**Федеральное государственное автономное**

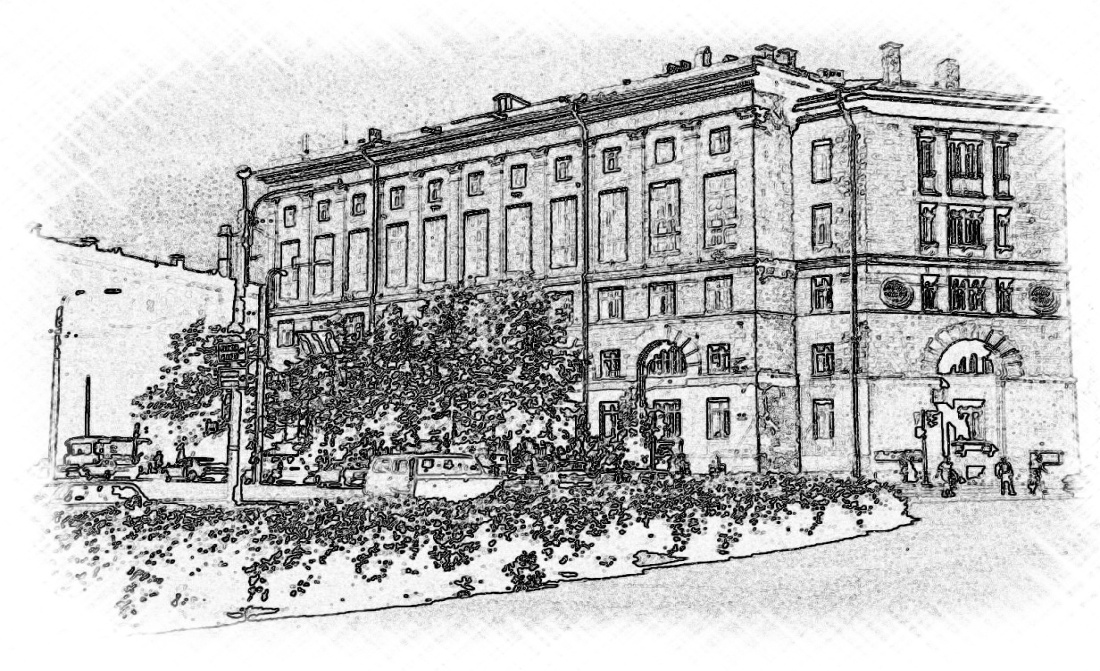
**образовательное учреждение высшего образования**

**«Санкт-Петербургский политехнический университет Петра Великого»**

**Институт среднего профессионального образования**

**Конспект лекций по предмету:**

**«Стандартизация, сертификация и техническая документация»**



Санкт-Петербург

2024

СОДЕРЖАНИЕ

[Введение 3](#_Toc417895518)

[Тема 1. Общие положения о стандартах 4](#_Toc417895519)

[Тема 2. Нормативная база в области документирования ПС 8](#_Toc417895520)

[Тема 3. Процессы жизненного цикла программных средств ГОСТ 19.102-77 13](#_Toc417895521)

[Тема 4. Процессы жизненного цикла программных средств ГОСТ Р ИСО/МЭК 12207 18](#_Toc417895522)

[Тема 5. Виды программных документов. Требования к программным документам 32](#_Toc417895523)

[Тема 6. Техническое задание 37](#_Toc417895524)

[Тема 7. Текст программы. Описание программы 49](#_Toc417895525)

[Тема 8. Пояснительная записка 53](#_Toc417895526)

[Тема 9. Руководство Оператора 55](#_Toc417895527)

[Тема 10. Схемы алгоритмов, программ, данных и систем. 60](#_Toc417895528)

[Тема 11. Порядок и методика испытаний 76](#_Toc417895529)

[Тема 12. Метрология 84](#_Toc417895530)

[Тема 13. Качество программных средств. 89](#_Toc417895531)

[Тема 14. Методы оценки технико-экономических показателей ПС 103](#_Toc417895532)

[Тема 15. Сертификация 111](#_Toc417895533)

[Тема 16. Интеллектуальная собственность и авторские права 120](#_Toc417895534)

[Использованная литература 122](#_Toc417895535)

# Введение

Данный курс лекций разработан для раздела «МДК 03.03 Документирование и сертификация» входящий в состав профессионального модуля «ПМ.03Участие в интеграции программных модулей». Студент в ходе освоения данной дисциплины должен

уметь:

* использовать методы для получения кода с заданной функциональностью и степенью качества;

знать:

* основы верификации и аттестации программного обеспечения;
* стандарты качества программного обеспечения;
* основные положения метрологии программных продуктов, принципы построения, проектирования и использования средств для измерений характеристик и параметров программ, программных систем и комплексов;
* методы и средства разработки программной документации.

# Общие положения о стандартах

Стандартизация — это деятельность, направленная на разработку и установление требований, норм, правил, характеристик, как обязательных для выполнения, так и рекомендуемых, обеспечивающая право потребителя на приобретение товаров надлежащего качества, а также право на безопасность и комфортность труда. Цель стандартизации — достижение оптимальной степени упорядочения в той или иной области посредством широкого и многократного использования установленных положений, требований, норм для решения реально существующих, планируемых или потенциальных задач. Основными результатами деятельности по стандартизации должны быть повышение степени соответствия продукта (услуги), процессов их функциональному назначению, устранение технических барьеров в международном товарообмене, содействие научно-техническому прогрессу и сотрудничеству в различных областях.

Стандартизация связана с такими понятиями, как объект стандартизации и область стандартизации. Объектом стандартизации обычно называют продукцию, процесс, услугу, для которых разрабатывают те или иные требования, характеристики, параметры, правила и т.п. Стандартизация может касаться либо объекта в целом, либо его отдельных составляющих (характеристик). Областью стандартизации называют совокупность взаимосвязанных объектов стандартизации.

Стандартизация осуществляется на разных уровнях. *Уровень стандартизации* зависит от того, участники какого географического, экономического, политического региона мира принимают стандарт. Если участие в стандартизации открыто для соответствующих органов любой страны, то это международная стандартизация. Региональная стандартизация — деятельность, открытая только для соответствующих органов государств одного географического, политического или экономического региона. Региональная и международная стандартизация осуществляется специалистами стран, представленных в соответствующих региональных и международных организациях (Рис. **1**).



Рис. 1 Схема уровней стандартизации

Национальная стандартизация — стандартизация в одном конкретном государстве. При этом национальная стандартизация также может осуществляться на разных уровнях: на государственном, отраслевом, в том или ином секторе экономики (например, на уровне министерств), на уровне ассоциаций, производственных фирм, предприятий (фабрик, заводов) и учреждений.

Стандартизацию, которая проводится в административно-территориальной единице (провинции, крае и т.п), принято называть *административно-территориальной стандартизацией.*

**Нормативные документы по стандартизации и виды стандартов**

В процессе стандартизации вырабатываются нормы, правила, требования, характеристики, касающиеся объекта стандартизации, которые оформляются в виде нормативного документа.

Рассмотрим разновидности нормативных документов, которые рекомендуются руководством 2-й Международной организации по стандартизации и Международной электротехнической комиссии (ИСО/МЭК), а также принятые в государственной системе стандартизации Российской Федерации (РФ). Руководство ИСО/МЭК рекомендует: стандарты, документы технических условий, своды правил, регламенты (технические регламенты), положения (Рис. 2).



Рис. 2 Схема разновидностей нормативных документов

***Стандарт*** (от англ. standard — норма, образец) — в широком смысле слова образец, эталон, модель, принимаемые за исходные для сопоставления с ними других подобных объектов. Стандарт как нормативно-технический документ устанавливает комплекс норм, правил, требований к объекту стандартизации. Стандарт может быть разработан как на материальные предметы (продукцию, эталоны, образцы веществ), так и на нормы, правила, требования в различных областях. В переносном смысле — шаблон, трафарет, не содержащий ничего оригинального.

***Стандарт*** — это нормативный документ, разработанный на основе консенсуса, утвержденный признанным органом, направленный на достижение оптимальной степени упорядочения в определенной области. В стандарте устанавливаются для всеобщего и многократного использования общие принципы, правила, характеристики, касающиеся различных видов деятельности или их результатов. Стандарт должен быть основан на обобщенных результатах научных исследований, технических достижений и практического опыта., тогда его использование принесет оптимальную выгоду для общества.

***Предварительный стандарт*** — это временный документ, который принимается органом по стандартизации и доводится до широкого круга потенциальных потребителей, а также до тех, кто может его применить. Информация, полученная в процессе использования предварительного стандарта, и отзывы об этом документе служат базой для решения вопроса о целесообразности принятия стандарта.

Стандарты бывают международными, региональными, национальными, административно-территориальными. Они принимаются соответственно международными, региональными, национальными, территориальными органами по стандартизации. Все эти категории стандартов предназначены для широкого круга потребителей. По существующим нормам стандартизации стандарты периодически пересматриваются для внесения изменений, чтобы их требования соответствовали уровню научно-технического прогресса, или, согласно терминологии ИСО/МЭК, стандарты должны представлять собой «признанные технические правила». Нормативный документ, в том числе и стандарт, считается признанным техническим правилом, если он разработан в сотрудничестве с заинтересованными сторонами путем консультаций и на основе консенсуса.

*Документ технических условий* (technical specification) устанавливает технические требования к продукции, услуге, процессу. Обычно в документе технических условий должны быть указаны методы или процедуры, которые следует использовать для проверки соблюдения требований данного нормативного документа в таких ситуациях, когда это необходимо.

*Свод правил,* как и предыдущий нормативный документ, может быть самостоятельным стандартом либо самостоятельным документом, а также частью стандарта. Свод правил обычно разрабатывается для процессов проектирования, монтажа оборудования и конструкций, технического обслуживания или эксплуатации объектов, конструкций, изделий. Технические правила, содержащиеся в документе, носят рекомендательный характер.

Все указанные выше нормативные документы являются рекомендательными. В отличие от их регламент носит обязательный характер. *Регламент —* это документ, в котором содержатся обязательные правовые нормы.

**Организации, разрабатывающие стандарты**

**Международная организация по стандартизации (ИСО)**

Международная организация по стандартизации создана в 1946 г. двадцатью пятью национальными организациями по стандартизации.

При создании организации и выборе ее названия учитывалась необходимость того, чтобы аббревиатура наименования звучала одинаково на всех языках. Для этого было решено использовать греческое слово «isos» — равный. Вот почему на всех языках мира Международная организация по стандартизации имеет краткое название ISO (ИСО).

Сфера деятельности ИСО касается стандартизации во всех областях, кроме электротехники и электроники, относящихся к компетенции Международной электротехнической комиссии (МЭК). Некоторые виды работ выполняются совместными усилиями этих организаций. Кроме стандартизации ИСО занимается и проблемами сертификации.

ИСО определяет свои задачи следующим образом: содействие развитию стандартизации и смежных видов деятельности в мире с целью обеспечения международного обмена товарами и услугами, а также развития сотрудничества в интеллектуальной, научно-технической и экономической областях.

Вопросы информационной технологии, микропроцессорной техники и т.п. входят в область совместных разработок ИСО/МЭК. В последние годы ИСО уделяет много внимания стандартизации систем обеспечения качества. Практическим результатом усилий в этих направлениях являются разработка и издание международных стандартов. При их разработке ИСО учитывает ожидания всех заинтересованных сторон — производителей продукции (услуг), потребителей, правительственных кругов, научно-технических и общественных организаций.

На сегодняшний день в состав ИСО входят 120 стран своими национальными организациями по стандартизации. Россию представляет Госстандарт РФ в качестве комитета — члена ИСО.

**Международная электротехническая комиссия (МЭК)**

Международная электротехническая комиссия создана на международной конференции, в работе которой участвовали 13 стран, в наибольшей степени заинтересованных в такой организации. Датой начала международного сотрудничества по электротехнике считается 1881 г., когда состоялся первый Международный конгресс по электричеству. Позже, в 1904 г., правительственные делегаты конгресса решили, что необходима специальная организация, которая бы занималась стандартизацией параметров электрических машин и терминологией в этой области.

После второй мировой войны, когда была создана ИСО, МЭК стала автономной организацией в ее составе. МЭК занимается стандартизацией в области электротехники, электроники, радиосвязи, приборостроения. Эти области не входят в сферу деятельности ИСО.

**Объединенный технический комитет(JTC1)**

В 1987 г. ИСО и МЭК объединили свою деятельность в области стандартизации информационных технологий (ИТ), создав единый орган JTC1 (Joint Technical Committee 1 — Объединенный технический комитет 1), предназначенный для формирования всеобъемлющей системы базовых стандартов в области ИТ и их расширений для конкретных сфер деятельности.

JTC1 имеет 17 подкомиссий, чья работа покрывает все: от техники программного обеспечения до языков программирования, компьютерной графики и обработки изображения, соединения оборудования, методов защиты и т.д. Работа над стандартами ИТ в JTC1 тематически распределена по подкомитетам (Subcommittees — SC). В дополнение создана специальная группа по функциональным стандартам (Special Group on Functional Standards — SGFS) для обработки предложений по международным стандартизованным профилям (International Standardized Profiles — ISPs), представляющим определения профилей ИТ.

**Государственный комитет РФ по стандартизации**

Национальным органом по стандартизации в России является Государственный комитет Российской Федерации по стандартизации и метрологии (Госстандарт России). Это федеральный орган исполнительной власти, осуществляющий межотраслевую координацию, а также функциональное регулирование в области стандартизации, метрологии и сертификации.

Государственный комитет Российской Федерации по стандартизации и метрологии — правопреемник упраздненного Министерства промышленности и торговли Российской Федерации в отношении функций по реализации государственной политики в сфере стандартизации, метрологии и сертификации. Государственный комитет Российской Федерации по стандартизации и метрологии — специально уполномоченный федеральный орган исполнительной власти в области сертификации. Председатель Государственного комитета Российской Федерации по стандартизации и метрологии является главным государственным инспектором Российской Федерации по надзору за государственными стандартами и обеспечением единства измерений.

В ведении Государственного комитета Российской Федерации по стандартизации и метрологии находятся службы по надзору за государственными стандартами и обеспечением единства измерений, а также центры стандартизации, метрологии и сертификации, предприятия, учреждения, учебные заведения и иные организации.

**Внутрифирменные (внутрикорпоративные)стандарты**

Внутрифирменные стандарты действуют внутри организации — разработчика программного обеспечения или любой другой компании, связанной с информационными технологиями. Такие стандарты, как правило, регламентируют порядок оформления документации, приказов и технической литературы внутри компании, пользовательский интерфейс разрабатываемых приложений (например, запрет на использование некоторых элементов интерфейса), стиль программирования, спецификацию модулей, имена используемых переменных, таблиц баз данных (БД). Внутрикорпоративные (внутрифирменные) стандарты имеют узкую сферу полномочий (одна или несколько фирм), но играют большую роль, так как они абсолютно конкретны.

**Назначение и классификация внутрикорпоративных стандартов**

Как уже отмечалось выше, внутрифирменные (внутрикорпоративные) стандарты занимают особое место в классификации стандартов. Это связано с тем, что они регламентируют технологические процессы, происходящие внутри фирмы (например, процессы анализа, кодирования, тестирования), они максимально конкретны и детализируют уровень мероприятий, если пользоваться управленческой терминологией.

Можно ли разработать универсальный стандарт и тиражировать его на различных предприятиях? Увы, нет. Существуют общие подходы, известны технологии разработки внутрикорпоративных стандартов, но всякий раз этот процесс уникален, поскольку не существует двух совершенно одинаковых предприятий — они различаются отраслевой спецификой, размерами, стратегией, организационной структурой и многими другими факторами. Кроме того, документы, особенно относящиеся к внутреннему документообороту, различаются в силу устоявшихся бизнес-правил, традиций, корпоративной культуры, отношений между подразделениями.

Внутрикорпоративные стандарты, разработанные для одного предприятия, не подойдут для другого. Поэтому типового внутрикорпоративного стандарта просто не может быть. При этом следует различать структуру бизнес-процессов, которая действительно может быть типовой, и внутрикорпоративный стандарт, согласующий структуру бизнес-процессов и организационную структуру конкретного предприятия.

**Для чего нужны внутрифирменные стандарты.**

Какую пользу дают внутрифирменные стандарты? С их помощью:

* достигаются лучшие показатели обучения персонала. Соответственно проще заменить человека в случае его увольнения. Отсюда следует, что можно брать на работу специалистов более низкой квалификации и доучивать их на месте без серьезных затрат для фирмы;
* повышаются надежность и качество программного обеспечения;
* повышается дружественность программного продукта, сокращаются сроки обучения конечного пользователя;
* улучшается обслуживание, сокращаются сроки внедрения программного продукта.

Внутрифирменные стандарты обычно создаются самыми квалифицированными людьми в своей области, которые хорошо знают разрабатываемый программный продукт, владеют методологией и богатой практикой создания программных средств, а также людьми, как правило, руководителями проекта или направлений, которые видят картину в целом, управляют процессом производства программного средства непосредственно и имеют связь с конечным пользователем.

**Вопросы для самопроверки**

1. Дайте определение понятию «стандартизация».
2. Охарактеризуйте основные уровни стандартизации.
3. Назовите основные виды нормативных документов.
4. Дайте определение понятию «стандарт».
5. Назовите известные вам международные организации, разрабаты вающие стандарты.
6. Объясните, почему нужны внутрифирменные стандарты.

# Нормативная база в области документирования ПС

Создание программной документации — важный этап, так как пользователь начинает свое знакомство с программным продуктом именно с документации. Для чего предназначен программный продукт, как установить программный продукт, как начать с ним работать — вот одни из первых вопросов, на которые должна отвечать программная документация (Installation Guide, Getting Started). Составлением программной документации обычно занимаются специальные люди — технические писатели (иногда программную документацию пишут сами программисты или аналитики). Этот этап является самым неприятным и тяжелым в программистской работе. К сожалению, обычно этому либо не учат совсем, либо в лучшем случае не обращают на качество получаемых документов должного внимания. Тем не менее владение этим искусством является одним из важнейших факторов, определяющих качество программиста. Умение создавать программную документацию определяет профессиональный уровень программиста. Заказчик не будет вникать в тонкости и особенности даже самой замечательной программы. Заказчик будет сначала читать документацию. Большую роль играет в этом и психологический фактор.

Грамотно составленный (точнее, созданный) пакет программной документации избавит вас от многих неприятностей. В частности, избавиться от назойливых вопросов и необоснованных претензий можно, просто отослав пользователя к документации. Это касается прежде всего важнейшего документа —Технического задания. Можно напомнить о многомиллионном иске к компании IBM, который предъявило одно крупное издательство, не удовлетворенное качеством вычислительной техники и программного обеспечения. IBM суд выиграла только благодаря тому, что предъявила подписанное обеими сторонами Техническое задание. Было это давно, еще в 70-х годах 20 века, однако сути дела это не меняет. На Западе важность программной документации поняли давно, вместе с программным обеспечением поставляется целый пакет документации. Вообще программную документацию можно разделить по отношению к пользователю на внутреннюю и внешнюю. Внешняя — всевозможные руководства для пользователей, техническое задание, справочники; внутренняя документация — та, которая используется в процессе разработки программного обеспечения и недоступна конечному пользователю (различные внутренние стандарты, комментарии исходного текста, технологии программирования и т.д.)

Когда программист-разработчик получает в той или иной форме задание на программирование, перед ним, перед руководителем проекта и перед всей проектной группой встают вопросы: что должно быть сделано, кроме собственно программы? что и как должно быть оформлено в виде документации? что передавать пользователям, а что — службе сопровождения? как управлять всем этим процессом?

Кроме упомянутых вопросов есть и другие, например, что должно входить в само задание на программирование?

На эти и массу других вопросов отвечали государственные стандарты на программную документацию — комплекс стандартов 19-й серии ГОСТ ЕСПД. Но уже тогда у программистов была масса претензий к этим стандартам. Что-то требовалось дублировать в документации много раз (как, казалось — неоправданно), а многое не было предусмотрено, как, например, отражение специфики документирования программ, работающих с интегрированной базой данных.

Прошло много лет, программирование происходит в среде совершенно новых технологий, многие программисты, работая в стиле drag-and-drop, могут годами не видеть текст своих программ. Это не значит, что исчезла необходимость в их документировании. Более того, вопросы о наличии хоть какой-то системы, регламентирующей эту сторону создания программных средств (ПС), продолжают задавать постоянно. Спрашивают и о том, есть ли обязательные для применения стандарты (особенно остро стоит этот вопрос, когда разработка выполняется по заказу государственной организации или предприятия). Интересуются и тем, где можно купить имеющиеся стандарты.

Попробуем ответить на некоторые из этих вопросов и дать свое представление о том, как целесообразно использовать существующие стандарты и развивать систему стандартов. Будем говорить только о стандартах на документирование ПС, не касаясь стандартов на разработку автоматизированных систем в целом или более специфических стандартов, например, направленных на стандартизацию языков программирования.

**Комплекс стандартов ГОСТ 19 (ЕСПД) - единая система программной документации** — комплекс государственных стандар­тов, устанавливающих взаимоувязанные правила разработки, оформ­ления и обращения программ и программной документации.

Основу отечественной нормативной базы в области документи­рования ПС составляет комплекс стандартов Единой системы про­граммной документации (ЕСПД). Основная и большая часть комплекса ЕСПД была разработана в 70-е и 80-е годы. Сейчас этот ком­плекс представляет собой систему межгосударственных стандартов стран СНГ (ГОСТ), действующих на территории Российской Федера­ции на основе межгосударственного соглашения по стандартизации.

Стандарты ЕСПД в основном охватывают ту часть документа­ции, которая создается в процессе разработки ПС, и связаны по боль­шей части с документированием функциональных характеристик ПС. Следует отметить, что стандарты ЕСПД (ГОСТ 19) носят рекоменда­тельный характер. Впрочем, это относится и ко всем другим стандар­там в области ПС (ГОСТ 34, Международному стандарту 18О/1ЕС и др.). Дело в том, что в соответствии с Законом РФ «О стандартиза­ции» эти стандарты становятся обязательными на контрактной осно­ве, то есть при ссылке на них в договоре на разработку (поставку) ПС. Говоря о состоянии ЕСПД в целом, можно констатировать, что большая часть стандартов ЕСПД морально устарела. К числу основ­ных недостатков **ЕСПД** можно отнести:

* ориентацию на единственную, «каскадную» модель жизненного цикла (ЖЦ) ПС;
* отсутствие четких рекомендаций по документированию характе­ристик качества ПС;
* отсутствие системной увязки с другими действующими отечест­венными системами стандартов по ЖЦ и документированию продук­ции в целом, например ЕСКД;
* нечетко выраженный подход к документированию ПС как товар­ной продукции;
* отсутствие рекомендаций по самодокументированию ПС, напри­мер, в виде экранных меню и средств оперативной помощи пользова­телю («хелпов»);
* отсутствие рекомендаций по составу, содержанию и оформлению перспективных документов на ПС, согласованных с рекомендациями международных и региональных стандартов.

