B0%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7_%D0%BA%D0%BE%D0%B4%D0%B0).

При [статическом тестировании](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B0%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7_%D0%BA%D0%BE%D0%B4%D0%B0) программный код не выполняется — анализ программы происходит на основе исходного кода, который вычитывается вручную, либо анализируется специальными инструментами. В некоторых случаях, анализируется не исходный, а промежуточный код (такой как [байт-код](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B9%D1%82-%D0%BA%D0%BE%D0%B4) или код на [MSIL](https://ru.wikipedia.org/wiki/MSIL)).

Также к статическому тестированию относят тестирование [требований](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B5%D0%B1%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%BC%D1%83_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8E), [спецификаций](https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B5%D1%86%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F), [документации](https://ru.wikipedia.org/wiki/%D0%94%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%86%D0%B8%D1%8F_%D0%BD%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5).

## Регрессионное тестирование

После внесения изменений в очередную версию программы, регрессионные тесты подтверждают, что сделанные изменения не повлияли на работоспособность остальной функциональности приложения. Регрессионное тестирование может выполняться как вручную, так и средствами [автоматизации тестирования](https://ru.wikipedia.org/wiki/%D0%90%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5).

## Тестовые сценарии

Тестировщики используют тестовые сценарии на разных уровнях: как в модульном, так и в интеграционном и системном тестировании. Тестовые сценарии, как правило, пишутся для проверки компонентов, в которых наиболее высока вероятность появления отказов или вовремя не найденная ошибка может быть дорогостоящей.

## Тестирование «белого ящика» и «чёрного ящика»

В зависимости от доступа разработчика тестов к исходному коду тестируемой программы различают «тестирование белого ящика» и «тестирование чёрного ящика».

При тестировании белого ящика (также говорят — *прозрачного ящика*), разработчик теста имеет доступ к исходному коду программ и может писать код, который связан с библиотеками тестируемого программного обеспечения. Это типично для модульного тестирования, при котором тестируются только отдельные части системы. Оно обеспечивает то, что компоненты конструкции — работоспособны и устойчивы, до определённой степени. При тестировании белого ящика используются метрики [покрытия кода](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BA%D1%80%D1%8B%D1%82%D0%B8%D0%B5_%D0%BA%D0%BE%D0%B4%D0%B0) или [мутационное тестирование](https://ru.wikipedia.org/wiki/%D0%9C%D1%83%D1%82%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%BE%D0%B5_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5).

При тестировании чёрного ящика, тестировщик имеет доступ к программе только через те же [интерфейсы](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81), что и заказчик или пользователь, либо через внешние интерфейсы, позволяющие другому компьютеру либо другому процессу подключиться к системе для тестирования. Например, тестирующий модуль может виртуально нажимать клавиши или кнопки мыши в тестируемой программе с помощью механизма взаимодействия процессов, с уверенностью в том, все ли идёт правильно, что эти события вызывают тот же отклик, что и реальные нажатия клавиш и кнопок мыши. Как правило, тестирование чёрного ящика ведётся с использованием спецификаций или иных документов, описывающих требования к системе. Обычно в данном виде тестирования [критерий покрытия](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D1%82%D0%B5%D1%80%D0%B8%D0%B9_%D1%82%D0%B5%D1%81%D1%82%D0%BE%D0%B2%D0%BE%D0%B3%D0%BE_%D0%BF%D0%BE%D0%BA%D1%80%D1%8B%D1%82%D0%B8%D1%8F) складывается из покрытия структуры входных данных, [покрытия требований](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BA%D1%80%D1%8B%D1%82%D0%B8%D0%B5_%D1%82%D1%80%D0%B5%D0%B1%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B9) и покрытия модели (в [тестировании на основе моделей](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BD%D0%B0_%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D0%B5_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D0%B5%D0%B9)).

При тестировании серого ящика разработчик теста имеет доступ к исходному коду, но при непосредственном выполнении тестов доступ к коду, как правило, не требуется.