Из этого выходит, что ЕСПД нуждается в полном пересмотре на основе стандарта ИСО/МЭК 12207-95, (об этом стандарте далее будет рассказано более подробнее).

Стоит заметить что, кроме ЕСПД  в официальной нормативной базе РФ в области документирования программ и в смежных областях есть  ряд перспективных стандартов, например:

* международный стандарт ISO/IEC 12207: 1995-08-01 на организацию жизненного цикла продуктов программного обеспечения.
* Стандарты комплекса ГОСТ 34 на создание и развитие автоматизированных систем.

Даже не смотря на свои недостатки, многие стандарты ЕСПД могут с пользой применяться для документирования программ  по тому что:

* стандарты ЕСПД вносят характерный порядок в процесс документирования программ;
* предусмотренный стандартами ЕСПД состав программных документов  возможно видоизменять  внося в комплект документации  дополнительные виды документов.
* стандарты ЕСПД позволяют  изменять структуру и содержание исходя из конкретных требований заказчика и пользователя.

При этом заказчик и руководитель проекта выбирают уместное в проекте подмножество стандартов и документации, дополняют их нужными разделами (исключают ненужные) и  привязывают создание этих документов к той схеме,  которая используется в проекте.

**Стандарты ЕСПД  подразделяют на группы, приведенные в таблице:**

|  |  |
| --- | --- |
| **Kод группы** | **Наименование группы** |
| 0 | Общие положения |
| 1 | Основополагающие стандарты |
| 2 | Правила выполнения документации разработки |
| 3 | Правила выполнения документации изготовления |
| 4 | Правила выполнения документации сопровождения |
| 5 | Правила выполнения эксплуатационной документации |
| 6 | Правила обращения программной документации |
| 7 | Резервные группы |
| 8 |
| 9 | Прочие стандарты |

Обозначение стандарта ЕСПД строят по классификационному признаку:

Обозначение стандарта ЕСПД должно состоять из:

* числа 19 (присвоенных классу стандартов ЕСПД);
* одной цифры (после точки), обозначающей код классификационной группы стандартов, указанной таблице;
* двузначного числа (после тире), указывающего год регистрации стандарта.

**Перечень документов ЕСПД:**

1. ГОСТ 19.001-77 ЕСПД. Общие положения.
2. ГОСТ 19.101-77 ЕСПД. Виды программ и программных документов.
3. ГОСТ 19.102-77 ЕСПД. Стадии разработки.
4. ГОСТ 19.103-77 ЕСПД. Обозначение программ и программных документов.
5. ГОСТ 19.104-78 ЕСПД. Основные надписи.
6. ГОСТ 19.105-78 ЕСПД. Общие требования к программным документам.
7. ГОСТ 19.106-78 ЕСПД. Требования к программным документам, выполненным печатным способом.
8. ГОСТ 19.201-78 ЕСПД. Техническое задание. Требования к содержанию и оформлению.
9. ГОСТ 19.202-78 ЕСПД. Спецификация. Требования к содержанию и оформлению.
10. ГОСТ 19.301-79 ЕСПД. Порядок и методика испытаний.
11. ГОСТ 19.401-78 ЕСПД. Текст программы. Требования к содержанию и оформлению.
12. ГОСТ 19.402-78 ЕСПД. Описание программы.
13. ГОСТ 19.404-79 ЕСПД. Пояснительная записка. Требования к содержанию и оформлению.
14. ГОСТ 19.501-78 ЕСПД. Формуляр. Требования к содержанию и оформлению.
15. ГОСТ 19.502-78 ЕСПД. Описание применения. Требования к содержанию и оформлению.
16. ГОСТ 19.503-79 ЕСПД. Руководство системного программиста. Требования к содержанию и оформлению.
17. ГОСТ 19.504-79 ЕСПД. Руководство программиста.
18. ГОСТ 19.505-79 ЕСПД. Руководство оператора.
19. ГОСТ 19.506-79 ЕСПД. Описание языка.
20. ГОСТ 19.508-79 ЕСПД. Руководство по техническому обслуживанию. Требования к содержанию и оформлению.
21. ГОСТ 19.604-78 ЕСПД. Правила внесения изменений в программные документы, выполняемые печатным способом.
22. ГОСТ 19.701-90 ЕСПД. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения.
23. ГОСТ 19.781-90. Обеспечение систем обработки информации программное.

**О других межгосударственных стандартах**

Наряду с ЕСПД на межгосударственном уровне действуют еще два стандарта, также относящихся к документированию ПС и принятых не так давно, как большая часть ГОСТ ЕСПД.

ГОСТ 19781-90 Обеспечение систем обработки информации программное. Термины и определения. Разработан взамен ГОСТ 19781-83 и ГОСТ 19.004-80 и устанавливает термины и определения понятий в области программного обеспечения (ПО) систем обработки данных (СОД), применяемые во всех видах документации и литературы, входящих в сферу работ по стандартизации или использующих результаты этих работ.

ГОСТ 28388-89 Системы обработки информации. Документы на магнитных носителях данных. Порядок выполнения и обращения. Распространяется не только на программные, но и на конструкторские, технологические и другие проектные документы, выполняемые на магнитных носителях.

Государственные стандарты РФ (ГОСТ Р)

В РФ действует ряд стандартов в части документирования ПС, разработанных на основе прямого применения международных стандартов ИСО. Это — самые «свежие» по времени принятия стандарты. Некоторые из них впрямую адресованы руководителям проекта или директорам информационных служб. Вместе с тем они неоправданно мало известны в среде профессионалов. Вот их представление.

ГОСТ Р ИСО/МЭК 9294-93 Информационная технология. Руководство по управлению документированием программного обеспечения. Стандарт полностью соответствует международному стандарту ИСО/МЭК ТО 9294:1990 и устанавливает рекомендации по эффективному управлению документированием ПС для руководителей, отвечающих за их создание. Целью стандарта является оказание помощи в определении стратегии документирования ПС; выборе стандартов по документированию; выборе процедур документирования; определении необходимых ресурсов; составлении планов документирования.

ГОСТ Р ИСО/МЭК 9126-93 Информационная технология. Оценка программной продукции. Характеристики качества и руководства по их применению. Стандарт полностью соответствует международному стандарту ИСО/МЭК 9126:1991. В его контексте под характеристикой качества понимается «набор свойств (атрибутов) программной продукции, по которым ее качество описывается и оценивается». Стандарт определяет шесть комплексных характеристик, которые с минимальным дублированием описывают качество ПС (ПО, программной продукции): функциональные возможности; надежность; практичность; эффективность; сопровождаемость; мобильность. Эти характеристики образуют основу для дальнейшего уточнения и описания качества ПС

ГОСТ Р ИСО 9127-94 Системы обработки информации. Документация пользователя и информация на упаковке для потребительских программных пакетов. Стандарт полностью соответствует международному стандарту ИСО 9127:1989. В контексте настоящего стандарта под потребительским программным пакетом (ПП) понимается «программная продукция, спроектированная и продаваемая для выполнения определенных функций; программа и соответствующая ей документация, упакованные для продажи как единое целое». Под документацией пользователя понимается документация, которая обеспечивает конечного пользователя информацией по установке и эксплуатации ПП. Под информацией на упаковке понимают информацию, воспроизводимую на внешней упаковке ПП. Ее целью является предоставление потенциальным покупателям первичных сведений о ПП.

ГОСТ Р ИСО/МЭК 8631-94 Информационная технология. Программные конструктивы и условные обозначения для их представления. Описывает представление процедурных алгоритмов.

ГОСТ Р ИСО/МЭК 12119:1994. Информационная технология. Пакеты программных средств. Требования к качеству и испытания. В этом стандарте установлены требования к качеству пакетов программ и инструкции по их испытаниям на соответствие заданным требованиям. Понятие «пакет программных средств» фактически отождествляется с более общим понятием «программный продукт», рассматриваемым как совокупность программ, процедур и правил, поставляемых нескольким пользователям для общего применения или функционирования. Каждый пакет программ должен иметь описание продукта и пользовательскую документацию.

**Вопросы для самопроверки**

1. Как можно охарактеризовать понятие «программная документация»?
2. Что представляет собой внешняя и внутренняя программная документация?
3. Дайте определение понятию «единая система программной документации».
4. В чем заключаются основные недостатки единой системы программной документации?
5. Почему несмотря на множество недостатков стандарты ЕСПД могут с пользой применяться?
6. Из чего формируется обозначение стандарта ЕСПД

# Процессы жизненного цикла программных средств ГОСТ 19.102-77

Жизненный цикл разработки ПО установлен стандартом ГОСТ 19.102-77 Стадии разработки программ и программной документации и содержит следующие стадии и этапы:

1. Техническое задание (ТЗ)
2. Эскизный проект (ЭП)
3. Технический проект (ТП)
4. Рабочий проект (РП)
5. Внедрение

В таблице показаны стадии разработки и этапы, их составляющие.

Неверно предполагать, что жизненный цикл разработки ПО согласно ГОСТ 19.102-77 есть последовательное выполнение стадий и этапов, определенных в Таблица 1. В реальном жизненном цикле трудно провести четкую и определенную границу между этапами, а сам процесс создания ПО является итеративным: после завершения некоторого этапа почти всегда есть необходимость в коррекции уже выполненных этапов и стадий с целью внесения уточнений. При разработке принципиально нового ПО иногда бывает необходимо осуществить пробную реализацию с целью получения информации, требующейся для принятия решения на некоторой стадии.

Таблица 1

| Стадии разработки | Этапы работ |
| --- | --- |
| Техническое задание | 1. Обоснование необходимости разработки программ. 2. Выполнение научно-исследовательских работ (НИР). 3. Разработка и утверждение технического задания. |
| Эскизный проект | 1. Разработка эскизного проекта. 2. Утверждение эскизного проекта. |
| Технический проект | 1. Разработка технического проекта. 2. Утверждение технического проекта. |
| Рабочий проект | 1. Разработка программы. 2. Разработка программной документации. 3. Испытание программы. |
| Внедрение | 1. Подготовка и передача программы. |

**Техническое задание.** На стадии Техническое задание выполняются следующие работы, входящие в состав соответствующих этапов.

1. **Обоснование необходимости разработки программ:**

* постановка задачи;
* сбор исходных материалов;
* выбор и обоснование критериев эффективности и качества;
* обоснование необходимости проведения НИР.

1. **Выполнение научно-исследовательских работ:**

* определение структуры входных и выходных данных;
* предварительный выбор методов решения задач;  
  обоснование целесообразности применения ранее разработанных программ;
* определение требований к техническим средствам;
* обоснование принципиальной возможности решения поставленных задач.

1. **Разработка и утверждение технического задания:**

* определение требований к программе;
* разработка технико-экономического обоснования разработки программы;
* определение стадий, этапов и сроков разработки программы и документации на нее;
* выбор языков программирования;
* определение необходимости проведения НИР на последующих стадиях;
* согласование и утверждение ТЗ.

Результатом выполнения данной стадии является техническое задание, оформленное в соответствии с ГОСТ 19.105-78 (изм. 09.1981.) Общие требования к программным документам и ГОСТ 19.106-78 Общие требования к программным документам, выполненным печатным способом на листах формата 11 и 12 (по ГОСТ 2.301-68).

**Эскизный проект.** Основные этапы и содержание работ на стадии Эскизный проект приведены в Таблица 2.

Таблица 2

|  |  |
| --- | --- |
| Этапы работ | Содержание |
| Разработка ЭП | 1. Предварительная разработка структуры входных и выходных данных. 2. Уточнение методов решения задач. 3. Разработка общего описания алгоритма решения задачи. 4. Разработка технико-экономического обоснования. |
| Утверждение ЭП | 1. Разработка пояснительной записки. 2. Согласование и утверждение эскизного проекта. |

Конкретное содержание работ стадии эскизного проекта и их объем определяет степень сложности разрабатываемого ПО. Результатом выполнения данной стадии является полное описание архитектуры ПО. Как правило, это описание делается на нескольких уровнях иерархии. На верхнем уровне детализации выделяются основные подсистемы, которым присваиваются имена, устанавливаются связи между подсистемами, их функции, получаемые путем декомпозиции предполагаемых функций ПО. Затем процедура декомпозиции выполняется для каждой подсистемы, выделяются модули, составляющие данную подсистему. В конечном итоге, получается иерархически организованная система, состоящая из уровней, каждый из которых представляет собой совокупность взаимосвязанных модулей.

Единицы, выделяемые на различных иерархических уровнях функциональной архитектуры системы, определяются по усмотрению разработчика. Стандарты ЕСПД различают программные единицы только с точки зрения их документирования.

Результаты эскизного проекта отображаются в документе Пояснительная записка к эскизному проекту, оформленному в соответствии с ГОСТ 19.105-78 и ГОСТ 19.404-79.

После утверждения пояснительной записки она становится программным документам, правила дублирования, учета, хранения которого определяется ГОСТ 19.601-78 Общие правила дублирования, обращения, учета и хранения и ГОСТ 19.602-78 Правила дублирования, учета и хранения программных документов, выполненных печатным способом. Последующие стадии и этапы разработки ПО могут выявить необходимость внесения изменений в ЭП. Эти изменения должны быть отражены в пояснительной записке в соответствии с ГОСТ 19.603-78 Общие правила внесения изменений в программные документы и ГОСТ 19.602-78 Правила внесения изменений в программные документы, выполненные печатным способом.

В качестве примера, ниже приводится фрагмент расширенного описания работ стадии эскизного проекта.

**Разработка эскизного проекта ПО**

* разработка плана совместных работ на разработку ПО;
* разработка и обоснование математической модели системы на ЭВМ и описание результатов моделирования;
* разработка и обоснование алгоритмов и временных графиков функционирования ПОпо всем режимам работы;
* разработка и обоснование ресурсов памяти для реализации алгоритмов;
* разработка перечня документов на ПО;
* разработка и обоснование структуры БД, внешних входных и выходных данных;
* разработка и обоснование алгоритмов информационного обеспечения;
* определение взаимосвязей между видами программ;
* разработка и обоснование набора тестов для проверки ПО;
* разработка и обоснование организации наращивания и развития ПО;
* оформление пояснительной записки и ведомости эскизного проекта ПО (в соответствии с ГОСТ 19.105-78, ГОСТ 19.404-79 и ГОСТ 2.106-68 ЕСКД. Текстовые документы);
* согласование и утверждение ЭП.

**Технический проект.**

Основные этапы и содержание работ на стадии Технический проект приведены в Таблица 3.

Содержанием работ на этой стадии является проектирование структуры ПО. Результатом - реализующий заданный и утвержденный в техническом задании комплекс программ как иерархическая структура программных модулей, заданных своими функциональными спецификациями. Форма представления результата - Пояснительная записка к техническому проекту согласно ГОСТ 19.105-78, ГОСТ 19.404-79.

Разработка структуры ПО заключается в выделении всех программных компонентов по функциональным признакам, определение функциональных спецификаций модулей и уточнение внешних функциональных спецификаций, структуры входных и выходных данных, определении операционной среды, языковых средств и конфигурации аппаратных средств.

Спецификации модулей являются внешними характеристиками и содержат все сведения, необходимые вызывающим модулям. На последующих стадиях разработки спецификации оформляются в виде комментариев в начале текста исходной программы модуля. На данной стадии спецификации оформляются в виде комментария на принятом в организации, занимающейся разработкой ПО, языке спецификаций.

Таблица 3

|  |  |
| --- | --- |
|  | Содержание |
| Разработка ТП | 1. Уточнение структуры входных и выходных данных. 2. Разработка алгоритмов решения задач. 3. Определение формы представления входных и выходных данных. 4. Определение синтаксиса и семантики языка. 5. Разработка структуры программы. 6. Окончательное определение конфигурации технических средств. |
| Утверждение ТП | 1. Разработка плана мероприятий по разработке и внедрению программ. 2. Разработка пояснительной записки. 3. Согласование и утверждение ТП. |

**Рабочий проект.** Основные этапы и содержание работ на стадии Рабочий проект приведены в Таблица 4.

Таблица 4

|  |  |
| --- | --- |
| Этапы работ | Содержание |
| Разработка ПО | Программирование и отладка программ. |
| Разработка программной документации | Разработка программных документов в соответствии с требованиями ГОСТ 19.101-77. |
| Испытание ПО | Разработка, согласование и утверждение программ и методики испытаний  Проведение предварительных государственных, межведомственных приемо-сдаточных и других видов испытаний.  Корректировка ПО и программной документации по результатам испытаний. |

Содержанием работ на этой стадии является описание ПО на выбранном проблемно-ориентированном языке (кодирование), отладка, разработка, согласование и утверждение порядка и методики испытаний, разработка программных документов, проведение тестирования, корректировка программ и программной документации по результатам тестирования, проведение приемо-сдаточных испытаний. Результат - ПО в форме программной документации, в форме документации на ПО или в форме программного изделия.

На стадии Внедрения осуществляется подготовка и передача ПО и программной документации для сопровождения и/или изготовления, оформление и утверждение акта о передаче ПО на сопровождение или изготовление, передача ПО в фонд алгоритмов и программ.

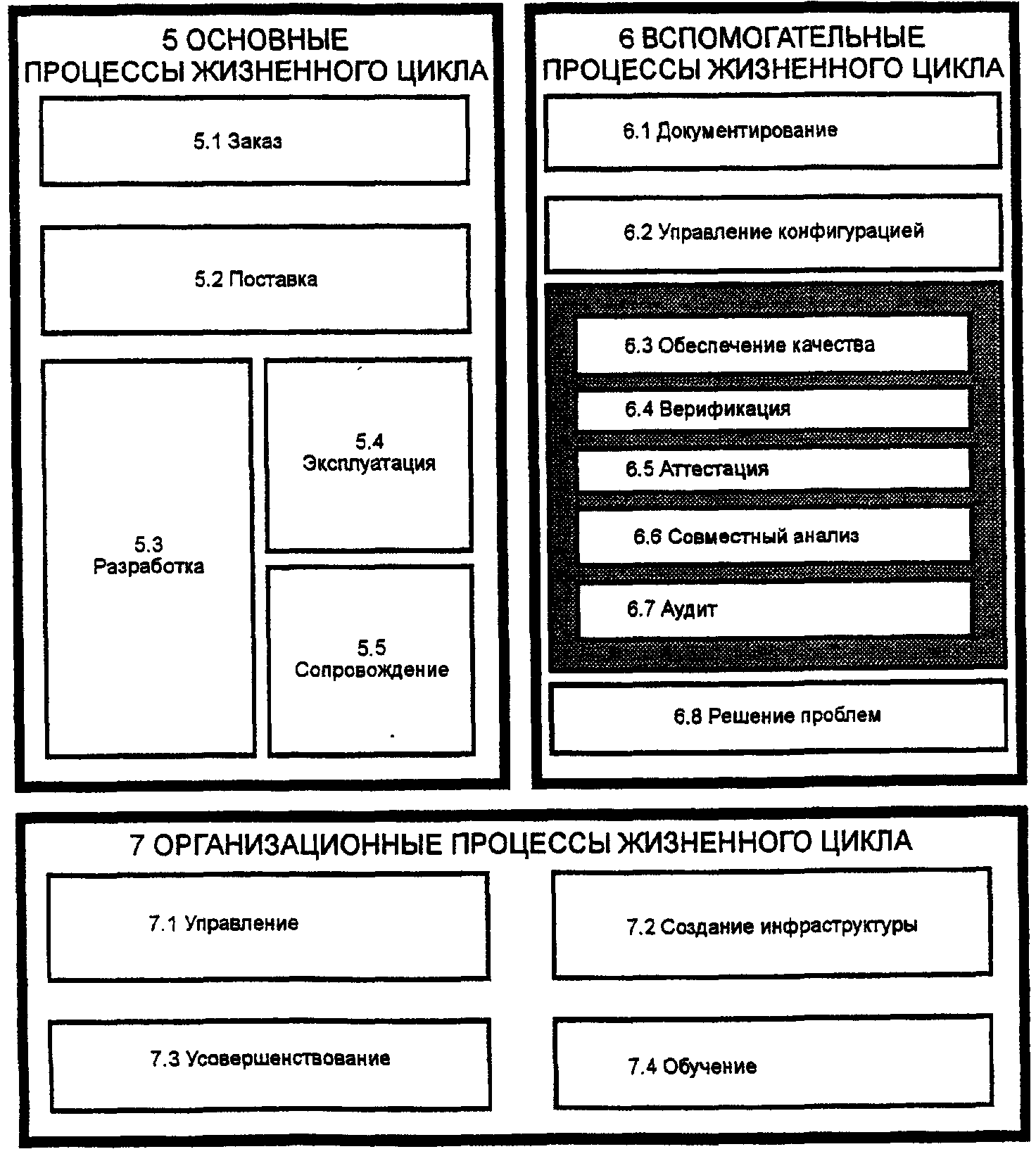
**Вопросы для самопроверки**

1. Назовите стадии жизненного цикла разработки ПО по ГОСТ 19.102-77?
2. Какие этапы работ проводятся на стадии ТЗ?
3. Какие этапы работ проводятся на стадии ЭП.
4. Какие этапы работ проводятся на стадии ТП?
5. Какие этапы работ проводятся на стадии РЗ?
6. Каковы результаты каждой стадии жизненного цикла разработки ПО?

# Процессы жизненного цикла программных средств ГОСТ Р ИСО/МЭК 12207

В стандарте ГОСТ Р ИСО/МЭК 12207 впервые реализован принцип структурной стандартизации ЖЦ ПС на основе регламентации требований к процессам, работам и задачам, входящим в полную типовую структуру ЖЦ ПС.

Процессы ЖЦ ПС выделены по принципу ответственности субъекта (заказчика, поставщика, разработчика и т. д.), реализующего конкретный процесс. В свою очередь, каждый из процессов состоит из ряда работ и решаемых при выполнении соответствующей работы задач. С точки зрения соподчиненности и важности данных процессов они разбиты на три группы: основные; вспомогательные; организационные (рис. 2.1).



Группа *основных* процессов включает в себя процессы: *приобретение; поставка; разработка; эксплуатация; сопровождение.*

Группа *вспомогательных процессов* включает в себя процессы, обеспечивающие выполнение основных процессов: *документирование; управление конфигурацией; обеспечение качества; верификация; аттестация; оценка; аудит; решение проблем.*

Группа *организационных* процессов включает в себя процессы: *управление проектами; создание инфраструктуры проекта; определение, оценка и улучшение самого ЖЦ; обучение.*

Очень важное отличие ISO: каждый процесс, действие или задача инициируется и выполняется другим процессом по мере необходимости, причем нет заранее определенных последовательностей (естественно, при сохранении логики связей по исходным сведениям задач и т. п.).

Процеcc приобретения (acquisition process) состоит из действий и задач заказчика, приобретающего программное средство (рис. 2.2).

Инициирование приобретения включает следующие задачи:

* определение заказчиком своих потребностей в приобретении, разработке или усовершенствовании системы, программных продуктов или услуг;
* анализ требований к системе;
* принятие решения относительно приобретения, разработки или усовершенствования существующего ПС;
* проверку наличия необходимой документации, гарантии, сер­тификатов, лицензий и поддержки в случае приобретения про­граммного продукта;
* подготовку и утверждение плана приобретения, включающего требования к системе, тип договора, ответственность сторон и т. д.

Заявочные предложения должны содержать: требования к системе; перечень программных продуктов; условия и соглашения;

технические ограничения (например, среда функционирования системы).

Заявочные предложения направляются выбранному постав­щику (или нескольким поставщикам в случае проведения тенде-). Поставщик — это организация, которая заключает договор с заказчиком на поставку системы, ПС или программной услуги на условиях, оговоренных в договоре.

Подготовка и корректировка договора включают следующие задачи:

* определение заказчиком процедуры выбора поставщика, вклю­чающей критерии оценки предложений возможных постав­щиков;
* ныбор конкретного поставщика на основе анализа предложений,
* подготовку и заключение договора с поставщиком;
* внесение изменений (при необходимости) в договор в процессе его выполнения.

**Процесс приобретения**

Инициирование приобретения

Подготовка заявочных предложений

Подготовка и корректировка договора

Надзор за деятельностью поставщика

Приемка и завершение работ

Определение заказчиком потребностей

Анализ требований к системе

Принятие решения о приобретении

Проверка необходимых документов

Подготовка плана приобретения

Требования к системе

Перечень программных продуктов

Условия и соглашения

Технические ограничения

Определение процедуры выбора поставщика

Выбор поставщика

Подготовка договора

Внесение изменений в договор

**Рис. 2.2.** Схема процесса приобретения программного средства

Надзор за деятельностью поставщика осуществляется в соот­ветствии с действиями, предусмотренными в процессах совмест­ной оценки и аудита.

В процессе приемки подготавливаются и выполняются необ­ходимые тесты. Завершение работ по договору осуществляется в случае удовлетворения всех условий приемки.

Процесс поставки (supply process) охватывает действия и за­дачи, выполняемые поставщиком, который снабжает заказчика программным продуктом или услугой (рис. 2.3).

**Процесс поставки**

Инициирование поставки

Планирование

Подготовка на заявочные предложения

Подготовка договора

Выполнение и контроль

Проверка и оценка

Поставка и завершение работ

Рис. 2.3. Схема процесса поставки

Инициирование поставки заключается в рассмотрении поставщи­ком заявочных предложений и принятии решения о согласии с выставленными требованиями и условиями или предложение своих. Планирование включает следующие задачи: принятие решения поставщиком относительно выполнения работ своими силами или с привлечением субподрядчика; разработку поставщиком плана управления проектом, содер­жащего организационную структуру проекта, разграничение ответственности, технические требования к среде разработки и ресурсам, управление субподрядчиками и др. Процесс разработки (development process) предусматривает Действия и задачи, выполняемые разработчиком, и охватывает работы по созданию ПС и его компонентов в соответствии с заданными требованиями, включая оформление проектной и экс­плуатационной документации; подготовку материалов, необхо­димых для проверки работоспособности и соответствующего качества программных продуктов, материалов, необходимых для организации обучения персонала, и т. д. (рис. 2.4).

**Процесс разработки**

Подготовительная работа

Кодирование и тестирование ПС

Интеграция ПС

Квалификационное тестирование ПС

Анализ требований к системе

Проектирование архитектуры системы

Анализ требований к ПС

Проектирование архитектуры ПС

Детальное проектирование ПС

Интеграция системы

Установка ПС

Квалификационное тестирование ПС

Приемка ПС

Рис. 2.4. Схема процесса разработки

Подготовительная работа начинается с выбора модели ЖЦ ПС, соответствующей масштабу, значимости и сложности про­екта. Действия и задачи процесса разработки должны соответ­ствовать выбранной модели. Разработчик должен выбрать, адап­тировать к условиям проекта и использовать согласованные с заказчиком стандарты, методы и средства разработки, а также составить план выполнения работ.

Анализ требований к системе подразумевает определение ее фун­кциональных возможностей, пользовательских требований, требо­ваний к надежности и безопасности, требований к внешним интер­фейсам и т. д. Требования к системе оцениваются исходя из крите­риев реализуемости и возможности проверки при тестировании.

Проектирование архитектуры системы на высоком уровне заключается в определении компонентов ее оборудования, ПС и операций, выполняемых эксплуатирующим систему персоналом. Архитектура системы должна соответствовать требованиям, предъявляемым к системе, а также принятым проектным стан­дартам и методам.

Анализ требований к ПС предполагает определение следую­щих характеристик для каждого компонента ПС:

* функциональных возможностей, включая характеристики про­изводительности и среды функционирования компонента;
* внешних интерфейсов;
* спецификаций надежности и безопасности;
* эргономических требований;
* требований к используемым данным;
* требований к установке и приемке;
* требований к пользовательской документации;
* требований к эксплуатации и сопровождению.

Требования к ПС оцениваются исходя из критериев соответствия имя требованиям к системе, реализуемости и возможности проверки при тестировании.

Проектирование архитектуры ПС включает следующие задачи (для каждого компонента ПС):

* трансформацию требований к ПС в архитектуру, определяю­щую на высоком уровне структуру ПС и состав его компо­нентов;
* разработку и документирование программных интерфейсов ПС
* и баз данных;
* разработку предварительной версии пользовательской документации;
* разработку и документирование предварительных требований к тестам и плана интеграции ПС.
* Архитектура компонентов ПС должна соответствовать требованиям, предъявляемым к ним, а также принятым проектным стандартам и методам.
* Детальное проектирование ПС включает следующие задачи:
* писание компонентов ПС и интерфейсов между ними на более низком уровне, достаточном для их последующего самостоя­тельного кодирования и тестирования;
* разработку и документирование детального проекта базы данных;
* обновление (при необходимости) пользовательской документации;
* разработку и документирование требований к тестам и плана тестирования компонентов ПС;
* обновление плана интеграции ПС.

Кодирование и тестирование ПС охватывают следующие за­дачи:

* разработку (кодирование) и документирование каждого ком­понента ПС и базы данных, а также совокупности тестовых процедур и данных для их тестирования;
* тестирование каждого компонента ПС и базы данных на соот­ветствие предъявляемым к ним требованиям. Результаты тес­тирования компонентов должны быть документированы;
* обновление (при необходимости) пользовательской докумен­тации;
* обновление плана интеграции ПС.

Интеграция ПС предусматривает сборку разработанных ком­понентов ПС в соответствии с планом интеграции и тестирова­ние агрегированных компонентов. Для каждого из агрегирован­ных компонентов разрабатываются наборы тестов и тестовые процедуры, предназначенные для проверки каждого из квалифи­кационных требований при последующем квалификационном тестировании. Квалификационное требование — это набор крите­риев или условий, которые необходимо выполнить, чтобы ква­лифицировать программный продукт как соответствующий сво­им спецификациям и готовый к использованию в условиях эксп­луатации.

Квалификационное тестирование ПС проводится разработчи­ком в присутствии заказчика (по возможности) для демонстра­ции того, что ПС удовлетворяет своим спецификациям и готово к использованию в условиях эксплуатации. Квалификационное тестирование выполняется для каждого компонента ПС по всем разделам требований при широком варьировании тестов. При этом также проверяются полнота технической и пользовательс­кой документации и ее адекватность самим компонентам ПС.

Интеграция системы заключается в сборке всех ее компонен­тов, включая ПС и оборудование. После интеграции система, в свою очередь, подвергается квалификационному тестированию на соответствие совокупности требований к ней. При этом также производятся оформление и проверка полного комплекта доку­ментации на систему.

Установка ПС осуществляется разработчиком в соответствии с планом в той среде и на том оборудовании, которые предусмотрены договором. В процессе установки проверяется работос­пособность ПС и баз данных. Если устанавливаемое ПС заменяет существующую систему, разработчик должен обеспечить их параллельное функционирование в соответствии с договором.

Приемка ПС предусматривает оценку результатов квалификационного тестирования ПС и системы и документирование

результатов оценки, которые проводятся заказчиком с помощью разработчика. Разработчик выполняет окончательную передачу ПС заказчику в соответствии с договором, обеспечивая при этом необходимое обучение и поддержку.

Процесс эксплуатации (operation process) охватывает действия и задачи оператора — организации, эксплуатирующей систему (рис. 2.5).

**Процесс эксплуатации**

Поддержка пользователей

Эксплуатация системы

Эксплуатационное тестирование

Подготовительная работа

Рис. 2.5. Схема процесса эксплуатации

Подготовительная работа включает проведение оператором следующих задач:

* планирование действий и работ, выполняемых в процессе эксплуатации, и установку эксплуатационных стандартов;
* определение процедур локализации и разрешения проблем, возникающих в процессе эксплуатации.

Эксплуатационное тестирование осуществляется для каждой очередной редакции программного продукта, после чего она передается в эксплуатацию.

Эксплуатация системы выполняется в предназначенной для этого среде в соответствии с пользовательской документацией.

Поддержка пользователей заключается в оказании помощи и консультации при обнаружении ошибок в процессе эксплуата­ции ПС.

Процесс сопровождения (maintenance process) предусматривает (действия и задачи, выполняемые сопровождающей организа­цией (службой сопровождения). Данный процесс активизируется при изменениях (модификациях) программного продукта и соответствующей документации, вызванных возникшими проблема ми или потребностями в модернизации либо адаптации ПС. соответствии со стандартом IEEE-90 под сопровождением пони мается внесение изменений в ПС в целях исправления ошибок повышения производительности или адаптации к изменившимся условиям работы или требованиям.

Изменения, вносимые в существующее ПС, не должны нарушать его целостность. Процесс сопровождения включает перенос ПС в другую среду (миграцию) и заканчивается снятием ПС с эксплуатации (рис. 2.6).

**Процесс сопровождения**

Подготовительная работа

Анализ проблем и запросов на модификацию ПО

Модификация ПО

Проверка и приемка

Перенос ПО в другую среду

Снятие ПО с эксплуатации

Рис. 2.6. Схема процесса сопровождения

Подготовительная работа службы сопровождения включает следующие задачи:

* планирование действий и работ, выполняемых в процессе сопровождения;
* определение процедур локализации и разрешения проблем, возникающих в процессе сопровождения.
* Анализ проблем и запросов на модификацию ПС, выполняемый службой сопровождения, включает следующие задачи:
* анализ сообщения о возникшей проблеме или запроса на модификацию ПС относительно его влияния на организацию, существующую систему и интерфейсы с другими системами. При этом определяются следующие характеристики возможной модификации: тип (корректирующая, улучшающая, профилактическая или адаптирующая к новой среде); масштаб (размеры модификации, стоимость и время ее реализации); критичность (воздействие на производительность, надежность или безопасность);
* оценку целесообразности проведения модификации и возможных вариантов ее проведения;
* утверждение выбранного варианта модификации.

Модификация ПС предусматривает определение компонентов ПС их версий и документации, подлежащих модификации, и внесение необходимых изменений в соответствии с правилами процесса наработки. Подготовленные изменения тестируются и проверяются по критериям, определенным в документации. При подтверждении корректности изменений в программах проводится корректировка документации.

Проверка и приемка заключаются в проверке целостности модифицированной системы и утверждении внесенных изменений. При переносе ПС в другую среду используются имеющиеся или разрабатываются новые средства переноса, затем выполняется конвертирование программ и данных в новую среду. С целью облегчить переход предусматривается параллельная эксплуата­ция ПС в старой и новой среде в течение некоторого периода, когда проводится необходимое обучение пользователей работе в новой среде.

Снятие ПС с эксплуатации осуществляется по решению заказчика при участии эксплуатирующей организации, службы сопровождения и пользователей. При этом программные продукты соответствующая документация подлежат архивированию в соответствии с договором. Аналогично переносу ПС в другую среду с целью облегчить переход к новой системе предусматривается параллельная эксплуатация старого и нового ПС в течение некоторого периода, когда выполняется необходимое обучение пользователей работе с новой системой.

**Вспомогательные процессы жизненного цикла программного средства**

Процесс документирования (documentation process) предусматривает формализованное описание информации, созданной в течении ЖЦ ПС. Данный процесс состоит из набора действий, с помощью которых планируют, проектируют, разрабатывают, выпускают, редактируют, распространяют и сопровождают документы, необходимые для всех заинтересованных лиц, таких, как руководители, технические специалисты и пользователи системы (рис. 2.7).

**Процесс документирования**

Подготовительная работа

Проектирование и разработка

Выпуск документации

Сопровождение

Рис. 2.7. Схема процесса документирования

Процесс управления конфигурацией (configuration management process) предполагает применение административных и технических процедур на всем протяжении ЖЦ ПС для определения состояния компонентов ПС в системе, управления модификациями ПС, описа­ния и подготовки отчетов о состоянии компонентов ПС и запросов на модификацию, обеспечения полноты, совместимости и коррект­ности компонентов ПС, управления хранением и поставкой ПС. Со­гласно стандарту IEEE-90 под конфигурацией ПС понимается сово­купность его функциональных и физических характеристик, уста­новленных в технической документации и реализованных в ПС.

Управление конфигурацией позволяет организовать, система­тически учитывать и контролировать внесение изменений в ПС на всех стадиях ЖЦ (рис. 2.8).

**Процесс управления конфигурацией**

Подготовительная работа

Идентификация конфигурации

Контроль конфигурации

Учет состояния конфигурации

Оценка конфигурации

Управление выпуском и поставками

Рис. 2.8. Схема процесса управления конфигурацией

Подготовительная работа заключается в планировании уп­равления конфигурацией.

Идентификация конфигурации устанавливает правила, с по­мощью которых можно однозначно идентифицировать и разли­чать компоненты ПС и их версии. Кроме того, каждому компоненту и его версиям соответствует однозначно обозначаемый комплект документации. В результате создается база для одно­значного выбора и манипулирования версиями компонентов ПС, использующая ограниченную и упорядоченную систему симво­лов, идентифицирующих различные версии ПС.

Контроль конфигурации предназначен для систематической оцен­ки предполагаемых модификаций ПС и координированной их ре­ализации с учетом эффективности каждой модификации и затрат на выполнение. Он обеспечивает контроль состояния и развития компонентов ПС и их версий, а также адекватность реально изме­няющихся компонентов их комплектной документации.

Учет состояния конфигурации представляет собой регистра­цию состояния компонентов ПС, подготовку отчетов обо всех реализованных и отвергнутых модификациях версий компонен­тов ПС. Совокупность отчетов обеспечивает однозначное отра­жение текущего состояния системы и ее компонентов, а также ведение истории модификаций.

Оценка конфигурации заключается в оценке функциональной полноты компонентов ПС, а также соответствия их физического состояния текущему техническому описанию.

Управление выпуском и поставка охватывают изготовление эта­лонных копий программ и документации, их хранение и поставку пользователям в соответствии с порядком, принятым в организации.

Процесс обеспечения качества (quality assurance process) обес­печивает соответствующие гарантии того, что ПС и процессы его ЖЦ соответствуют заданным требованиям и утвержденным планам. Под качеством ПС понимается совокупность свойств, которые характеризуют способность ПС удовлетворять задан­ным требованиям (рис. 2.9).

**Процесс обеспечения качества**

Подготовительная работа

Обеспечение качества продукта

Обеспечение качества процесса

Обеспечение прочих показателей качества системы

Рис. 2.9. Схема процесса обеспечения качества

Для получения достоверных оценок создаваемого ПС процесс обеспечения его качества должен происходить независимо от субъектов, непосредственно связанных с разработкой ПС. При этом могут использоваться результаты других вспомогательных процессов, таких, как верификация, аттестация, совместная оцен­ка, аудит и разрешение проблем.

Подготовительная работа заключается в координации с дру­гими вспомогательными процессами и планировании самого про­цесса обеспечения качества с учетом используемых стандартов, методов, процедур и средств.

Обеспечение качества продукта подразумевает гарантирова­ние полного соответствия программных продуктов и документа­ции на них требованиям заказчика, предусмотренным в договоре.

Обеспечение качества процесса предполагает гарантирование соответствия процессов ЖЦ ПС, методов разработки, среды раз­работки и квалификации персонала условиям договора, установ­ленным стандартам и процедурам.

Обеспечение прочих показателей качества системы осуществ­ляется в соответствии с условиями договора и стандартом каче­ства ISO 9001.

Процесс верификации (verification process) состоит в определе­нии того, что программные продукты, являющиеся результата­ми некоторого действия, полностью удовлетворяют требовани­ям или условиям, обусловленным предшествующими действиями (верификация в «узком» смысле означает формальное доказатель­ство правильности ПС). Для повышения эффективности верифи­кация должна как можно раньше интегрироваться с использую­щими ее процессами (такими, как поставка, разработка, эксплуа­тация или сопровождение). Данный процесс может включать анализ, оценку и тестирование (рис. 2.10).

**Процесс верификации**

Верификация

Подготовительная работа

Рис. 2.10. Схема процесса верификации

Верификация может проводиться с различными степенями независимости. Степень независимости может варьироваться от выполнения верификации самим исполнителем или другим спе­циалистом данной организации до ее выполнения специалистом (другой организации с различными вариациями). Если процесс ве­рификации осуществляется организацией, не зависящей от по­ставщика, разработчика, оператора или службы сопровождения, он называется процессом независимой верификации.

В процессе верификации проверяются следующие условия:

• непротиворечивость требований к системе и степень учета по­требностей пользователей;

• возможности поставщика выполнить заданные требования;

• соответствие выбранных процессов ЖЦ ПС условиям договора;

• адекватность стандартов, процедур и среды разработки про­цессам ЖЦ ПС;

• соответствие проектных спецификаций ПС заданным требова­ниям;

• корректность описания в проектных спецификациях входных и выходных данных, последовательности событий, интерфей­сов, логики и т.д.;

• соответствие кода проектным спецификациям и требованиям;

• тестируемость и корректность кода, его соответствие приня­тым стандартам кодирования;

• корректность интеграции компонентов ПС в систему;

• адекватность, полнота и непротиворечивость документации.

Процесс аттестации (validation process) предусматривает оп­ределение полноты соответствия заданных требований и создан­ной системы или программного продукта их конкретному функ­циональному назначению. Под аттестацией обычно понимают­ся подтверждение и оценка достоверности проведенного тестирования ПС. Аттестация должна гарантировать полное со­ответствие ПС спецификациям, требованиям и документации, а также возможность его безопасного и надежного применения пользователем. Аттестацию рекомендуется выполнять путем тес­тирования во всех возможных ситуациях и использовать при этом независимых специалистов. Аттестация может проводиться на начальных стадиях ЖЦ ПС или как часть работы по приемке ПС (рис. 2.11).

**Процесс аттестации**

Аттестация

Подготовительная работа

**Рис. 2.11.** Схема процесса аттестации

Аттестация, так же как и верификация, может осуществлять­ся с различными степенями независимости. Если процесс аттес­тации выполняется организацией, не зависящей от поставщика, разработчика, оператора или службы сопровождения, то он на­зывается процессом независимой аттестации.

Процесс совместной оценки (joint review process) предназначен для оценки состояния работ по проекту и ПС, создаваемому при вы­полнении данных работ (действий). Он сосредоточен в основном на контроле планирования и управления ресурсами, персоналом, ап­паратурой и инструментальными средствами проекта (рис. 2.12).

**Процесс совместной оценки**

Оценка управления проектом

Подготовительная работа

Техническая оценка

**Рис. 2.12.** Схема процесса оценки

Оценка применяется как на уровне управления проектом, так и на уровне технической реализации проекта и проводится в те­чение всего срока действия договора. Данный процесс может выполняться двумя любыми сторонами, участвующими в дого­воре, при этом одна сторона проверяет другую.

Процесс аудита (audit process) представляет собой определе­ние соответствия требованиям, планам и условиям договора.

Аудит может выполняться двумя любыми сторонами, участвую­щими в договоре, когда одна сторона проверяет другую.

Аудит — это ревизия (проверка), проводимая компетентным органом (лицом) в целях обеспечения независимой оценки степе­ни соответствия ПС или процессов установленным требованиям. Аудит служит для установления соответствия реальных работ и отчетов требованиям, планам и контракту. Аудиторы (ревизо­ры) не должны иметь прямой зависимости от разработчиков ПС. Они определяют состояние работ, использование ресурсов, со­ответствие документации спецификациям и стандартам, коррек­тность тестирования (рис. 2.13).

**Процесс аудита**

Аудит

Подготовительная работа

**Рис. 2.13.** Схема процесса аудита

Процесс разрешения проблем (problem resolution process) пре­дусматривает анализ и решение проблем (включая обнаружен­ные несоответствия) независимо от их происхождения или ис­точника, которые обнаружены в ходе разработки, эксплуатации, сопровождения или других процессов. Каждая обнаруженная проблема должна быть идентифицирована, описана, проанали­зирована и разрешена (рис. 2.14).

**Процесс разрешения проблем**

Разрешение проблем

Подготовительная работа

**Рис. 2.14.** Схема процесса разрешения проблем

**Организационные процессы жизненного цикла программного средства**

Процесс управления (management process) состоит из действий и задач, которые могут выполняться любой стороной, управляющей своими процессами. Данная сторона (менеджер) отвечает за управление выпуском продукта, управление проектом и управление задачами соответствующих процессов, таких, как приобретение, поставка, разработка, эксплуатация, сопровождение и др. (рис. 2.15).

**Процесс управления**

Инициирование и определение области управления

Планирование

Выполнение и контроль

Проверка и оценка

Завершение

Рис. 2.15. Схема процесса управления

При инициировании менеджер должен убедиться, что необходимые для управления ресурсы (персонал, оборудование и технология) имеются в его распоряжении в достаточном количестве.

Планирование подразумевает выполнение как минимум следующих задач:

* составление графиков выполнения работ;
* оценку затрат;
* выделение требуемых ресурсов;
* распределение ответственности;
* оценку рисков, связанных с конкретными задачами;
* создание инфраструктуры управления.

Процесс создания инфраструктуры (infrastructure process) охватывает выбор и поддержку (сопровождение) технологии, стандартов и инструментальных средств, выбор и установку аппаратных и программных средств, используемых для разработки, эксплуатации или сопровождения ПС. Инфраструктура должна модифицироваться и сопровождаться в соответствии с изменениями требований к соответствующим процессам. Инфраструктура, в свою очередь, является одним из объектов управления конфигурацией (рис. 2.16).