Если «альфа-» и «бета-тестирование» относятся к стадиям до выпуска продукта (а также, неявно, к объёму тестирующего сообщества и ограничениям на методы тестирования), тестирование «белого ящика» и «чёрного ящика» имеет отношение к способам, которыми тестировщик достигает цели.

**Бета-тестирование** в целом ограничено техникой чёрного ящика (хотя постоянная часть тестировщиков обычно продолжает тестирование белого ящика параллельно бета-тестированию). Таким образом, термин «бета-тестирование» может указывать на состояние программы (ближе к выпуску чем «альфа»), или может указывать на некоторую группу тестировщиков и процесс, выполняемый этой группой. То есть, тестировщик может продолжать работу по тестированию белого ящика, хотя программа уже «бета-стадии», но в этом случае он не является частью «бета-тестирования».

[**Покрытие кода**](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BA%D1%80%D1%8B%D1%82%D0%B8%D0%B5_%D0%BA%D0%BE%D0%B4%D0%B0)

Покрытие кода, по своей сути, является тестированием методом белого ящика. Тестируемое программное обеспечение собирается со специальными настройками или библиотеками или запускается в особом окружении, в результате чего для каждой используемой (выполняемой) функции программы определяется местонахождение этой функции в исходном коде. Этот процесс позволяет разработчикам и специалистам по обеспечению качества определить части системы, которые, при нормальной работе, используются очень редко или никогда не используются (такие как код обработки ошибок и т. п.). Это позволяет сориентировать тестировщиков на тестирование наиболее важных режимов.

Тестировщики могут использовать результаты теста покрытия кода для разработки тестов или тестовых данных, которые расширят покрытие кода на важные функции.

Как правило, инструменты и библиотеки, используемые для получения покрытия кода, требуют значительных затрат производительности и/или памяти, недопустимых при нормальном функционировании ПО. Поэтому они могут использоваться только в лабораторных условиях.

**Документирование**

**Документа́ция на программное обеспечение** — печатные руководства пользователя, диалоговая (оперативная) документация и справочный текст, описывающие, как пользоваться программным продуктом[[1]](https://ru.wikipedia.org/wiki/%D0%94%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%86%D0%B8%D1%8F_%D0%BD%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5#cite_note-.D0.93.D0.9E.D0.A1.D0.A215910-1).

[Документ](https://ru.wikipedia.org/wiki/%D0%94%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82) — элемент документации: целевая информация, предназначенная для конкретной аудитории, размещенная на конкретном носителе (например, в книге, на диске, в краткой справочной карте) в заданном формате[[1]](https://ru.wikipedia.org/wiki/%D0%94%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%86%D0%B8%D1%8F_%D0%BD%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5#cite_note-.D0.93.D0.9E.D0.A1.D0.A215910-1).

Программный документ — документ, содержащий в зависимости от назначения данные, необходимые для разработки, производства, эксплуатации, сопровождения программы или программного средства

## Типы документации

Существует четыре основных типа документации на ПО:

* архитектурная/проектная — обзор программного обеспечения, включающий описание рабочей среды и принципов, которые должны быть использованы при создании ПО
* техническая — документация на [код](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4), [алгоритмы](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC), интерфейсы, [API](https://ru.wikipedia.org/wiki/API)
* пользовательская — руководства для конечных пользователей, администраторов системы и другого персонала
* маркетинговая

Проектная документация обычно описывает продукт в общих чертах. Не описывая того, как что-либо будет использоваться, она скорее отвечает на вопрос «почему именно так». Например, в проектном документе программист может описать обоснование того, почему структуры данных организованы именно таким образом. Описываются причины, почему какой-либо [класс](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%29) сконструирован определённым образом, выделяются [паттерны](https://ru.wikipedia.org/wiki/%D0%A8%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD%D1%8B_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), в некоторых случаях даже даются идеи как можно будет выполнить улучшения в дальнейшем. Ничего из этого не входит в техническую или пользовательскую документацию, но всё это действительно важно для проекта.