**Процесс создания инфраструктуры**

Создание инфраструктуры

Подготовительная работа

Сопровождение инфраструктуры

Рис. 2.16. Схема процесса создания инфраструктуры

Процесс усовершенствования (improvement process) предусматривает оценку, измерение, контроль и усовершенствование процессов ЖЦ ПС (рис. 2.17).

**Процесс усовершенствования**

Оценка процесса

Создание процесса

Усовершенствование процесса

Рис. 2.17. Схема процесса усовершенствования

Усовершенствование процессов ЖЦ ПС направлено на повышение производительности труда всех участвующих в них специалистов за счет совершенствования используемой технологии, методов управления, выбора инструментальных средств и обучения персонала. Усовершенствование основано на анализе достоинств и недостатков каждого процесса. Такому анализу в большой степени способствует накопление в организации исторической, технической, экономической и иной информации по реализованным проектам.

Процесс обучения (training process) охватывает первоначальное обучение и последующее постоянное повышение квалификации персонала. Приобретение, поставка, разработка, эксплуатация и сопровождение ПС в значительной степени зависят от уровня знаний и квалификации персонала. Например, разработчики ПС должны пройти необходимое обучение методам и средствам программной инженерии. Содержание процесса обучения определяется требованиями к проекту. Оно должно учитывать необходимые ресурсы и технические средства обучения. Должны быть разработаны и представлены методические материалы, необходимые для обучения пользователей в соответствии с учебным планом (рис. 2.18)

**Процесс обучения**

Разработка учебных материалов

Подготовительная работа

Реализация плана обучения

**Рис. 2.18.** Схема процесса обучения

**Вопросы для самопроверки**

1. Что понимается под профилем стандарта?
2. Объясните понятие жизненного цикла программного средства.
3. Назовите основные стандарты, характеризующие жизненный цикл программного средства.
4. Назовите и кратко охарактеризуйте процессы жизненного цикла программного средства, описанные в стандарте ГОСТ Р ИСО/МЭК 12207.

# Виды программных документов. Требования к программным документам

ГОСТ 19.101 устанавливает виды программ и программных документов для вычислительных машин, комплексов и систем независимо от их назначения и области применения.

**Виды программ**

Программу (по [ГОСТ 19781](http://www.docload.ru/Basesdoc/23/23066/index.htm)-83) допускается идентифицировать и применять самостоятельно и (или) в составе других программ.

Программы подразделяют на виды, приведенные в [табл. 1.](http://www.docload.ru/Basesdoc/6/6575/index.htm#i23654)

Таблица 1

|  |  |
| --- | --- |
| Вид программы | Определение |
| Компонент | Программа, рассматриваемая как единое целое, выполняющая законченную функцию и применяемая самостоятельно или в составе комплекса |
| Комплекс | Программа, состоящая из двух или более компонентов и (или) комплексов, выполняющих взаимосвязанные функции, и применяемая самостоятельно или в составе другого комплекса |

Документация, разработанная на программу, может использоваться для реализации и передачи программы на носителях данных, а также для изготовления программного изделия.

**Виды программных документов**

К программным относят документы, содержащие сведения, необходимые для разработки, изготовления, сопровождения и эксплуатации программ.

Виды программных документов и их содержание приведены в [табл. 2.](http://www.docload.ru/Basesdoc/6/6575/index.htm#i35007)

Таблица 2

|  |  |
| --- | --- |
| Вид программного документа | Содержание программного документа |
| Спецификация | Состав программы и документации на нее |
| Ведомость держателей подлинников | Перечень предприятий, на которых хранят подлинники программных документов |
| Текст программы | Запись программы с необходимыми комментариями |
| Описание программы | Сведения о логической структуре и функционировании программы |
| Программа и методика испытаний | Требования, подлежащие проверке при испытании программы, а также порядок и методы их контроля |
| Техническое задание | Назначение и область применения программы, технические, технико-экономические и специальные требования, предъявляемые к программе, необходимые стадии и сроки разработки, виды испытаний |
| Пояснительная записка | Схема алгоритма, общее описание алгоритма и (или) функционирования программы, а также обоснование принятых технических и технико-экономических решений |
| Эксплуатационные документы | Сведения для обеспечения функционирования и эксплуатации программы |

Виды эксплуатационных документов и их содержание приведены в табл. 3.

Таблица 3

|  |  |
| --- | --- |
| Вид эксплуатационного документа | Содержание эксплуатационного документа |
| Ведомость эксплуатационных документов | Перечень эксплуатационных документов на программу |
| Формуляр | Основные характеристики программы, комплектность и сведения об эксплуатации программы |
| Описание применения | Сведения о назначении программы, области применения, применяемых методах, классе решаемых задач, ограничениях для применения, минимальной конфигурации технических средств |
| Руководство системного программиста | Сведения для проверки, обеспечения функционирования и настройки программы на условия конкретного применения |
| Руководство программиста | Сведения для эксплуатации программы |
| Руководство оператора | Сведения для обеспечения процедуры общения оператора с вычислительной системой в процессе выполнения программы |
| Описание языка | Описание синтаксиса и семантики языка |
| Руководство по техническому обслуживанию | Сведения для применения тестовых и диагностических программ при обслуживании технических средств |

В зависимости от способа выполнения и характера применения программные документы подразделяются на подлинник, дубликат и копию ([ГОСТ 2.102-68](http://www.docload.ru/Basesdoc/4/4555/index.htm)), предназначенные для разработки, сопровождения и эксплуатации программы.

Виды программных документов, разрабатываемых на разных стадиях, и их коды приведены в [табл. 4](http://www.docload.ru/Basesdoc/6/6575/index.htm#i44303).

Таблица 4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Код вида документа | Вид документа | Стадии разработки | | | |
|  |  |  |  | Рабочий проект | |
|  |  | Эскизный проект | Технический проект | Компонент | Комплекс |
| - | Спецификация | - | - |  |  |
| 05 | Ведомость держателей подлинников | - | - | - | О |
| 12 | Текст программы | - | - |  | О |
| 13 | Описание программы | - | - | О | О |
| 20 | Ведомость эксплуатационных документов | - | - | О | О |
| 30 | Формуляр | - | - | О | О |
| 31 | Описание применения | - | - | О | О |
| 32 | Руководство системного программиста | - | - | О | О |
| 33 | Руководство программиста | - | - | О | О |
| 34 | Руководство оператора | - | - | О | О |
| 35 | Описание языка | - | - | О | О |
| 46 | Руководство по техническому обслуживанию | - | - | О | О |
| 51 | Программа и методика испытаний | - | - | О | О |
| 81 | Пояснительная записка | О | О | - | - |
| 90-99 | Прочие документы | О | О | О | О |

Условные обозначения:

 - документ обязательный;

 - документ обязательный для компонентов, имеющих самостоятельное применение;

О - необходимость составления документа определяется на этапе разработки и утверждения технического задания;

- - документ не составляют.

Допускается объединять отдельные виды эксплуатационных документов (за исключением ведомости эксплуатационных документов и формуляра). Необходимость объединения этих документов указывается в техническом задании. Объединенному документу присваивают наименование и обозначение одного из объединяемых документов.

В объединенных документах должны быть приведены сведения, которые необходимо включать в каждый объединяемый документ.

На этапе разработки и утверждения технического задания определяют необходимость составления технических условий, содержащих требования к изготовлению, контролю и приемки программы.

Технические условия разрабатывают на стадии «рабочий проект».

Необходимость составления технического задания на компоненты, не предназначенные для самостоятельного применения, и комплексы, входящие в другие комплексы, определяются по согласованию с заказчиком.

**Требования к программным документам, выполненным печатным способом (ГОСТ 19.106-78)**

Этот стандарт устанавливает правила выполнения программных документов для вычислительных машин, комплексов и систем независимо от их назначения и области применения и предусмотренных стандартами ЕСПД.

**Общие требования.** Вписывать в программные документы, выполненные машинописным, машинным и рукописным способами, отдельные слова, формулы, условные знаки (от руки чертежным шрифтом), буквы латинского и греческого алфавитов, а так же выполнять схемы и рисунки необходимо черными чернилами или тушью.

Опечатки и графические неточности, обнаруженные в процессе выполнения допускается исправлять подчисткой некачественно выполненной части текста (чертежа) и нанесением на том же листе исправленного текста (графики) машинописью или черной тушью в зависимости от способа выполнения документа.

Повреждение листов документов, помарки и следы не полностью удаленного текста (графики) не допускаются.

Программные документы оформляют на листах формата А4. Кроме того:

* допустимо оформление на листах формата А3;
* при машинном способе выполнения документа допускаются отклонения размеров листов, соответствующих форматам А4 и А3, определяемые возможностями применяемых технических средств; на листах форматов А4 и А3, предусматриваемых выходными характеристиками устройств вывода данных, при изготовлении документа машинным способом;
* при изготовлении документа типографским способом возможно использование листов типографических форматов.

Расположение материалов программного документа осуществляется в следующей последовательности:

* титульная часть:
  + лист утверждения (не входит в общее количество листов документа);
  + титульный лист (первый лист документа);
  + информационная часть:
  + аннотация;
  + лист содержания;
  + основная часть:
  + текст документа (с рисунками, таблицами и т.п.);
  + перечень терминов и их определений;
  + перечень сокращений;
  + приложения;
  + предметный указатель;
  + перечень ссылочных документов;
* часть регистрации изменений:
  + лист регистрации изменений.

**Построение документа.** При необходимости допускается делить документ на части. Деление на части осуществляется на уровне не ниже раздела. Каждую часть комплектуют отдельно, при этом в конце содержания первой части следует перечислить названия остальных частей.

Допускается включение в документ частей текста программы, оформляемых в соответствии с правилами языка, на котором написан текст программы.

Аннотацию размещают на отдельной странице (страницах), снабжают заголовком "АННОТАЦИЯ", нумеруют и включают в содержание документа.

Содержание документа размещают на отдельной странице (страницах), снабжают заголовком "СОДЕРЖАНИЕ" и включают в общее количество страниц документа. В содержании документа дается перечисление наименований разделов и подразделов и номеров страниц.

Текст каждого документа, при необходимости, разбивается на пункты, а пункты - на подпункты, независимо от того, разделен документ на части, разделы и подразделы или нет.

Заголовки разделов пишут прописными буквами и размещают симметрично относительно правой и левой границ текста. Заголовки подразделов записывают с абзаца строчными буквами (кроме первой прописной). Переносы слов в заголовках не допускаются. Точку в конце заголовка не ставят. Каждый раздел рекомендуется начинать с нового листа.

Разделы, подразделы, пункты и подпункты следует нумеровать арабскими цифрами с точкой. Разделы должны иметь порядковый номер (1, 2 и т.д.)

**Текст документа.** Текст документа должен быть кратким, четким, исключающим возможность неверного толкования. Термины и определения должны быть едиными и соответствовать установленным стандартам, а при их отсутствии - общепринятым в научно-технической литературе, и приводиться в перечне терминов.

Необходимые пояснения к тексту документа могут оформляться сносками. Сноска обозначается цифрой со скобкой, вынесенной на уровень линии верхнего обреза шрифта.

Если сноска относится к отдельному слову, знак сноски помещается непосредственно возле этого слова, если же к предложению целом, то в конце предложения. Текст сноски располагают в конце страницы и отделяют от основного текста линией длиной 3 см, проведенной в левой части страницы.

**Иллюстрации.** Иллюстрации могут быть расположены в тексте документа и (или) в приложениях. Иллюстрации, если их в данном документе более одной, нумеруют арабскими цифрами в пределах всего документа.

В приложениях иллюстрации нумеруются в пределах каждого приложения в порядке, установленном для основного текста документа. Ссылки на иллюстрации дают по типу: "рис.12" или "(рис.12)". Иллюстрации могут иметь тематический заголовок и подрисуночный текст, поясняющий содержание иллюстрации.

**Формулы.** Формулы в документе, если их более одной, нумеруются арабскими цифрами, номер ставят с правой стороны страницы, в скобках на уровне формулы. В пределах всего документа или его частей, в случае деления документа на части, формулы имеют сквозную нумерацию.

Ссылки в тексте на порядковый номер формулы дают в скобках, например: "в формуле (3)". При делении документа на части номер части ставится перед порядковым номером формулы и отделяется от последней точкой, например: "в формуле (1.4)".

Значение символов, входящих в формулу, должны быть приведены непосредственно под формулой. Значение каждого символа печатают с новой строки в той последовательности, в какой они приведены в формуле. Первая строка расшифровки должна начинаться со слова "где", без двоеточия после него.

**Ссылки.** В программных документах допускаются ссылки на стандарты и другие документы. Ссылаться следует на документ в целом или на его разделы (с указанием обозначения и наименования документа, номера и наименования раздела или приложения).

Допускается указывать только обозначение документа и (или) разделов без указания их наименований. Ссылки на отдельные подразделы, пункты и иллюстрации другого документа не допускаются. Допускаются ссылки внутри документа на пункты, иллюстрации и отдельные подразделы.

**Примечания.** В примечаниях к тексту и таблицам указывают только справочные и пояснительные данные. Одно примечание не нумеруется. После слова "Примечание" ставят точку. Несколько примечаний следует нумеровать по порядку арабскими цифрами с точкой. После слова "Примечание" ставят двоеточие. Текст примечаний допускается печатать только через один интервал.

**Сокращения.** Сокращения слов в тексте и надписях под иллюстрациями не допускаются, за исключением:

* сокращений, установленных в ГОСТ 2.316-68, и общепринятых в русском языке;
* сокращений, применяемых для обозначения программ, их частей и режимов работы, в языках управления заданиями, в средствах настройки программы и т.п., обозначаемых буквами латинского алфавита.

**Приложения.** Иллюстрированный материал, таблицы или текст вспомогательного характера допускается оформлять в виде приложений. Приложения оформляют как продолжение данного документа на последующих страницах или выпускают в виде отдельного документа.

Каждое приложение должно начинаться с новой страницы с указанием в правом верхнем углу слова "Приложение" и иметь тематический заголовок. При наличии в документе более одного приложения все приложения нумеруют арабскими цифрами (без знака №), например:

*Приложение 1, Приложение 2 и т.д.*

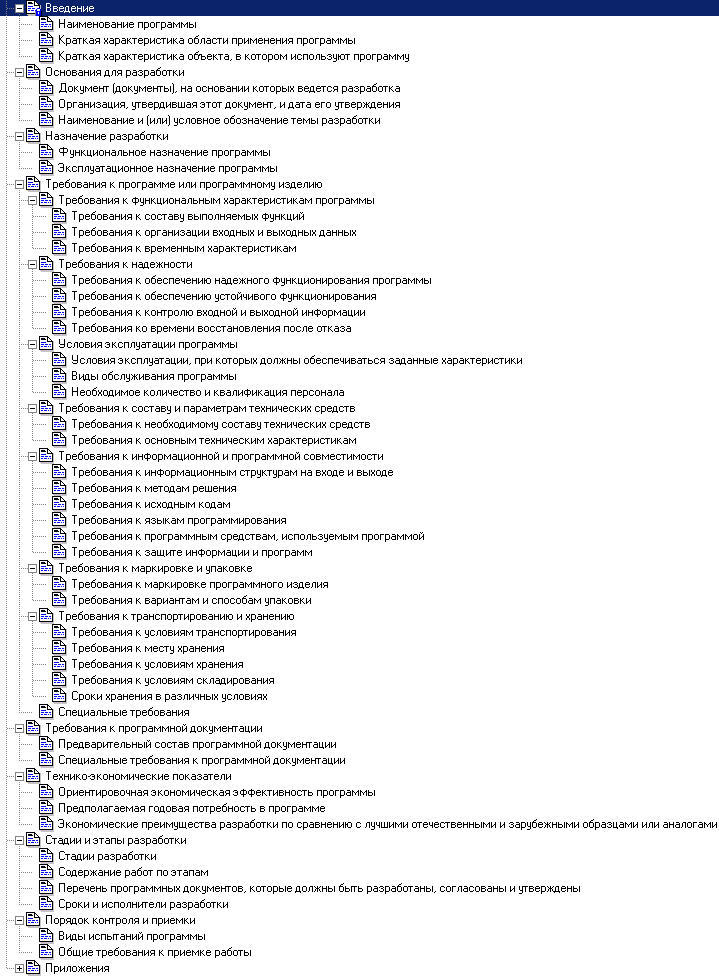
При выпуске приложения отдельным документом, на титульном листе под наименованием документа следует указывать слово "Приложение", а при наличии нескольких приложений указывают также его порядковый номер.

**Вопросы для самопроверки**

1. Дайте определение понятиям «Компонент» и «Комплекс».
2. Перечислите виды программных документов.
3. Что входит в титульную часть программного документа?
4. Как оформляется сноска?
5. Как оформляются иллюстрации?
6. Как оформляются формулы?
7. Как оформляются примечания?
8. Можно ли использовать сокращения в тексте и надписях под иллюстрациями?

# Техническое задание

Техническое задание оформляют в соответствии с [ГОСТ 19.106-78](javascript:void(18928)) на листах формата 11 и 12 по ГОСТ 2.301-68, как правило, без заполнения полей листа. Номера листов (страниц) проставляются в верхней части листа над текстом [из п. 1.1 ГОСТ 19.201-78]



**Лист утверждения и титульный лист**

Лист [утверждения](javascript:void(9746)) и титульный лист оформляют в соответствии с ГОСТ 19.104-78.

Информационную часть (аннотацию и содержание), лист регистрации изменений допускается в документ не включать [из п. 1.2 ГОСТ 19.201-78]

Указанной возможностью следует воспользоваться. Меньше слов – меньше вопросов.

**Изменения и дополнения**

Для внесения [изменений](javascript:void(9689)) или дополнений в техническое задание на последующих [стадиях разработки](javascript:void(11989)) [программы](javascript:void(16831)) или [программного изделия](javascript:void(10682)) выпускают дополнение к нему. [Согласование](javascript:void(9745)) и [утверждение](javascript:void(9746)) дополнения к техническому заданию проводят в том же порядке, который установлен для технического задания [из п. 1.3 ГОСТ 19.201-78]

Учесть все детали на начальных [стадиях разработки](javascript:void(11989)) невозможно, поэтому на практике указанный подход применяется весьма часто. В [разделе](javascript:void(22803)) «Стадии и этапы разработки» следует явно указать возможность внесения изменений и дополнений в техническое задание: «Содержимое разделов настоящего технического задания может быть изменено и дополнено по согласованию с заказчиком».

**Состав разделов технического задания**

Техническое задание должно содержать следующие разделы:

* введение;
* основания для разработки;
* назначение разработки;
* требования к [программе](javascript:void(16831)) или [программному изделию](javascript:void(10682));
* требования к [программной документации](javascript:void(10666));
* технико-экономические показатели;
* стадии и этапы разработки;
* порядок контроля и приемки;
* в техническое задание допускается включать приложения.

В зависимости от особенностей программы или программного изделия допускается уточнять содержание разделов, вводить новые разделы или объединять отдельные из них [из п. 1.4 ГОСТ 19.201-78]

Любые манипуляции с разделами - строго по [согласованию](javascript:void(9745)) с [заказчиком](javascript:void(9750)).

**Содержание разделов**

Отдельные подразделы технического задания могут подействовать на условного заказчика, как красная тряпка на быка. Заказчика, даже условного, раздражать не следует. В спорных подразделах будут рассмотрены пути поиска компромиссных решений. Ключевые позиции, в которых уступка заказчику равносильна затягиванию петли на шее исполнителя, будут также откомментированы с обоснованием жесткой позиции исполнителя.

Чтобы излишне не отягощать ход повествования, в качестве [учебно-тренировочной](javascript:void(21708)) будем использовать реальную программу с [графическим пользовательским интерфейсом](http://tdocs.su/1808), обеспечивающую возможность выполнения нескольких шаблонных функций. Пусть такой программой станет несложный [текстовый редактор](javascript:void(17049)).

**Введение**

В разделе «Введение» указывают [наименование](javascript:void(23779)), краткую характеристику области применения [программы](javascript:void(16831)) или [программного изделия](javascript:void(10682)) и объекта, в котором используют программу или программное изделие [из п. 2.1 ГОСТ 19.201-78]

Основное правило работы с [текстом](javascript:void(21016)) – детализация, дробление текста на структурные единицы, - разделы, подразделы, пункты и подпункты. [Содержание](javascript:void(25195)) документа будет иметь четкую структуру, способствующую легкому [поиску](javascript:void(22834)) требуемого материала. Текст документа станет структурированным и удобным для чтения. Создаем подразделы:

*Наименование программы*

Наименование программы – «Текстовый редактор для работы с файлами формата rtf».

*Краткая характеристика области применения*

Программа предназначена к применению в профильных подразделениях на объектах заказчика.

Содержимое отдельных пунктов не всегда очевидно. При затруднениях следует подходить формально. [Скорректировать](javascript:void(9688)) документ можно будет в ходе [согласования](javascript:void(9745)) технического задания с заказчиком.

**Основания для разработки**

В разделе «Основания для разработки» должны быть указаны:

* документ (документы), на основании которых ведется разработка;
* организация, [утвердившая](javascript:void(9746)) этот документ, и дата его утверждения;
* [наименование](javascript:void(23779)) и (или) условное обозначение темы разработки.

[из п. 2.2 ГОСТ 19.201-78]

В подразделе следует привести сведения, содержащиеся в договоре между заказчиком и исполнителем.

|  |
| --- |
|  |

**Основание для проведения разработки**

Основанием для проведения разработки является Договор (письмо и т.д.) № 666 от 32 мартобря 2004 года (входящий № такой-то от такого-то). Договор утвержден Директором ФГУП «Спецтяжмонтажстройсельхозавтоматика» Ивановым Петром Ивановичем, именуемым в дальнейшем Заказчиком, и утвержден Генеральным директором ОАО «Суперсофт» Блюмкинсом Иваном Ароновичем, именуемым в дальнейшем Исполнителем, такого-то мартобря 2004.

Удобно воспользоваться разделом «Общие сведения» [ГОСТ 34.602-89](http://tdocs.su/taxonomy/term/49), поскольку разработчик имеет полное право дополнять и удалять разделы технического задания на свое усмотрение. В то же время сведения, указанные выше, содержатся в договоре. Следует ли приводить их в техническом задании – зависит от конкретного случая.

**Наименование и условное обозначение темы разработки**

Наименование темы разработки – «Разработка текстового редактора для работы с файлами формата rtf». Условное обозначение темы разработки (шифр темы) – «РТФ-007»

**Назначение разработки**

В разделе «Назначение разработки» должно быть указано [функциональное](javascript:void(16611)) и эксплуатационное назначение [программы](javascript:void(16831)) или [программного изделия](javascript:void(10682)) [из п. 2.3 ГОСТ 19.201-78]

|  |
| --- |
|  |

*Функциональное назначение*

Функциональным назначением программы является предоставление пользователю возможности работы с текстовыми документами в формате rtf.

В подразделе должно быть указано «укрупненное» функциональное назначение программы. Детали – перечень функций и т.д. – будут приведены ниже, в соответствующих разделах.

*Эксплуатационное назначение* может трактоваться достаточно широко. Где, как, кем, с чем должна эксплуатироваться программа?

Резина одного типоразмера может успешно экслуатироваться на Жигулях и Волгах, но не на КаМАЗе. По причине отсутствия [размерной технической совместимости](javascript:void(24275)). И наоборот. Но для каждого конкретного типоразмера резины можно определить ее эксплуатационное назначение.

Применим формальный подход:

*Эксплуатационное назначение*

Программа должна эксплуатироваться в профильных подразделениях на объектах заказчика. [Пользователями программы](javascript:void(16603)) должны являться сотрудники профильных подразделений объектов заказчика.

Два крайних предложения - не в тему. При проведении переговоров с реальным заказчиком подраздел будет откорректирован.

**Требования к программе или программному изделию**

Раздел «Требования к программе или программному изделию» должен содержать следующие подразделы:

* требования к функциональным характеристикам;
* требования к [надежности](javascript:void(17179));
* [условия эксплуатации](javascript:void(10583));
* требования к составу и параметрам [технических средств](javascript:void(17000));
* требования к [информационной](javascript:void(8702)) и [программной совместимости](javascript:void(8696));
* требования к маркировке и упаковке;
* требования к [транспортированию](javascript:void(9713)) и [хранению](javascript:void(9714));
* специальные требования.

[из п. 2.4 ГОСТ 19.201-78]

Если существуют стандарты, содержащие общие (технические) требования к программе, [системе](javascript:void(9919)) или [изделию](javascript:void(9632)), к примеру, «ГОСТ 12345-67. Автоматизированные информационно-измерительные системы. Общие (технические) требования», разработка технического задания существенно упрощается. Большая часть содержимого указанного стандарта просто переписывается в техническое задание.

|  |
| --- |
|  |

**Требования к функциональным характеристикам**

В подразделе «Требования к функциональным характеристикам» должны быть указаны требования к составу выполняемых функций, организации [входных](javascript:void(17031)) и [выходных данных](javascript:void(17032)), временным характеристикам и т. п. [из п. 2.4.1 ГОСТ 19.201-78]

|  |
| --- |
|  |

**Требования к составу выполняемых функций**

Программа должна обеспечивать возможность выполнения перечисленных ниже функций:

* функции создания нового (пустого) файла;
* функции открытия (загрузки) существующего файла;
* функции редактирования открытого (далее - текущего) файла путем ввода, [замены](javascript:void(22833)), [удаления](javascript:void(22823)) содержимого файла с применением стандартных [устройств ввода](javascript:void(23329));
* функции редактирования текущего файла с применением [буфера обмена](javascript:void(22840)) [операционной системы](javascript:void(17011));
* функции [сохранения](javascript:void(22824)) файла с исходным [именем](javascript:void(23779));
* функции сохранения файла с именем, отличным от исходного;
* функции отправки содержимого текущего файла электронной почтой с помощью внешней клиентской почтовой программы;
* функции вывода оперативных справок в [строковом](javascript:void(22807)) [формате](javascript:void(22805)) (подсказок);
* функции [интерактивной](javascript:void(17037)) [справочной системы](javascript:void(25199));
* функции отображения названия программы, версии программы, копирайта и комментариев разработчика.

Клише «обеспечивать возможность выполнения» применимо к современным программным средствам, разработанным с использованием графического пользовательского интерфейса. Указанные программные средства большей частью «простаивают» (idle), ожидая [действий оператора](javascript:void(16933)). Применение клише - шаблонного построения фраз - детально расписано в статье «[Как писать техническое задание?!](http://tdocs.su/1349)».

**Требования к организации входных данных**

[Входные данные](javascript:void(21045)) программы должны быть организованы в виде отдельных файлов формата rtf, соответствующих RFC... Файлы указанного формата должны размещаться ([храниться](javascript:void(21055))) на локальных или съемных [носителях](javascript:void(23332)), [отформатированных](javascript:void(23420)) согласно требованиям операционной системы.

Любой [файл](javascript:void(18693)) иного [формата](javascript:void(21025)), но с расширением rtf, открываться не должен.

Файлы http://domain.net/file.rtf или ftp://domain.net/file.rtf открываться не должны. Если файловая система отформатирована как FAT32, файлы с локального или съемного [носителя](javascript:void(16999)), отформатированного, к примеру, в формате ext3, открываться не должны.

**Требования к организации выходных данных**

Требования те же, что и к организации выходных данных. Тот самый случай, когда следует объединить оба пункта технического задания.

**Требования к временным характеристикам**

Требования к временным характеристикам программы не предъявляются.

Следует уточнить, предъявляет ли заказчик требования к быстродействию программы, к примеру, за какое время программа должна стартовать, открывать и закрывать файлы заданного объема. Если заказчик укажет конкретные цифры, следует подстраховаться и заложить в требованиях к составу и параметрам технических средств [суперкомпьютер](javascript:void(17004)) стоимостью от $2500. Правда, такую сумму придется обосновывать. Если временные характеристики для заказчика не принципиальны, следует обязательно написать об отказе от требований к временным характеристикам, см. формулировку выше.

**Требования к надежности**

В подразделе «Требования к надежности» должны быть указаны требования к обеспечению [надежного](javascript:void(11781)) функционирования (обеспечения [устойчивого функционирования](javascript:void(17180)), контроль входной и выходной информации, [время восстановления](javascript:void(11751)) после [отказа](javascript:void(11807)) и т.п.) [из п. 2.4.2 ГОСТ 19.201-78]

[Надежность](javascript:void(9580)) – штука тонкая и очень опасная. Но перечень функций и видов их [отказов](javascript:void(11807)), согласно п. 1.3.2. [ГОСТ 24.701-86](javascript:void(14504)), обязан составить заказчик и согласовать с исполнителем.

Скорее всего, дождаться от заказчика чего-либо вразумительного не удастся. Стоит разъяснить заказчику, что надежное функционирование программы зависит не столько от исполнителя, сколько от надежности технических средств и [операционной системы](javascript:void(17011)), а также предложить заказчику ряд жестких мер для повышения [надежности](javascript:void(13314)) и [устойчивости функционирования программы](javascript:void(17180)).

**Требования к обеспечению надежного (устойчивого) функционирования программы**

Надежное (устойчивое) функционирование программы должно быть обеспечено выполнением заказчиком совокупности организационно-технических мероприятий, перечень которых приведен ниже:

* организацией бесперебойного питания технических средств;
* использованием [лицензионного](javascript:void(9609)) программного обеспечения;
* регулярным выполнением рекомендаций Министерства труда и социального развития РФ, изложенных в Постановлении от 23 июля 1998 г. «Об утверждении межотраслевых типовых норм времени на работы по сервисному обслуживанию ПЭВМ и оргтехники и сопровождению программных средств»;
* регулярным выполнением требований ГОСТ 51188-98. Защита инфоpмации. Испытания пpогpаммных сpедств на наличие [компьютеpных виpусов](javascript:void(22638)).

К списку можно добавить еще несколько десятков [нормативно-технических документов](javascript:void(21263)). В ходе первичного согласования технического задания заказчик, скорее всего, начнет проявлять склонность к компромиссу.

Возможен более гуманный подход. Под надежностью (правда, системы, по тому же ГОСТу) можно считать [безотказное](javascript:void(11743)) выполнение некой i-той функции в течение конкретного интервала времени. Предложим заказчику считать критерием надежной работы программы следующий показатель: заказчик в течение часа 100 раз открывает и закрывает файл. Если в указанном интервале времени программа не даст [сбоев](javascript:void(11842)), [требования по надежности](javascript:void(11900)) считаются выполненными.

Если заказчик, наконец, убедился, что надежность зависит не столько от исполнителя, сколько от надежности технических средств и операционной системы, и махнул рукой – в разделе обязательно следует написать такую фразу:

Требования к обеспечению надежного (устойчивого) функционирования программы не предъявляются.

**Время восстановления после отказа**

[Время восстановления после отказа](javascript:void(11751)), вызванного сбоем электропитания технических средств (иными внешними факторами), не фатальным сбоем (не крахом) операционной системы, не должно превышать стольких-то минут при условии соблюдения [условий эксплуатации](javascript:void(10583)) технических и программных средств. Время восстановления после отказа, вызванного неисправностью технических средств, фатальным сбоем (крахом) операционной системы, не должно превышать времени, требуемого на устранение [неисправностей](javascript:void(11790)) технических средств и переустановки программных средств.

Перечень аварийных ситуаций ситуаций также составляет заказчик и согласовывает с исполнителем. Фактически, это время на перезагрузку операционной системы, если отказ не фатален, не вызван крахом операционной системы или выходом из строя технических средств.

**Отказы из-за некорректных действий оператора**

Отказы программы возможны вследствие [некорректных действий оператора (пользователя)](javascript:void(20258)) при взаимодействии с операционной системой. Во избежание возникновения отказов программы по указанной выше причине следует обеспечить работу пользователя без предоставления ему административных [привилегий](javascript:void(11340)).

**Условия эксплуатации**

В подразделе «Условия эксплуатации» должны быть указаны [условия эксплуатации](javascript:void(10583)) ([температура окружающего воздуха](javascript:void(16348)), относительная влажность и т.п. для выбранных типов [носителей данных](javascript:void(16999))), при которых должны обеспечиваться заданные характеристики, а также [вид обслуживания](javascript:void(10887)), необходимое количество и квалификация персонала [из п. 2.4.3 ГОСТ 19.201-78]

Очень опасный подраздел для тех, кто делает первые шаги в разработке технического задания.

|  |
| --- |
|  |

**Климатические условия эксплуатации**

Климатические условия эксплутатации, при которых должны обеспечиваться заданные характеристики, должны удовлетворять требованиям, предъявляемым к техническим средствам в части условий их эксплуатации.

Программа будет прекрасно работать от плюс 5 до плюс 35 °C при относительной влажности 90 % и атмосферном давлении 462 мм.рт.ст., поскольку такие условия приблизительно соответствуют условиям эксплуатации современных компьютеров непромышленного исполнения. Но как только в техническом задании окажется конкретика и задание будет утверждено, заказчик получает отличный шанс заставить исполнителя провести [климатические испытания](javascript:void(24754)) в полном объеме за счет исполнителя.

Много лет тому назад автор статьи, в силу молодости и неукротимого желания отстоять свою позицию (в техническом задании, в частности), «попал на климатику», причем «попал конкретно» (так, со слов Путина, выражается просвещенная интеллигенция), на довольно крутом «железе». Автор статьи мигом усвоил, что такое «показать кузькину мать» и «где раки зимуют». Упаси Вас господь «попадать на климатику»!

Примечание от 10.02.2011 г. - По иронии судьбы специалисты «Технической документации» не так давно снова «попали на климатику», а точнее - провели разработку [программы и методики испытаний на воздействие внешних факторов](http://tdocs.su/18173), по [данным](javascript:void(24713)) и [результатам испытаний](javascript:void(24714)) подготовили [протокол испытаний](javascript:void(9799)), открыв для себя еще одно [направление деятельности](http://tdocs.su/taxonomy/term/772). Не зря говорят, что история развивается по восходящей спирали...

Автор выражает искреннюю благодарность «тогдашнему» заказчику за хороший урок.

**Требования к видам обслуживания**

Программа не требует проведения каких-либо видов [обслуживания](javascript:void(10887)).

Виды обслуживания следует позаимствовать из подраздела «Требования к обеспечению надежного (устойчивого) функционирования».

Если заказчик в ходе согласования технического задания сошлется на отсутствие [ресурсов](javascript:void(22236)) или желание проводить все виды обслуживания собственными силами, имеет смысл предложить разработку технического задания на [сопровождение программного изделия](javascript:void(11449)) за отдельные деньги отдельным договором. Откажется – следует считать программу [необслуживаемой](javascript:void(11792)).

**Требования к численности и квалификации персонала**

Минимальное количество персонала, требуемого для работы программы, должно составлять не менее 2 штатных единиц – системный администратор и пользователь программы – оператор.

Системный администратор должен иметь высшее профильное образование и [сертификаты](javascript:void(9807)) компании-производителя операционной системы. В перечень задач, выполняемых системным администратором, должны входить:

* задача поддержания [работоспособности](javascript:void(11825)) технических средств;
* задачи установки (инсталляции) и поддержания работоспособности системных программных средств – операционной системы;
* задача установки (инсталляции) программы.

Пользователь программы (оператор) должен обладать практическими навыками работы с [графическим пользовательским интерфейсом](http://tdocs.su/1808) операционной системы.

Персонал должен быть аттестован на II квалификационную группу по электробезопасности (для работы с конторским оборудованием).

При отсутствии самой ключевой фразы в утвержденном техническом задании заказчик вправе затребовать от исполнителя разработку руководства по эксплуатации графического пользовательского интерфейса операционной системы, мотивируя тем, что оператор «не справляется» с программой.

Персонал, не имеющий II квалификационной группы по электробезопасности, не имеет права даже близко подходить к [ПЭВМ](javascript:void(17008)) и конторскому оборудованию.

**Требования к составу и параметрам технических средств**

В подразделе «Требования к составу и параметрам технических средств» указывают необходимый состав [технических средств](javascript:void(17000)) с указанием их основных технических характеристик [из п. 2.4.4 ГОСТ 19.201-78]

Следует подбирать [технику](javascript:void(20787)) не хуже той, на которой будет производиться разработка. Логично затребовать, чтобы технику предоставил заказчик не позднее указанного срока. Речь идет, разумеется, о компьютере.

В состав технических средств должен входить IBM-совместимый персональный компьютер (ПЭВМ), включающий в себя:

* процессор Pentium-1000 с тактовой частотой, ГГц - 10, не менее;
* материнскую плату с FSB, ГГц - 5, не менее;
* оперативную память объемом, Тб - 10, не менее;
* и так далее…

**Требования к информационной и программной совместимости**

В подразделе «Требования к информационной и программной совместимости» должны быть указаны требования к информационным структурам на входе и выходе и методам решения, [исходным кодам](javascript:void(16845)), [языкам программирования](javascript:void(23473)) и [программным средствам](javascript:void(17150)), используемым программой.

При необходимости должна обеспечиваться [защита информации](javascript:void(11498)) и [программ](javascript:void(16831)) [из п. 2.4.5 ГОСТ 19.201-78]

|  |
| --- |
|  |

**Требования к информационным структурам и методам решения**

Иформационная структура файла должна включать в себя текст, содержащий [разметку](javascript:void(21034)), предусмотренную спецификацией формата rtf.

или

Требования к информационным структурам (файлов) на входе и выходе, а также к методам решения не предъявляются.

**Требования к исходным кодам и языкам программирования**

Исходные коды программы должны быть реализованы на [языке](javascript:void(23473)) C++. В качестве интегрированной [среды разработки](javascript:void(16850)) программы должна быть использована среда Borland C++ Buider.

**Требования к программным средствам, используемым программой**

[Системные программные средства](javascript:void(16833)), используемые программой, должны быть представлены лицензионной локализованной версией операционной системы такой-то. Допускается применение пакета обновления такого-то.

**Требования к защите информации и программ**

Требования к [защите информации](javascript:void(20372)) и программ не предъявляются.

Подобных требований следует избегать, если нет особого желания разработать что-то вроде [концепции обеспечения информационной безопасности](http://tdocs.su/13028) согласно [ГТК РФ. Руководящий документ. Защита от несанкционированного доступа к информации. Термины и определения](http://tdocs.su/taxonomy/term/52) Обеспечить некоторый уровень [защиты информации](javascript:void(11498)) и программ возможно, обеспечить безопасность невозможно. Заказчик, скорее всего, это осознает и проявлять настойчивость не станет.

**Требования к маркировке и упаковке**

В подразделе «Требования к маркировке и упаковке» в общем случае указывают требования к маркировке [программного изделия](javascript:void(10682)), варианты и способы упаковки [из п. 2.4.6 ГОСТ 19.201-78]

Программа поставляется в виде программного изделия - на дистрибутивном (внешнем [оптическом](javascript:void(23454))) носителе (компакт-диске). Речь идет, разумеется, о маркировке и упаковке дистрибутивного [носителя данных](javascript:void(16999)).

|  |
| --- |
|  |

**Требование к маркировке**

Программное изделие должно иметь маркировку с обозначением [товарного знака](javascript:void(9608)) компании-разработчика, типа (наименования), номера версии, порядкового номера, даты изготовления и номера [сертификата соответствия](javascript:void(9807)) Госстандарта России (если таковой имеется). Маркировка должна быть нанесена на программное изделие в виде наклейки, выполненной полиграфическим способом с учетом требований ГОСТ 9181-74.

Качество маркировки проверяется самыми изощренными способами – сначала пытаются смыть маркировку водой, затем бензином и прочими органическими растворителями. Пусть полиграфическое предприятие несет ответственность за некачественную маркировку. Задача исполнителя - прикрыться сертификатом соответствия (затребовать сертификат у полиграфистов).

**Требования к упаковке**

Упаковка программного изделия должна осуществляться в упаковочную тару [предприятия-изготовителя](javascript:void(9769)) ([поставщика](javascript:void(9755))).

Именно предприятия-изготовителя (поставщика). Исполнитель не может и не должен нести ответственность большую, чем предприятие-изготовитель (поставщик) тары.

|  |
| --- |
|  |

**Условия упаковывания**

Упаковка программного изделия должна проводиться в закрытых вентилируемых помещениях при температуре от плюс 15 до плюс 40 °С и относительной влажности не более 80 % при отсутствии агрессивных примесей в окружающей среде.

Заказчик получит программное изделие надлежащего внешнего вида. В случае возврата программного изделия (по [рекламации](javascript:void(9605))) в ненадлежащем виде (наличие царапин, трещин и прочих [дефектов](javascript:void(9586))) исполнитель сможет предъявить претензии в части нарушения заказчиком условий упаковывания и не принять программное изделие.

**Порядок упаковки**

Подготовленные к упаковке программные изделия укладывают в тару, представляющую собой коробки из картона гофрированного (ГОСТ 7376-89 или ГОСТ 7933- 89) согласно чертежам предприятия-изготовителя тары.

Программное изделие упаковывается с применением чехлов из водонепроницаемой пленки с обязательным наличием химически неагрессивных влагопоглотителей (силикагеля).

Для заполнения свободного пространства в упаковочную тару укладываются прокладки из гофрированного картона или пенопласта.

[Эксплуатационная документация](javascript:void(9775)) должна быть уложены в потребительскую тару вместе с программным изделием.

На верхний слой прокладочного материала укладывается [товаросопроводительная документация](javascript:void(9780)) - упаковочный лист и ведомость упаковки.

Потребительская тара должна быть оклеена лентой клеевой 6-70 по ГОСТ 18251-87.

Упакованные в потребительскую тару программные изделия должны быть уложены на поддон, стянуты лентой для предотвращения потери формы груза и упакованы в полиэтиленовую пленку М 0,2 для защиты от попадания влаги.

В коробку поддона должна быть вложена товаросопроводительная документация, в том числе упаковочный лист согласно ГОСТ 25565-88.

Габариты грузового места должны быть не более 1250 • 820 • 1180 мм.

Масса НЕТТО - не более 200 кг.

Масса БРУТТО - не более 220 кг.

Приведен порядок упаковки из ранее разработанного документа на какие-то технические средства. Выглядит несколько необычно в контексте программного изделия. Говоря простым русским языком - полнейший стёб, но требования есть и остаются требованиями.

**Требования к транспортированию и хранению**

В подразделе «Требования к транспортированию и хранению» должны быть указаны для [программного изделия](javascript:void(10682)) условия [транспортирования](javascript:void(9713)), места [хранения](javascript:void(9714)), условия хранения, условия складирования, [сроки хранения](javascript:void(11855)) в различных условиях [из п. 2.4.7 ГОСТ 19.201-78]

В подразделе приведены условия транспортирования и хранения из ранее разработанного документа на какие-то технические средства. Это касается и требований к порядку упаковки. Выглядит несколько необычно в контексте программного изделия.

Заказчик не вправе нарушать условий транспортирования и хранения. Исполнитель сможет отказать заказчику в возврате программного изделия, утверждая, что ненадлежащий внешний вид программного изделия является следствием несоблюдения условий транспортирования и хранения.

**Условия транспортирования и хранения**

Допускается транспортирование программного изделия в транспортной таре всеми видами транспорта (в том числе в отапливаемых герметизированных отсеках самолетов без ограничения расстояний). При перевозке в железнодорожных вагонах вид отправки - мелкий малотоннажный.

При транспортировании и хранении программного изделия должна быть предусмотрена защита от попадания [пыли](javascript:void(16242)) и [атмосферных осадков](javascript:void(16238)). Не допускается кантование программного изделия. Климатические условия транспортирование приведены ниже:

* температура окружающего воздуха, °С - от плюс 5 до плюс 50;
* [атмосферное давление](javascript:void(16249)), кПа - такое-то;
* относительная влажность воздуха при 25 °С - такая-то.

**Специальные требования**

Программа должна обеспечивать взаимодействие с пользователем ([оператором](javascript:void(20435))) посредством [графического пользовательского интерфейса](http://tdocs.su/1808), разработанного согласно рекомендациям компании-производителя операционной системы.

Разработчики настоящего стандарта смотрели в будущее. Не существовало в те годы программ с графическим пользовательским интерфейсом.

**Требования к программной документации**

В разделе «Требования к программной документации» должен быть указан предварительный состав [программной документации](javascript:void(10666)) и, при необходимости, специальные требования к ней [из п. 2.5а ГОСТ 19.201-78]

**Предварительный состав программной документации**

В состав программной документации должны входить:

* [техническое задание](javascript:void(11959));
* [программа и методика испытаний](javascript:void(11958));
* [руководство системного программиста](javascript:void(12012));
* [руководство оператора](javascript:void(12017));
* [ведомость эксплуатационных документов](javascript:void(12037)).

Программа и методика испытаний потребуется, чтобы показать заказчику, что разработанная исполнителем программа соответствует требованиям согласованного и утвержденного технического задания. После проведения совместных (приемо-сдаточных) испытаний заказчик и исполнитель подпишут [акт завершения работы](javascript:void(11386)). И, тем самым, работа будет закрыта, условия договора выполнены.

Допускается объединять отдельные виды [эксплуатационных документов](javascript:void(10564)) (за исключением [ведомости эксплуатационных документов](javascript:void(12037)) и [формуляра](javascript:void(11999))). Необходимость объединения этих документов указывается в [техническом задании](javascript:void(11959)). Объединенному документу присваивают наименование и обозначение одного из объединяемых документов. В объединенных документах должны быть приведены сведения, которые необходимо включать в каждый объединяемый документ [из п. 2.6 ГОСТ 19.101-77]

Но тем, кто впервые занялся разработкой программной документации, лучше придерживаться принципа «мухи отдельно, котлеты отдельно».