### [Техническая документация](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%85%D0%BD%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%B4%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%86%D0%B8%D1%8F)

Подобная документация имеет сильно выраженный технический характер и в основном используется для определения и описания [API](https://ru.wikipedia.org/wiki/API), [структур данных](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) и [алгоритмов](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC).

Часто при составлении технической документации используются автоматизированные средства — [генераторы документации](https://ru.wikipedia.org/wiki/%D0%93%D0%B5%D0%BD%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80_%D0%B4%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%86%D0%B8%D0%B8), такие как [Doxygen](https://ru.wikipedia.org/wiki/Doxygen), [javadoc](https://ru.wikipedia.org/wiki/Javadoc), [NDoc](https://ru.wikipedia.org/w/index.php?title=NDoc&action=edit&redlink=1) и другие. Они получают информацию из специальным образом оформленных комментариев в исходном коде, и создают справочные руководства в каком-либо формате, например, в виде текста или [HTML](https://ru.wikipedia.org/wiki/HTML).

Использование генераторов документации и документирующих комментариев упрощает поддержку документации в актуальном состоянии. При таком подходе документация является частью исходного кода, и одни и те же инструменты могут использоваться для сборки программы и одновременной сборки документации к ней.

### Пользовательская документация

Обычно, пользовательская документация представляет собой [руководство пользователя](https://ru.wikipedia.org/wiki/%D0%A0%D1%83%D0%BA%D0%BE%D0%B2%D0%BE%D0%B4%D1%81%D1%82%D0%B2%D0%BE_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F), которое описывает каждую функцию программы, а также шаги, которые нужно выполнить для использования этой функции. Хорошая пользовательская документация идёт ещё дальше и предоставляет инструкции о том что делать в случае возникновения проблем.

Руководство должно иметь чёткую структуру; очень полезно, если имеется сквозной [предметный указатель](https://ru.wikipedia.org/w/index.php?title=%D0%9F%D1%80%D0%B5%D0%B4%D0%BC%D0%B5%D1%82%D0%BD%D1%8B%D0%B9_%D1%83%D0%BA%D0%B0%D0%B7%D0%B0%D1%82%D0%B5%D0%BB%D1%8C&action=edit&redlink=1). Логическая связность и простота также имеют большое значение.

Существует три подхода к организации пользовательской документации. Вводное руководство ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *tutorial*), наиболее полезное для новых пользователей, последовательно проводит по ряду шагов, служащих для выполнения каких-либо типичных задач. Тематический подход, при котором каждая глава руководства посвящена какой-то отдельной теме, больше подходит для совершенствующихся пользователей. В последнем, третьем подходе, команды или задачи организованы в виде алфавитного справочника — часто это хорошо воспринимается продвинутыми пользователями, хорошо знающими, что они ищут. Жалобы пользователей обычно относятся к тому, что документация охватывает только один из этих подходов, и поэтому хорошо подходит лишь для одного класса пользователей.

Во многих случаях разработчики программного продукта ограничивают набор пользовательской документации лишь встроенной системой помощи ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *online help*), содержащей справочную информацию о командах или пунктах меню.

### Маркетинговая документация

Для многих приложений необходимо располагать рядом с ними рекламные материалы, с тем чтобы заинтересовать людей, обратив их внимание на продукт. Такая форма документации имеет целью:

* подогреть интерес к продукту у потенциальных пользователей
* информировать их о том, что именно делает продукт, с тем чтобы их ожидания совпадали с тем, что они получат
* объяснить положение продукта по сравнению с конкурирующими решениями

Одна из хороших маркетинговых практик — предоставление [слогана](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D0%BE%D0%B3%D0%B0%D0%BD) — простой запоминающейся фразы, иллюстрирующей то, что мы хотим донести до пользователя, а также характеризующей *ощущение*, которое создаёт продукт.

Часто бывает так, что коробка продукта и другие маркетинговые материалы дают более ясную картину о возможностях и способах использования программы, чем всё остальное.