**Технико-экономические показатели**

В разделе «Технико-экономические показатели» должны быть указаны: ориентировочная экономическая [эффективность](javascript:void(16613)), предполагаемая годовая потребность, экономические преимущества разработки по сравнению с лучшими отечественными и зарубежными образцами или аналогами [из п. 2.5 ГОСТ 19.201-78]

Ориентировочная экономическая эффективность не рассчитываются. Предполагаемое число использования программы в год – 365 сеансов работы на одном рабочем месте.

Как рассчитать экономическую эффективность? Следовало бы получить от заказчика цифры. Заказчик, в свою очередь, вряд ли заинтересован раскрывать свои финансовые дела. Скорее всего, вопрос отпадет сам собой.

Положим, заказчик оснащает программой десяток рабочих мест. Исполнитель потребовал за разработку $1000. Заказчик мог бы установить на рабочие места программный продукт третьей фирмы, стоимостью $500 за дистрибутив и по $100 за лицензию на каждое рабочее место.

**Экономические преимущества разработки**

Экономические преимущества разработки в сравнении с лучшими отечественными и зарубежными аналогами составит:

|  |  |  |  |
| --- | --- | --- | --- |
| число рабочих мест | аналоги | разработка | экономические преимущества |
| 10 | $1500 | $1000 | $500 |
| 100 | $11500 | $1000 | $10500 |
| и так далее… | ... | ... | ... |
|  |  |  |  |

**Стадии и этапы разработки**

В разделе «Стадии и этапы разработки» устанавливают необходимые [стадии разработки](javascript:void(11989)), этапы и содержание работ (перечень [программных документов](javascript:void(10666)), которые должны быть разработаны, [согласованы](javascript:void(9745)) и [утверждены](javascript:void(9746))), а также, как правило, сроки разработки и определяют [исполнителей](javascript:void(9761)) [из п. 2.6 ГОСТ 19.201-78]

Стадии разработки и этапы регламентированы [ГОСТ 19.102-77](javascript:void(365)). ГОСТ 19.102-77 не препятствует исключению отдельных стадий работ, а также объединению отдельных этапов работ.

**Стадии разработки**

Разработка должна быть проведена в три стадии:

* техническое задание;
* технический (и рабочий) проекты;
* внедрение.

**Этапы разработки**

На стадии «Техническое задание» должен быть выполнен этап разработки, согласования и утверждения настоящего технического задания.

На стадии «Технический (и рабочий) проект» должны быть выполнены перечисленные ниже этапы работ:

* разработка программы;
* разработка программной документации;
* испытания программы.

На стадии «Внедрение» должен быть выполнен этап разработки «Подготовка и передача программы».

**Содержание работ по этапам**

На этапе разработки техзадания должны быть выполнены перечисленные ниже работы:

* постановка задачи;
* определение и уточнение требований к техническим средствам;
* определение требований к программе;
* определение стадий, этапов и сроков разработки программы и документации на нее;
* выбор языков программирования;
* согласование и утверждение технического задания.

На этапе разработки программы должна быть выполнена работа по [программированию](javascript:void(16832)) (кодированию) и [отладке программы](javascript:void(11446)).

На этапе разработки программной документации должна быть выполнена разработка программных документов в соответствии с требованиями ГОСТ 19.101-77.

На этапе испытаний программы должны быть выполнены перечисленные ниже виды работ:

* разработка, согласование и утверждение программы (в ГОСТ, похоже, опечатка – «порядка») и методики испытаний;
* проведение приемо-сдаточных испытаний;
* корректировка программы и программной документации по результатам испытаний.

На этапе подготовки и передачи программы должна быть выполнена работа по подготовке и передаче программы и программной документации в эксплуатацию на объектах заказчика.

**Порядок контроля и приемки**

В разделе «Порядок контроля и приемки» должны быть указаны виды [испытаний](javascript:void(11444)) и общие требования к приемке работы [из п. 2.7 ГОСТ 19.201-78]

**Виды испытаний**

[Приемосдаточные испытания](javascript:void(9704)) должны проводиться на объекте заказчика в сроки…

Приемосдаточные испытания программы должны проводиться согласно разработанной (не позднее такого-то срока) исполнителем и согласованной заказчиком «Программы и методики испытаний».

Ход проведения приемо-сдаточных испытаний заказчик и исполнитель документируют в [протоколе испытаний](javascript:void(9799)).

**Общие требования к приемке работы**

На основании протокола испытаний исполнитель совместно с заказчиком подписывают акт приемки-сдачи программы в эксплуатацию.

**Приложения**

В приложениях к техническому заданию, при необходимости, приводят:

* перечень [научно-исследовательских](javascript:void(9697)) и других работ, обосновывающих разработку;
* схемы [алгоритмов](javascript:void(9065)), таблицы, описания, обоснования, расчеты и другие документы, которые могут быть использованы при разработке;
* другие источники разработки.

[из п. 2.8 ГОСТ 19.201-78]

Если есть, почему не привести. И обязательно выложить перечень ГОСТов, на основании которых должна проводиться разработка. Например:

* ГОСТ 19.201-78. Техническое задание, требования к содержанию и оформлению;
* и так далее..

**Выводы**

Настоящий стандарт, несмотря на свой возраст, позволяет разработать полноценное техническое задание на современную программу с графическим пользовательским интерфейсом. Разработчики ГОСТ 19.201-78 смотрели в будущее и учли практически все аспекты, касающиеся разработки программных средств.

Что осталось неучтенным? Сроки, объемы и этапы финансирования? Техническое задание всегда разрабатывается на основании Договора, письма, [заявки](javascript:void(9786)) и т.д. Указанные сведения должны быть отражены в Договоре.

Каковы спорные моменты? Отсутствие в стандарте конктретных требований, положим, к пользовательскому интерфейсу? Разработчиками стандарта предусмотрен раздел «Специальные требования», возможность добавления новых разделов, допустим, разделов «Дополнительные требования» или «Требования к интерфейсу».

**Вопросы для самопроверки**

1. Какой ГОСТ определяет требования к Техническому заданию?
2. На основании чего разрабатывается ТЗ?
3. Какие разделы входят в ТЗ?
4. В какой раздел можно внести требования к Интерфейсу?
5. Допускается ли вводить новые разделы, объединять существующие?

# Текст программы. Описание программы

**Описание программы (ГОСТ 19.402-78)**

Этот стандарт ориентирован на документирование результирующего продукта разработки.

Строго говоря, существуют два разных документа, имеющих, правда, много общего. Это ОБЩЕЕ ОПИСАНИЕ (ГОСТ 19.502-78) и ОПИСАНИЕ ПРОГРАММЫ (ГОСТ 19.402-78). Однако, в силу того, что реально создать качественно и тот, и другой, не прибегая к почти полному дублированию, выдирая куски, весьма сложно, было бы достаточно реализовать один, более общий, "гибридный" документ. Назовем его "Описанием программы".

На самом деле "Описание программы" в своей содержательной части может дополняться разделами и пунктами, взятыми и из стандартов для других описательных документов и руководств: ГОСТ 19.404-79 ЕСПД. Пояснительная записка, ГОСТ 19.503-79 ЕСПД. Руководство системного программиста, ГОСТ 19.504-79 ЕСПД. Руководство программиста, ГОСТ 19.505-79 ЕСПД. Руководство оператора и т.п. В частности, из Пояснительной записки можно взять схему алгоритма, общее описание алгоритма и (или) функционирования программы, а также обоснование принятых технических и технико-экономических решений.

Описание программы обязательно должно включать информационную часть - аннотацию и содержание.

Основная часть документа должна состоять из вводной части и следующих разделов:

* функциональное назначение;
* описание логики.
* условия применения;
* состав и функции.

В зависимости от особенностей программы допускается введение дополнительных разделов.

В Вводной части документа приводится информация общего характера о программе - полное наименование, обозначение, ее возможные применения и т.п.

Например: *Программа "Автоматизированное рабочее место разработчика САУ" предназначена для … реализована на …. Программа поддерживает …*

В разделе Назначение указывают назначение программы и приводят общее описание функционирования программы, ее основные характеристики, сведения об ограничениях, накладываемых на область применения программы, а также указывают типы электронных вычислительных машин и устройств, которые используются при работе.

Например: *Программа предназначена для решения задач … Программа представляет собой ядро автоматизированного рабочего места …*

*Пользователь имеет возможность …, осуществить …, запустить …, проанализировать …, получить результаты анализа и обработки …, построить … и т.п.*

В разделе "Описание логики" указывают:

* описание структуры программы и ее основных частей

(например: *В состав программы входит следующее:*

* + *пользовательский интерфейс,*
  + *модуль определения путей в графе,*
  + *модуль расчета передаточной функции,*
  + *модуль построения амплитудно- и фазочастотных характеристик,*
  + *модуль построения реакции на полиномиальное воздействие,*
  + *текстовый редактор).*
* описание функций составных частей и связей между ними;

Например*: Программа состоит из шести модулей: интерфейсный модуль; модуль определения …; модуль расчета …; модуль …и т.п..*

*Интерфейсный модуль построен на двух типах диалогов: диалог "вопрос - ответ" и диалог типа "меню". Интерфейсный модуль управляет …*

*Модуль определения … Он является …*

*Модуль расчета …и т.д.*

* сведения о языке программирования;

Например*: Программа написана на языке …с использованием компилятора …*

* описание входных и выходных данных для каждой из составных частей;

Например*: ВХОДНЫЕ ДАННЫЕ. Входными данными для программы является текстовый файл, описывающий расширенную матрицу инциденций графа исследуемой системы.*

*ВЫХОДНЫЕ ДАННЫЕ. Выходными данными являются:*

* *выводимая на экран графическая и текстовая информация (результаты анализа системы);*
* *файлы в одном из графических форматов - копии изображения построенных характеристик (АЧХ,ФЧХ и т.д.);*
* *текстовые файлы - отчеты о проведенных исследованиях;*
* *диагностика состояния системы и сообщения о всех возникших ошибках.*
* описание логики составных частей (при необходимости следует составлять описание схем программ).

При описании логики программы необходима привязка к тексту программы.

В разделе **Состав и функции** указывают описание состава и функции программ, применяемых методов решения задач.

В разделе **Условия применения** указываются условия, необходимые для выполнения программы (требования к необходимым для данной программы техническим средствам, и другим программам, общие характеристики входной и выходной информации, а также требования и условия организационного, технического и технологического характера и т.п.).

Например*: Программа эксплуатируется на персональном компьютере (ПК) типа IBM PC/AT. Для работы в диалоговом режиме используется экран дисплея, клавиатура и манипулятор типа "мышь". Для поддержки графического режима необходим адаптер EGA (VGA). Входные данные хранятся на флоппи- и/или жестком дисках. Программа работает под управлением ОС …*

В приложение к описанию могут быть включены справочные материалы (иллюстрации, таблицы, графики, примеры и т.п.)

И не забудьте указать имя загрузочного модуля, а также описание всей процедуры

Вызова и загрузки системы

Например*: Загрузка программы осуществляется набором в командной строке DOS имени загрузочного модуля – SBM80N.EXE с возможным указанием имени файла данных.*

**ТЕКСТ ПРОГРАММЫ (ГОСТ 19.401-78)**

Требования к оформлению текста программы достаточно просты и естественны для грамотного программиста. Основное, чем требуется руководствоваться при создании этого документа – это то, что текст программы должен быть удобочитаемым.

По-прежнему обязательным является составление информационной части - аннотации и содержания.

Основная часть документа должна состоять из текстов одного или нескольких разделов, которым даны наименования.

Текст каждого программного файла начинается с "шапки", в которой указывается:

* + наименование программы,
  + автор,
  + дата создания программы,
  + номер версии,
  + дата последней модификации.

Обязательными являются комментарии, а также строгое соблюдение правил отступа. Помните, оправдать можно даже неумение создавать программную документацию. А некрасивый текст программы – никогда. Ссылки на то, что этот текст понятен самому автору всерьез не воспринимаются. Тексты программ должно быть не стыдно давать читать другим людям.

Ниже приведен пример подобного хорошо читаемого текста программы (взят с сайта Николая Гехта, e-mail:geht@omskreg.ru, http://users.omskreg.ru/~geht)

/\* Исходные тексты Windows'98

Автор: Nobody Really

Source Code to Windows 98

\*/

#include "win31.h"

#include "win95.h"

#include "evenmore.h"

#include "oldstuff.h"

#include "billrulz.h"

#include "monopoly.h"

#define INSTALL = HARD

char make\_prog\_look\_big[1600000];

void main()

{

while(!CRASHED)

{

display\_copyright\_message();

display\_bill\_rules\_message();

do\_nothing\_loop();

if(first\_time\_installation)

{

make\_50\_megabyte\_swapfile();

do\_nothing\_loop();

totally\_screw\_up\_HPFS\_file\_system();

search\_and\_destroy\_the\_rest\_of\_OS/2();

disable\_Netscape();

disable\_RealPlayer();

disable\_Corel\_Products();

hang\_system();

}

write\_something(anything);

display\_copyright\_message();

do\_nothing\_loop();

do\_some\_stuff();

if(still\_not\_crashed)

{

display\_copyright\_message();

do\_nothing\_loop();

basically\_run\_windows\_3.1();

do\_nothing\_loop();

do\_nothing\_loop();

}

}

if(detect\_cache())

disable\_cache();

if(fast\_cpu())

{

set\_wait\_states(lots);

set\_mouse(speed, very\_slow);

set\_mouse(action, jumpy);

set\_mouse(reaction, sometimes);

}

/\* printf("Welcome to Windows 3.11"); \*/

/\* printf("Welcome to Windows 95"); \*/

printf("Welcome to Windows 98");

if(system\_ok())

crash(to\_dos\_prompt)

else

system\_memory = open("a:\swp0001.swp", O\_CREATE);

while(something)

{

sleep(5);

get\_user\_input();

sleep(5);

act\_on\_user\_input();

sleep(5);

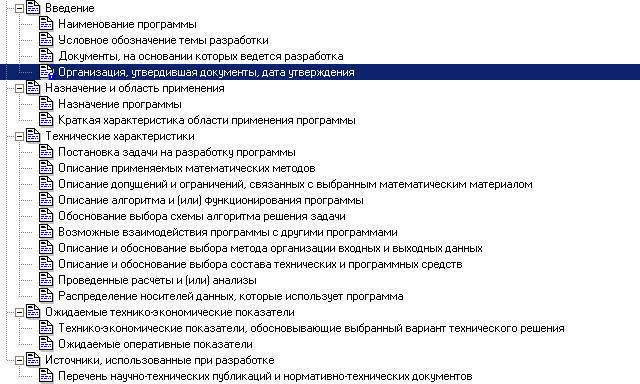
}

create\_general\_protection\_fault();

}

# Пояснительная записка

Пояснительная записка оформляется согласно ГОСТ 19.404-79



**Общие положения**

[Структура](javascript:void(28055)" \o "Структура документа по ГОСТ 2.612-2011" \t "_self) и [оформление](javascript:void(27773)) [документа](javascript:void(20984)) устанавливаются в соответствии с [ГОСТ 19.105-78](javascript:void(12030)).

Составление информационной части ([аннотация](javascript:void(28459)) и [содержание](javascript:void(25195))) является необязательным [из п. 1.1 ГОСТ 19.404-79]

Пояснительная записка должна содержать следующие [разделы](javascript:void(22803)):

* введение;
* назначение и область применения;
* технические характеристики;
* ожидаемые технико-экономические показатели;
* источники, использованные при разработке.

В зависимости от особенностей документа отдельные разделы (подразделы) допускается объединять, а также вводить новые разделы (подразделы) [из п. 1.2 ГОСТ 19.404-79]

**Содержание разделов**

**Введение**

В разделе «Введение» указывают [наименование](javascript:void(23779)) [программы](javascript:void(16831)) и (или) условное обозначение темы разработки, а также документы, на основании которых ведется разработка с указанием [организации](javascript:void(21314)) и даты [утверждения](javascript:void(9746)) [из п. 2.1 ГОСТ 19.404-79]

**Назначение и область применения**

В разделе «Назначение и область применения» указывают назначение программы, краткую характеристику области применения программы [из п. 2.2 ГОСТ 19.404-79]

**Технические характеристики**

Раздел «Технические характеристики» должен содержать следующие подразделы:

* постановка задачи на [разработку](javascript:void(9719)) программы, описание применяемых математических [методов](javascript:void(25228)) и, при необходимости, описание допущений и ограничений, связанных с выбранным математическим материалом;
* описание [алгоритма](javascript:void(9065)) и (или) [функционирования](javascript:void(26260)) программы с обоснованием выбора схемы алгоритма решения задачи, возможные взаимодействия программы с другими программами;
* описание и обоснование выбора метода организации [входных](javascript:void(17031)) и [выходных данных](javascript:void(17032));
* описание и обоснование выбора состава [технических](javascript:void(17000)) и [программных средств](javascript:void(17150)) на основании проведенных расчетов и (или) анализов, распределение [носителей данных](javascript:void(16999)), которые использует программа.

[из п. 2.3 ГОСТ 19.404-79]

**Ожидаемые технико-экономические показатели**

В разделе «Ожидаемые технико-экономические показатели» указывают технико-экономические показатели, обосновывающие выбранного варианта технического решения, а также, при необходимости, ожидаемые оперативные показатели [из п. 2.4 ГОСТ 19.404-79]

**Источники, использованные при разработке**

В разделе «Источники, использованные при разработке» указывают перечень научно-технических [публикаций](javascript:void(23800)), нормативно-технических документов и других [научно-технических материалов](javascript:void(23792)), на которые есть ссылки в основном тексте [из п. 2.5 ГОСТ 19.404-79]

**Приложения**

В приложение к документу могут быть включены таблицы, обоснования, [методики](javascript:void(25229)), расчеты и другие документы, использованные при разработке [из п. 2.6 ГОСТ 19.404-79]

# Руководство Оператора

Руководство Оператора оформляется согласно [ГОСТ 19.505-79](http://www.philosoft.ru/index.php?topic=espd)

Цели и задачи

Оператор по характеру своего взаимодействия с программой во многом похож на [пользователя](http://www.philosoft.ru/index.php?topic=users-manual). От последнего он отличается тем, что напрямую не решает никаких целостных прикладных задач. С другой стороны, в отличие от [администратора](http://www.philosoft.ru/index.php?topic=admins-manual) и [системного администратора](http://www.philosoft.ru/index.php?topic=sysadmins-manual) оператор не занимается настройкой системы. Его работа — в оперативном режиме обслуживать систему, в состав которой входит программа.

Так, в обязанности оператора биллинговой системы может входить ежедневный запуск программ для приема учетных данных от телекоммуникационного оборудования и ежемесячный запуск программы формирования счетов. В обязанности оператора системы потокового ввода анкет может входить проверка правильности распознавания текста и исправление ошибок. Оператор колл-центра принимает и переводит адресатам входящие звонки.

Следовательно, задача руководства оператора в том, чтобы оператор мог четко сделать все, что от него требуется, и никогда или как можно реже оказывался перед необходимостью принимать нетривиальные решения. По крайней мере, в отношении программы. В содержательной части его действия с такой же строгостью могут быть заданы [регламентами](http://www.philosoft.ru/index.php?topic=serv-reglament) или [технологическими инструкциями](http://www.philosoft.ru/iindex.php?topic=tehnoinstr).

Содержание документа

С точки зрения содержания, руководство оператора может рассматриваться как упрощенный вариант [руководства пользователя](http://www.philosoft.ru/index.php?topic=users-manual). При этом в большинстве случаев в руководстве оператора приводятся не описания функций, а как можно более четко сформулированные процедуры решения отдельных задач.

*Методика и стиль изложения*

Руководство оператора должно быть написано как можно более простым языком и давать как можно более четкие инструкции по выполнению предусмотренных операций. Можно сформулировать несколько рекомендаций, которые помогут сделать его таким.

Минимум теоретических введений и концептуальных разделов. Если теория все-таки необходима, она вся должна быть собрана в один раздел.

Минимум явных и неявных ссылок внутри документа. Описания повторяющихся при выполнении разных операций процедур по возможности лучше дублировать.

Минимум ветвлений при описании процедур. Никаких условий вида *«если вы хотите»*. У оператора нет желаний, а есть обязанности.

Оператору важно понимать, все ли у него идет так, как надо. Поэтому необходимо описывать не только его действия, но и результаты таковых в случае успеха.

Типовая структура

Типовая структура руководства оператора приведена в [ГОСТ 19.505-79](http://www.philosoft.ru/index.php?topic=espd). Руководство [оператора](javascript:void(20247)) должно содержать следующие [разделы](javascript:void(22803)):

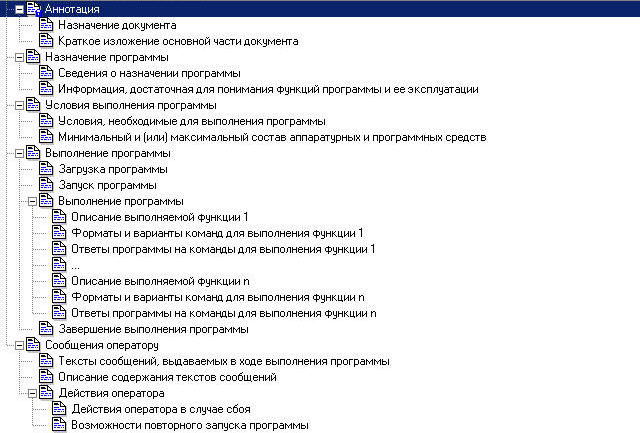
* Назначение программы.
* Условия выполнения программы.
* Выполнение программы.
* Сообщения оператору.

В современных условиях в эту структуру целесообразно добавить раздел «Интерфейс пользователя», актуальный сегодня для большинства программ. В случае программ, предполагающих диалоговый режим работы, можно переименовать раздел «Выполнение программы» в «Выполнение операций» либо описать каждую операцию или группу операций в самостоятельном разделе первого уровня.

В зависимости от особенностей документы допускается объединять отдельные разделы или вводить новые [из п. 1.2 ГОСТ 19.505-79] Последняя фраза предоставляет разработчикам [программной документации](javascript:void(10666)) пространство для маневра.

Рассмотрим составление Руководства пользователя на примере программы Блокнот

[Структура](javascript:void(22800)) [разделов](javascript:void(22803)) руководства оператора по ГОСТ 19.505-79 приведена на рисунке ниже.



В разделе «*Назначение программы*» должны быть указаны сведения о назначении программы и [информация](javascript:void(20458)), достаточная для понимания [функций программы](javascript:void(16611)) и ее [эксплуатации](javascript:void(9718)) [из п. 2.1 ГОСТ 19.505-79]«...должны быть указаны сведения о назначении программы». Сведения о назначении программы изложены в основополагающем документе – в [техническом задании](javascript:void(11959)).

*Функциональное назначение*

[Функциональным](javascript:void(16611)) назначением программы является предоставление [пользователю](javascript:void(16603)) возможности [работы с текстовыми документами](javascript:void(17049)) в формате rtf.

*Эксплуатационное назначение*

Программа должна эксплуатироваться в профильных подразделениях на объектах заказчика.

Пользователями программы должны являться сотрудники профильных подразделений объектов заказчика.

*Состав функций*

Программа обеспечивает возможность выполнения перечисленных ниже функций:

Функции создания нового (пустого) [файла](javascript:void(18693));

Функции открытия (загрузки) существующего файла;

Функции редактирования открытого (далее - текущего) файла путем [ввода](javascript:void(17031)), замены, удаления [содержимого](javascript:void(21032)) файла с применением стандартных устройств ввода;

Функции редактирования текущего файла с применением буфера обмена [операционной системы](javascript:void(17011));

Функции сохранения файла с исходным именем;

Функции сохранения файла с именем, отличным от исходного;

Функции отправки содержимого текущего файла электронной почтой с помощью внешней клиентской почтовой программы;

Функции вывода [оперативных справок](javascript:void(25199)) в строковом [формате](javascript:void(21025)) (подсказок);

Функции интерактивной справочной системы;

Функции отображения [названия](javascript:void(23779)) программы, версии программы, копирайта и комментариев разработчика.

*Условия выполнения программы*

В разделе «Условия выполнения программы» должны быть указаны условия, необходимые для выполнения программы (минимальный и (или) максимальный состав [аппаратурных](javascript:void(17000)) и [программных средств](javascript:void(17150)) и т.п.) [из п. 2.2 ГОСТ 19.505-79]  
Создаем соответствующие подразделы. Поскольку «аппаратурных» звучит старообразно, меняем его на «технических».

Климатические условия эксплуатации

Климатические [условия эксплутатации](javascript:void(10583)), при которых должны обеспечиваться заданные характеристики, должны удовлетворять [требованиям](javascript:void(21350)), предъявляемым к [техническим средствам](javascript:void(17000)) в части условий их эксплуатации.

*Минимальный состав технических средств*

В состав технических средств должен входить IBM-совместимый персональный компьютер ([ПЭВМ](javascript:void(17008))), включающий в себя:

[процессор](javascript:void(17013)) Pentium-1000 с тактовой частотой, ГГц - 10, не менее;

материнскую плату с FSB, ГГц - 5, не менее;

[оперативную память](javascript:void(17017)) объемом, Тб - 10, не менее;

и так далее…

*Минимальный состав программных средств*

[Системные программные средства](javascript:void(16833)), используемые программой, должны быть представлены [лицензионной](javascript:void(9609)) локализованной версией операционной системы. Допускается использование пакета обновления такого-то.

*Требования к персоналу (пользователю)*

Минимальное количество персонала, требуемого для работы программы, должно составлять не менее 2 штатных единиц – системный администратор и пользователь программы – [оператор](javascript:void(20247)).

Системный администратор должен иметь высшее профильное образование и [сертификаты](javascript:void(9807)) компании-производителя операционной системы. В перечень задач, выполняемых системным администратором, должны входить:

задача поддержания [работоспособности](javascript:void(11825)) технических средств;

задачи установки (инсталляции) и поддержания [работоспособности системных программных средств](javascript:void(17181)) – операционной системы;

задача установки (инсталляции) программы.

Пользователь программы (оператор) должен обладать практическими навыками работы с [графическим пользовательским интерфейсом](http://tdocs.su/1808) операционной системы.

Персонал должен быть аттестован на II квалификационную группу по электробезопасности (для работы с конторским оборудованием).

*Выполнение программы*

В разделе «Выполнение программы» должна быть указана последовательность действий оператора, обеспечивающих [загрузку](javascript:void(17033)), запуск, выполнение и завершение программы, приведено описание функций, формата и возможных вариантов [команд](javascript:void(16933)), с помощью которых [оператор](javascript:void(16603)) осуществляет загрузки и управляет выполнением программы, а также ответы программы на эти команды [из п. 2.3 ГОСТ 19.505-79]

Автоматически, «пальцами», создаем подразделы:

Загрузка и запуск программы;

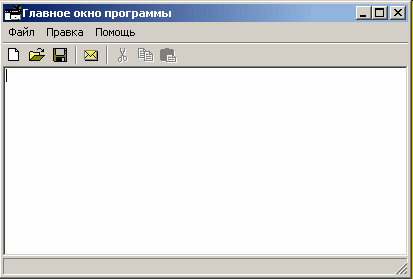
Выполнение программы;

Завершение работы программы.

Загрузка и запуск программы

Загрузка и запуск программы осуществляется способами, детальные сведения о которых изложены в руководстве пользователя операционной системы.

В случае успешного запуска программы на [рабочем столе](javascript:void(23576)) будет отображено [Главное окно](javascript:void(23136)) программы.



*Выполнение программы*

«В подразделе следует привести «описание функций, формата и возможных вариантов команд, с помощью которых оператор … управляет выполнением программы».  
Выше был приведен перечень функций, возможность выполнения которых обеспечивает программа. Для каждой функции из перечня следует создать подраздел.

Выполнение функции сохранения файла с именем, отличным от исходного

...  
Наверное, достаточно. Нет смысла дублировать (фактически) описания выполнения типовых функций программы в [учебно-тренировочном](javascript:void(21708)) документе.

Завершение работы программы

Завершение работы программы обеспечиваются стандартными средствами операционной системы.

или

Выполнение указанной функции возможно любым из перечисленных ниже способов:

последовательным выбором пунктов меню Файл-Выход (см. рисунок такой-то);

нажатием кнопки - Кнопка Закрыть.

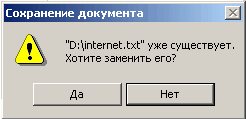
*Сообщения оператору*

В разделе «Сообщения оператору» должны быть приведены тексты [сообщений](javascript:void(16932)), выдаваемых в ходе выполнения программы, описание их содержания и соответствующие действия оператора (действия оператора в случае [сбоя](javascript:void(11842)), возможности повторного запуска программы и т.п.) [из п. 2.4 ГОСТ 19.505-79]

Поскольку программа не консольная (с интерфейсом [командной](javascript:void(16933)) строки), а с графическим пользовательским интерфейсом, классических текстовых сообщений не предвидится. Сообщения об ошибках отображаются в виде окон на рабочем столе.  
«описание их содержания»

Ошибка сохранения файла

При попытке сохранения файла с именем уже существующего файла на рабочем столе программы будет отображено сообщение об ошибке.



«и соответствующие действия оператора»

Для сохранения файла с именем уже существующего файла следует нажать кнопку Да.

Для сохранения файла с именем, отличным от имени существующего файла, следует:

нажать кнопку Нет;

повторно выполнить указания п. [Выполнение функции сохранения файла с именем, отличным от исходного](http://tdocs.su/12354#o12424) с указанием имени файла, отличного от имени существующего файла.

*Об иллюстрациях*

Допускается содержание разделов иллюстрировать поясняющими примерами, таблицами, схемами, графиками [из п. 2.5 ГОСТ 19.505-79]

В настоящем [учебно-тренировочном](javascript:void(21708)) руководстве оператора в качестве иллюстраций используются [экранные формы](javascript:void(11433)) (окна), отображаемые на рабочем столе.

*О приложениях*

В приложения к руководству оператора допускается включать различные материалы, которые нецелесообразно включать в разделы руководства [из п. 2.6 ГОСТ 19.505-79]

**Особенности**

На оператора возложены определенные обязанности по обслуживанию системы. Если что-нибудь пойдет не так, допустим, биллинговая система не сформирует вовремя счета, или с какой-нибудь промышленной установкой случится авария, начнется служебное расследование, а если пострадают люди, то и следствие. Будут выяснять, кто виноват. Один из первых подозреваемых — оператор. Вдруг это он нарушил какие-нибудь правила, вдруг из-за его разгильдяйства все взорвалось и пошло прахом. Окажись это так, многие вздохнут с облегчением (вспомните, кто чаще всего оказывается виноват в крушениях самолетов). Все ли персонал сделал по правилам — вот что заинтересует комиссию в первую очередь. Поэтому от руководства оператора требуется повышенная четкость изложения, а каждое описанное в нем действие оператора должно иметь очевидное содержание, а также внятно описанный, объективный и проверяемый результат.

# Схемы алгоритмов, программ, данных и систем.

В ЕСПД определены правила оформления блок-схем алгоритмов (**ГОСТ 19.701-90**)

Рассматриваемый ГОСТ практически полностью соответствует международному стандарту ISO 5807:1985.

ОПИСАНИЕ СХЕМ

**Схема данных**

Схемы данных отображают путь данных при решении задач и определяют этапы обработки, а также различные применяемые носители данных.

Схема данных состоит из:

* символов данных (символы данных могут также указывать вид носителя данных);
* символов процесса, который следует выполнить над данными (символы процесса могут также указывать функции, выполняемые вычислительной машиной);
* символов линий, указывающих потоки данных между процессами и (или) носителями данных;
* специальных символов, используемых для облегчения написания и чтения схемы.

Символы данных предшествуют и следуют за символами процесса. Схема данных начинается и заканчивается символами данных (за исключением специальных символов).

**Схема программы**

Схемы программ отображают последовательность операций в программе.

Схема программы состоит из:

* символов процесса, указывающих фактические операции обработки данных (включая символы, определяющие путь, которого следует придерживаться с учетом логических условий);
* линейных символов, указывающих поток управления;
* специальных символов, используемых для облегчения написания и чтения схемы.

**Схема работы системы**

Схемы работы системы отображают управление операциями и поток данных в системе.

Схема работы системы состоит из:

* символов данных, указывающих на наличие данных (символы данных могут также указывать вид носителя данных);
* символов процесса, указывающих операции, которые следует выполнить над данными, а также определяющих логический путь, которого следует придерживаться;
* линейных символов, указывающих потоки данных между процессами и (или) носителями данных, а также поток управления между процессами;
* специальных символов, используемых для облегчения написания и чтения блок-схемы.

**Схема взаимодействия программ**

Схемы взаимодействия программ отображают путь активации программ и взаимодействий с соответствующими данными. Каждая программа в схеме взаимодействия программ показывается только один раз (в схеме работы системы программа может изображаться более чем в одном потоке управления).

Схема взаимодействия программ состоит из:

* символов данных, указывающих на наличие данных;
* символов процесса, указывающих на операции, которые следует выполнить над данными;
* линейных символов, отображающих поток между процессами и данными, а также инициации процессов;
* специальных символов, используемых для облегчения написания и чтения схемы.

**Схема ресурсов системы**

Схемы ресурсов системы отображают конфигурацию блоков данных и обрабатывающих блоков, которая требуется для решения задачи или набора задач.

Схема ресурсов системы состоит из:

* символов данных, отображающих входные, выходные и запоминающие устройства вычислительной машины;
* символов процесса, отображающих процессоры (центральные процессоры, каналы и т.д.);
* линейных символов, отображающих передачу данных между устройствами ввода-вывода и процессорами, а также передачу управления между процессорами;
* специальных символов, используемых для облегчения написания и чтения схемы.

Операции обработки данных и носители информации изображаются на схеме соответствующими ***блоками.*** Большая часть блоков по построению условно вписана в прямоугольник со сторонами *а и b.* Минимальное значение ***а* = 10 мм**, увеличение *а* производится на число, кратное **5 мм**. Размер **b=1,5a**. Для отдельных блоков допускается соотношение между *а и b,* равное 1:2. В пределах одной схемы рекомендуется изображать блоки одинаковых размеров. Все блоки нумеруются. Виды и назначение основных блоков приведены в табл.

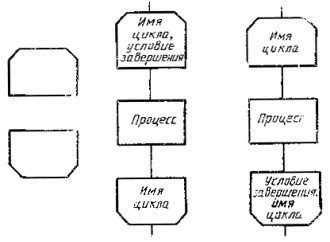
*Таблица Условные обозначения блоков схем алгоритмов*

|  |  |  |
| --- | --- | --- |
| **Наименование** | Обозначение | **Функции** |
| Процесс | http://kdinf.ru/algoritm/images/image001.gif | Выполнение операции или группы операций, в результате которых изменяется значение, форма представления или расположение данных. |
| Ввод-вывод | http://kdinf.ru/algoritm/images/image002.gif | Преобразование данных в форму, пригодную для обработки (ввод) или отображения результатов обработки (вывод). |
| Решение | http://kdinf.ru/algoritm/images/image003.gif | Выбор направления выполнения алгоритма в зависимости от некоторых переменных условий. |
| Предопределенный процесс | http://kdinf.ru/algoritm/images/image004.gif | Использование ранее созданных и отдельно написанных программ (подпрограмм). |
| Документ | http://kdinf.ru/algoritm/images/image005.gif | Вывод данных на бумажный носитель. |
| Магнитный диск | http://kdinf.ru/algoritm/images/image006.gif | Ввод-вывод данных, носителем которых служит магнитный диск. |
| Пуск-останов | http://kdinf.ru/algoritm/images/image007.gif | Начало, конец, прерывание процесса обработки данных. |
| Соединитель | http://kdinf.ru/algoritm/images/image008.gif | Указание связи между прерванными линиями, соединяющими блоки. |
| Межстраничный соединитель | http://kdinf.ru/algoritm/images/image009.gif | Указание связи между прерванными линиями, соединяющими блоки, расположенные на разных листах. |
| Комментарий | http://kdinf.ru/algoritm/images/image010.gif | Связь между элементом схемы и пояснением. |

Граница цикла

Символ, состоящий из двух частей, отображает начало и конец цикла. Обе части символа имеют один и тот же идентификатор. Условия для инициализации, приращения, завершения и т.д. помещаются внутри символа в начале или в конце в зависимости от расположения операции, проверяющей условие.

Пример:



**Линия**

Символ отображает поток данных или управления.

http://snipov.net/snip/7/7707/x037.jpg

При необходимости или для повышения удобочитаемости могут быть добавлены стрелки-указатели.

**Специфические символы линий**

Передача управления

Символ отображает непосредственную передачу управления от одного процесса к другому, иногда с возможностью прямого возвращения к инициирующему процессу после того, как инициированный процесс завершит свои функции. Тип передачи управления должен быть назван внутри символа (например, запрос, вызов, событие).

http://snipov.net/snip/7/7707/x039.gif

Канал связи

Символ отображает передачу данных по каналу связи.

http://snipov.net/snip/7/7707/x041.jpg

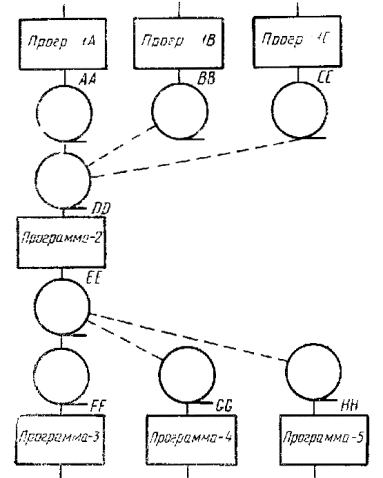
Пунктирная линия

Символ отображает альтернативную связь между двумя или более символами. Кроме того, символ используют для обведения аннотированного участка.

http://snipov.net/snip/7/7707/x043.jpg

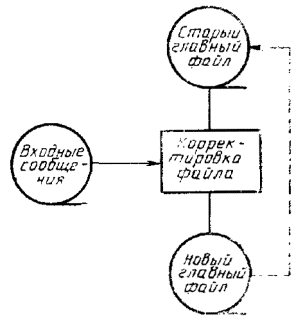
Пример

Если один из ряда альтернативных выходов используют в качестве входа в процесс либо когда выход используется в качестве входа в альтернативные процессы, эти символы соединяют пунктирными линиями.



Пример

Выход, используемый в качестве входа в следующий процесс, может быть соединен с этим входом с помощью пунктирной линии.



**Специальные символы**

Соединитель

Символ отображает выход в часть схемы и вход из другой части этой схемы и используется для обрыва линии и продолжения ее в другом месте. Соответствующие символы-соединители должны содержать одно и то же уникальное обозначение.

http://snipov.net/snip/7/7707/x049.gif

Терминатор

Символ отображает выход во внешнюю среду и вход из внешней среды (начало или конец схемы программы, внешнее использование и источник или пункт назначения данных).

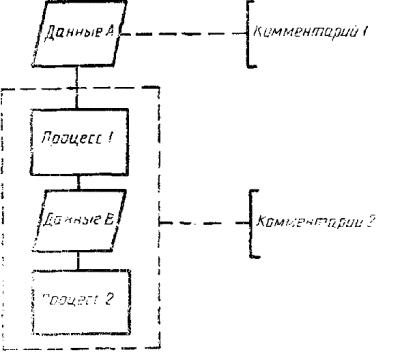
http://snipov.net/snip/7/7707/x051.gif

Комментарий

Символ используют для добавления описательных комментариев или пояснительных записей в целях объяснения или примечаний. Пунктирные линии в символе комментария связаны с соответствующим символом или могут обводить группу символов. Текст комментариев или примечаний должен быть помещен около ограничивающей фигуры.



Пример.



Пропуск

Символ (три точки) используют в схемах для отображения пропуска символа или группы символов, в которых не определены ни тип, ни число символов. Символ используют только в символах линии или между ними. Он применяется главным образом в схемах, изображающих общие решения с неизвестным числом повторений.



Пример.



**Правила применения символов и выполнения схем**

Правила применения символов

Символ предназначен для графической идентификации функции, которую он отображает, независимо от текста внутри этого символа.

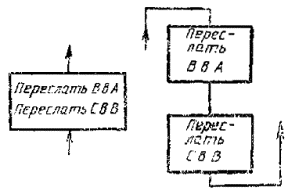
Символы в схеме должны быть расположены равномерно. Следует придерживаться разумной длины соединений и минимального числа длинных линий.

Большинство символов задумано так, чтобы дать возможность включения текста внутри символа. Формы символов, установленные настоящим стандартом, должны служить руководством для фактически используемых символов. Не должны изменяться углы и другие параметры, влияющие на соответствующую форму символов. Символы должны быть, по возможности, одного размера.

Символы могут быть вычерчены в любой ориентации, но, по возможности, предпочтительной является горизонтальная ориентация. Зеркальное изображение формы символа обозначает одну и ту же функцию, но не является предпочтительным.

Минимальное количество текста, необходимого для понимания функции данного символа, следует помещать внутри данного символа. Текст для чтения должен записываться слева направо и сверху вниз независимо от направления потока.

Пример.

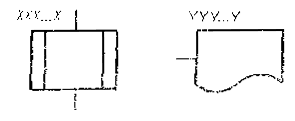


Если объем текста, помещаемого внутри символа, превышает его размеры, следует использовать символ комментария.

Если использование символов комментария может запутать или разрушить ход схемы, текст следует помещать на отдельном листе и давать перекрестную ссылку на символ.

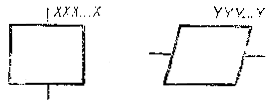
В схемах может использоваться идентификатор символов. Это связанный с данным символом идентификатор, который определяет символ для использования в справочных целях в других элементах документации (например, в листинге программы). Идентификатор символа должен располагаться слева над символом.

Пример.



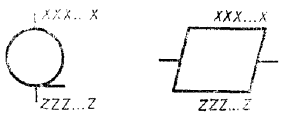
В схемах может использоваться описание символов - любая другая информация, например, для отображения специального применения символа с перекрестной ссылкой, или для улучшения понимания функции как части схемы. Описание символа должно быть расположено справа над символом.

Пример.



В схемах работы системы символы, отображающие носители данных, во многих случаях представляют способы ввода-вывода. Для использования в качестве ссылки на документацию текст на схеме для символов, отображающих способы вывода, должен размещаться справа над символом, а текст для символов, отображающих способы ввода - справа под символом.

Пример.

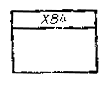
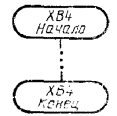


В схемах может использоваться подробное представление, которое обозначается с помощью символа с полосой для процесса или данных. Символ с полосой указывает, что в этом же комплекте документации в другом месте имеется более подробное представление.

Символ с полосой представляет собой любой символ, внутри которого в верхней части проведена горизонтальная линия. Между этой линией и верхней линией символа помещен идентификатор, указывающий на подробное представление данного символа.

В качестве первого и последнего символа подробного представления должен быть использован символ указателя конца. Первый символ указателя конца должен содержать ссылку, которая имеется также в символе с полосой.

Символ с полосой                                           Подробное представление

**Правила выполнения соединений**

Потоки данных или потоки управления в схемах показываются линиями. Направление потока слева направо и сверху вниз считается стандартным.

В случаях, когда необходимо внести большую ясность в схему (например, при соединениях), на линиях используются стрелки. Если поток имеет направление, отличное от стандартного, стрелки должны указывать это направление.

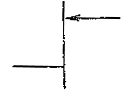
В схемах следует избегать пересечения линий. Пересекающиеся линии не имеют логической связи между собой, поэтому изменения направления в точках пересечения не допускаются.

Пример.

http://snipov.net/snip/7/7707/x070.gif

Две или более входящие линии могут объединяться в одну исходящую линию. Если две или более линии объединяются в одну линию, место объединения должно быть смещено.

Пример.



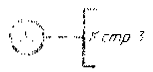
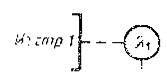
Линии в схемах должны подходить к символу либо слева, либо сверху, а исходить либо справа, либо снизу. Линии должны быть направлены к центру символа.

При необходимости линии в схемах следует разрывать для избежания излишних пересечений или слишком длинных линий, а также, если схема состоит из нескольких страниц. Соединитель в начале разрыва называется внешним соединителем, а соединитель в конце разрыва - внутренним соединителем.

Ссылки к страницам могут быть приведены совместно с символом комментария для их соединителей.

Пример.

Внешний соединитель                                 Внутренний соединитель

**Специальные условные обозначения**

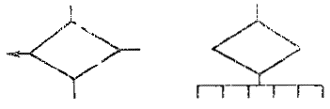
Несколько выходов

Несколько выходов из символа следует показывать:

1) несколькими линиями от данного символа к другим символам;

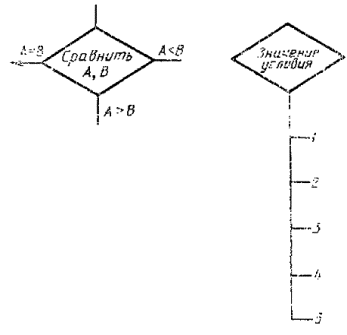
2) одной линией от данного символа, которая затем разветвляется в соответствующее число линий.

Примеры.



Каждый выход из символа должен сопровождаться соответствующими значениями условий, чтобы показать логический путь, который он представляет, с тем, чтобы эти условия и соответствующие ссылки были идентифицированы.

Примеры.



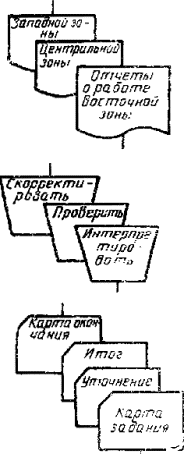
**Повторяющееся представление**

Вместо одного символа с соответствующим текстом могут быть использованы несколько символов с перекрытием изображения, каждый из которых содержит описательный текст (использование или формирование нескольких носителей данных или файлов, производство множества копий печатных отчетов или форматов перфокарт).

Когда несколько символов представляют упорядоченное множество, это упорядочение должно располагаться от переднего (первого) к заднему (последнему).

Линии могут входить или исходить из любой точки перекрытых символов, однако требования [п. 4.2.4](http://snipov.net/c_4712_snip_100096.html#i74295) должны соблюдаться. Приоритет или последовательный порядок нескольких символов не изменяется посредством точки, в которой линия входит или из которой исходит.

Пример.



Линии, соединяющие блоки и указывающие последовательность связей между ними, должны проводится параллельно линиям рамки. Стрелка в конце линии может не ставиться, если линия направлена слева направо или сверху вниз. В блок может входить несколько линий, то есть блок может являться преемником любого числа блоков. Из блока (кроме логического) может выходить только одна линия. Логический блок может иметь в качестве продолжения один из двух блоков, и из него выходят две линии. Если на схеме имеет место слияние линий, то место пересечения выделяется точкой. В случае, когда одна линия подходит к другой и слияние их явно выражено, точку можно не ставить.

Схему алгоритма следует выполнять как единое целое, однако в случае необходимости допускается обрывать линии, соединяющие блоки.

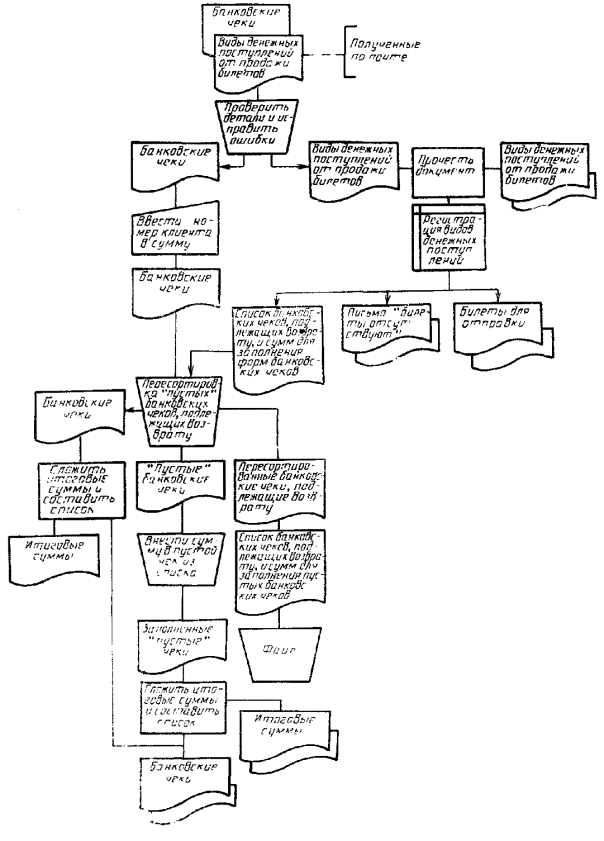
Если при обрыве линии продолжение схемы находится на этом же листе, то на одном и другом конце линии изображается специальный символ соединитель — окружность диаметром 0,5 а. Внутри парных окружностей указывается один и тот же идентификатор. В качестве идентификатора, как правило, используется порядковый номер блока, к которому направлена соединительная линия.

Если схема занимает более одного листа, то в случае разрыва линии вместо окружности используется межстраничный соединитель. Внутри каждого, соединителя указывается адрес — откуда и куда направлена соединительная линия. Адрес записывается в две строки: в первой указывается номер листа, во второй — порядковый номер блока.

Блок-схема должна содержать все разветвления, циклы и обращения к подпрограммам, содержащиеся в программе.

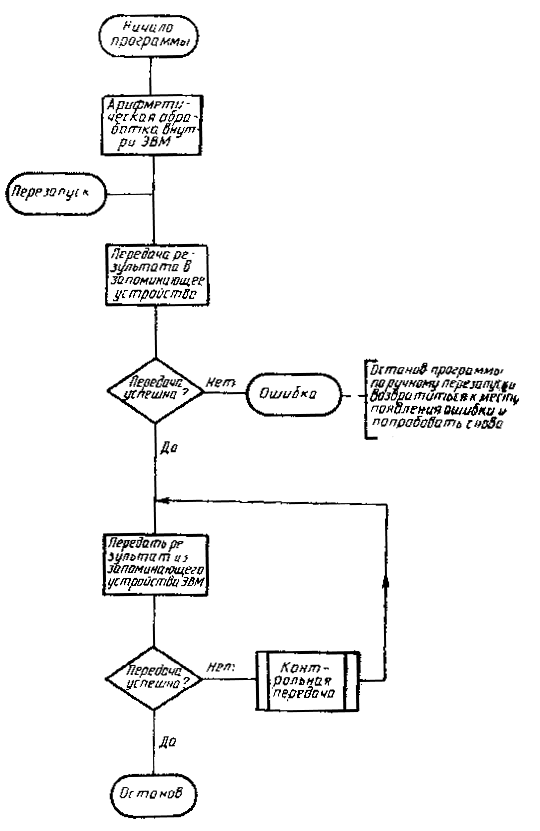
**Примеры выполнения схем**

1. Схема данных

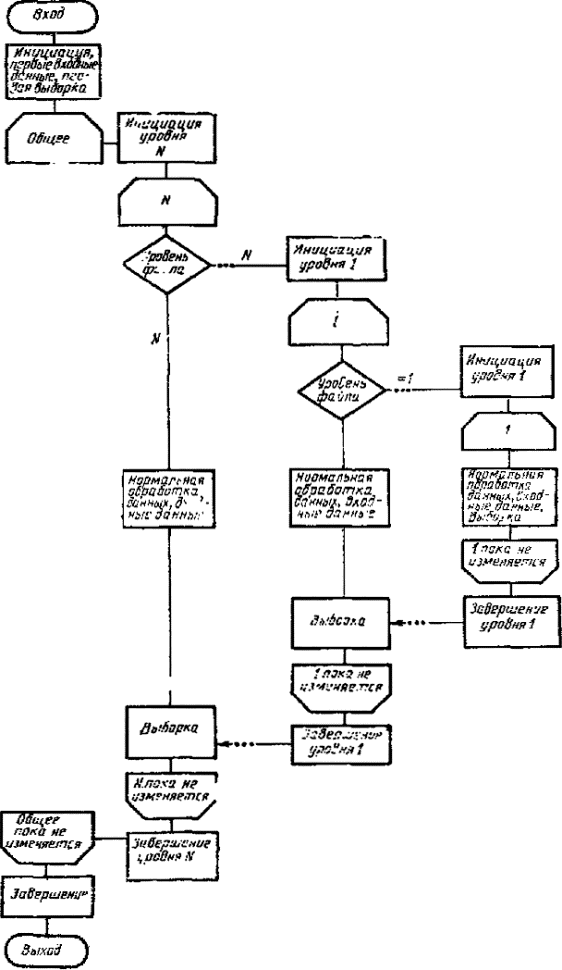


Схемы программы

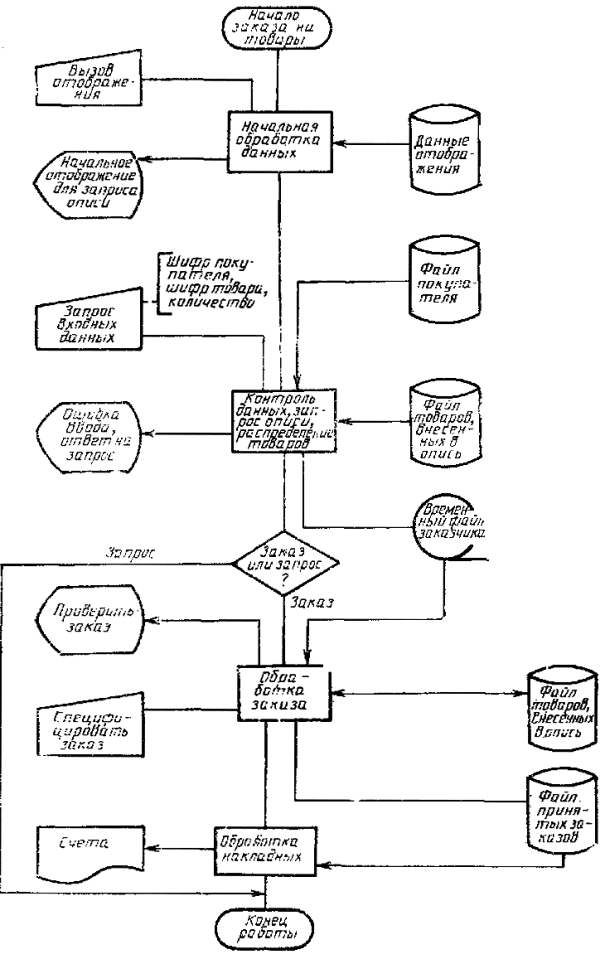
**Пример 1.**



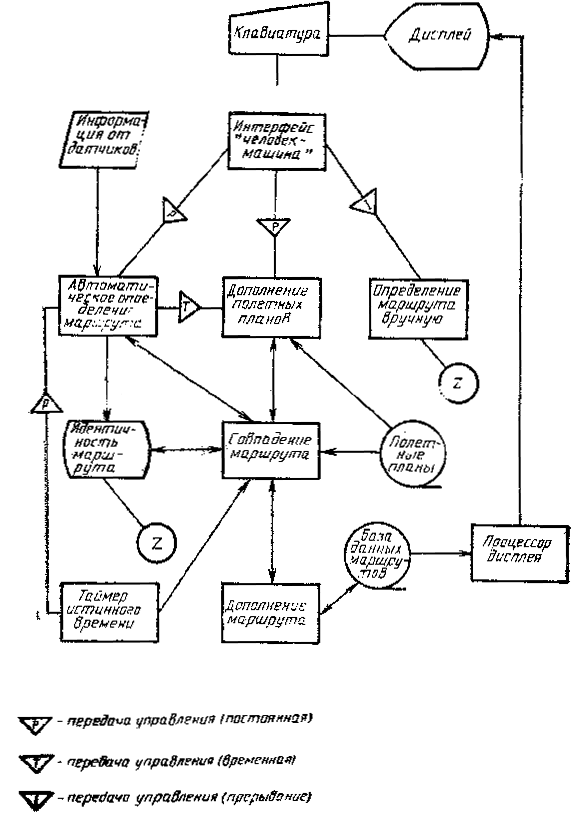
**Пример 2.**



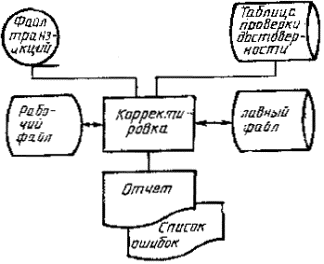
3. Схема работы системы



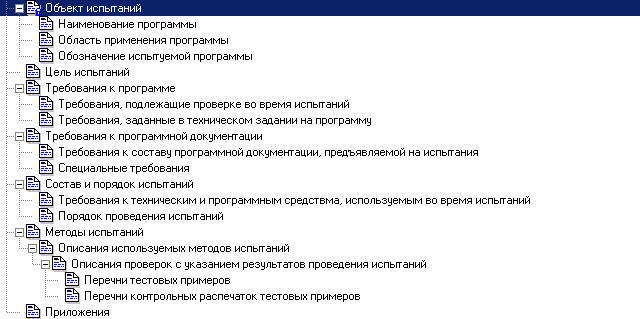
4. Схема взаимодействия программ



5. Схема ресурсов системы



# Порядок и методика испытаний



Структура и оформление документа устанавливается в соответствии с [ГОСТ 19.105-78](javascript:void(12030)). Составление информационной части (аннотации и содержания) является необязательным [из п. 1.1 ГОСТ 19.301-79]

Необязательным – и не надо.

|  |
| --- |
|  |

**Разделы документа**

Документ «Программа и методика испытаний» должен содержать следующие [разделы](javascript:void(22803)):

* объект [испытаний](javascript:void(11444));
* цель испытаний;
* [требования](javascript:void(21350)) к [программе](javascript:void(16831));
* требования к [программной документации](javascript:void(10666));
* состав и порядок испытаний;
* [методы испытаний](javascript:void(9798)).

В зависимости от особенностей [документа](javascript:void(20984)) допускается вводить дополнительные разделы [из п. 1.2 ГОСТ 19.301-79]

Разработчики стандарта в очередной раз обеспечили исполнителю пространство для маневра.

**Содержание разделов**

Имеет смысл дополнить ПМ, разработанную по [ГОСТ 19.301-79](javascript:void(407)), сведениями «системотехнического» характера. Поскольку разработчики [стандарта](javascript:void(21264)) предоставили возможность дополнять ПМ, следует воспользоваться этим и открыть  [РД 50-34.698-90](http://tdocs.su/taxonomy/term/54).

Такой подход оправдан. В договоре не должно, по идее, быть ни слова по существу организации и проведения испытаний. В договоре просто указывается, что испытания должны иметь место.

**Объект испытаний**

В разделе «Объект испытаний» указывают [наименование](javascript:void(23779)), область применения и обозначение испытуемой программы [из п. 2.1 ГОСТ 19.301-79]

Перечисленные сведения заимствуются из соответствующих [разделов](javascript:void(22803)) [технического](javascript:void(12282)) задания.

**Наименование**

[Наименование](javascript:void(23779)) – «Текстовый редактор для работы с файлами формата rtf».

**Область применения**

Программа предназначена к применению в профильных подразделениях на объектах заказчика.

**Обозначение программы**

Наименование темы разработки – «Разработка текстового редактора для работы с файлами формата rtf».

**Цель испытаний**

В разделе «Цель испытаний» должна быть указана цель проведения испытаний [из п. 2.2 ГОСТ 19.301-79]

Цель проведения испытаний - проверка соответствия характеристик разработанной [программы](javascript:void(16831)) ([программного изделия](javascript:void(10682))) функциональным и отдельным иным видам [требований](javascript:void(21350)), изложенным в документе Техническое задание.

Фактически, цель проведения испытаний – сдача работы заказчику. Формально – подтверждение соответствия функциональных и иных характеристик разработанной программы требованиям, сформулированным в техническом задании.

**Общие положения**

Подраздел заимствован из [РД 50-34.698-90](http://tdocs.su/taxonomy/term/54). Названия подразделов несколько изменены.

**Основания для проведения испытаний**

Испытания проводятся на основании Приказа Директора ФГУП «Спецтяжмонтажстройсельхозавтоматика» за № таким-то от такого-то 2004 г.

Основанием проведения испытаний является Приказ о проведении испытаний с составом [приемочной комиссии](javascript:void(9759)). Документ разрабатывается согласно, к примеру, гл. 6  РД 50-34.698-90.

Следующий подраздел заимствуется из технического задания.

**Место и продолжительность испытаний**

[Приемосдаточные испытания](javascript:void(9704)) должны проводиться на объекте заказчика в сроки…

Приемосдаточные испытания программы должны проводиться согласно разработанной (не позднее такого-то срока) исполнителем и согласованной с заказчиком Программы и методики испытаний.

Ход проведения приемо-сдаточных испытаний заказчик и исполнитель документируют в [Протоколе испытаний](javascript:void(9799)).

**Организации, участвующие в испытаниях**

Испытания проводятся комиссией, в состав которой входят представители [организаций](javascript:void(21314)) заказчика и исполнителя. Состав комиссии утверждается [Приказом](javascript:void(11414)).

Поскольку в техническом задании нет указанного пункта, для проведения испытаний достаточно представителей заказчика и исполнителя. В случае привлечения сторонних организаций следует указать их минимальные [реквизиты](javascript:void(21023)).

**Перечень документов, предъявляемых на испытания**

Состав программной документации должен включать в себя:

* [техническое задание](javascript:void(11959));
* [программу и методику испытаний](javascript:void(11958));
* [руководство системного программиста](javascript:void(12012));
* [руководство оператора](javascript:void(12017));
* [ведомость эксплуатационных документов](javascript:void(12037)).

Перечень заимствуется из п. «Предварительный состав программной документации» технического задания. В случае необходимости в ходе разработки перечень может быть пополнен. Стоит согласовать с заказчиком.

**Объем испытаний**

Подраздел заимствован из  РД 50-34.698-90.

**Перечень этапов испытаний**

Испытания проводятся в два этапа:

* ознакомительный;
* испытания.

**Перечень проверок, проводимых на 1 этапе испытаний**

Перечень проверок, проводимых на 1-м этапе испытаний, должен включать в себя:

* [проверку](javascript:void(21340)) комплектности [программной документации](javascript:void(10666));
* проверку комплектности состава технических и программных средств.

[Методика](javascript:void(25229)) проведения проверок, входящих в перечень по 1-му этапу испытаний, изложена в документе...

Методику проведения удобно вынести в приложение. Программа - отдельно, методика – отдельно.

**Перечень проверок, проводимых на 2 этапе испытаний**

Перечень проверок, проводимых на 2-м этапе испытаний, должен включать в себя:

* проверку соответствия технических характеристик программы;
* проверку степени выполнения требований [функционального назначения программы](javascript:void(16611)).

Методика проведения проверок, входящих в перечень по 2-му этапу испытаний, изложена в документе...

**Количественные и качественные характеристики, подлежащие оценке**

Количественные характеристики – все, что можно взвесить, измерить или просто сосчитать. Качественные – не требующие проведения измерений.

**Количественные характеристики, подлежащие оценке**

В ходе проведения [приемосдаточных испытаний](javascript:void(9704)) оценке подлежат количественные характеристики, такие как:

* комплектность [программной документации](javascript:void(11448));
* комплектность состава технических и программных средств.

**Качественные характеристики, подлежащие оценке**

В ходе проведения приемосдаточных испытаний оценке подлежат качественные (функциональные) характеристики программы. Проверке подлежит возможность выполнения программой перечисленных ниже функций:

* функции создания нового (пустого) файла.
* функции открытия (загрузки) существующего файла.
* функции редактирования открытого (далее - текущего) файла путем ввода, [замены](javascript:void(22833)), [удаления](javascript:void(22823)) содержимого файла с применением стандартных [устройств ввода](javascript:void(23329)).
* функции [редактирования](javascript:void(22797)) текущего файла с применением [буфера обмена](javascript:void(22840)) [операционной системы](javascript:void(17011)).
* функции [сохранения](javascript:void(22824)) файла с исходным [именем](javascript:void(23779)).
* функции сохранения файла с именем, отличным от исходного.
* функции отправки содержимого текущего файла электронной почтой с помощью внешней клиентской почтовой программы.
* функции вывода [оперативных справок](javascript:void(25199)) в [строковом](javascript:void(22807)) [формате](javascript:void(22805)) (подсказок).
* функции интерактивной справочной системы.
* функции отображения названия программы, версии программы, копирайта и комментариев разработчика.

Приведенный выше перечень – результат вставки в настоящий документ п. «Требования к составу выполняемых функций» из технического задания.

**Последовательность проведения и режимы испытаний**

Испытания проводятся в последовательности, указанной в п. «Перечень этапов испытаний».

**Перечень работ, проводимых после завершения испытаний**

В случае успешного проведения испытаний в полном объеме исполнитель совместно с заказчиком на основании Протокола испытаний утверждают Акт приемки-сдачи работ… ([Акт завершения работ согласно п.1 РД 50-34.698-90](javascript:void(11386))).

Исполнитель передает заказчику программное изделие, [программную (эксплуатационную) документацию](javascript:void(10666)) и т.д.

В случае выявления несоответствия разработанной программы отдельным требованиям техзадания исполнитель проводит корректировку программы и программной документации по результатам испытаний в сроки, согласованные с заказчиком.

По завершении корректировки программы и программной документации исполнитель и заказчик проводят повторные испытания согласно настоящей программы и методик в объеме, требуемом для проверки проведения корректировок.

Мелкие, несущественные недоработки могут быть устранены в рабочем порядке.

**Требования к программе**

В разделе «Требования к программе» должны быть указаны требования, подлежащие [проверке](javascript:void(11447)) во время испытаний и заданные в [техническом задании на программу](javascript:void(11959)) [из п. 2.3 ГОСТ 19.301-79]

При проведении испытаний функциональные характеристики (возможности) программы подлежат проверке на соответствие требованиям, изложенным в п. «Требования к составу выполняемых функций» технического задания.

Подлежат проверке требования, результат выполнения которых можно взвесить, измерить, посчитать. Выполнение многих требований бывает очевидно. Например требование «Программа должна обеспечивать свое выполнение под управлением операционной системы такой-то». Развернутый перечень требований предъявлять не обязательно, поскольку техническое задание входит в состав программных документов, предъявляемых для проведения испытаний. Но желательно.

**Требования к программной документации**

В разделе «Требования к программной документации» должны быть указаны состав [программной документации](javascript:void(10668)), предъявляемой на испытания, а также специальные требования, если они заданы в техническом задании на программу [из п. 2.4 ГОСТ 19.301-79]

Состав программной документации должен включать в себя:

* техническое задание;
* программу и методику испытаний;
* руководство системного программиста;
* руководство оператора;
* ведомость эксплуатационных документов.

В подраздел следует вставить содержание п. «Предварительный состав программной документации» технического задания. Что и сделано.

**Средства и порядок испытаний**

В разделе «Средства и порядок испытаний» должны быть указаны [технические](javascript:void(17000)) и [программные средства](javascript:void(16601)), используемые во время испытаний, а также порядок проведения испытаний [из п. 2.7 ГОСТ 19.301-79]

**Технические средства, используемые во время испытаний**

В состав технических средств должен входить IBM-совместимый персональный компьютер (ПЭВМ), включающий в себя:

* [процессор](javascript:void(17013)) Pentium-1000 с тактовой частотой, ГГц - 10, не менее;
* материнскую плату с FSB, ГГц - 5, не менее;
* [оперативную память](javascript:void(17017)) объемом, Тб - 10, не менее;
* и так далее…

Испытания проводятся на технических средствах, перечень которых следует позаимствовать из подраздела «Требования к составу и параметрам технических средств» технического задания. Испытания проводятся с использованием программных средств, состав которых приведен в подразделе «Требования к программным средствам, используемым программой» технического задания. Должна, очевидно, иметь место и разработанная программа.

**Программные средства, используемые во время испытаний**

Системные программные средства, используемые программой, должны быть представлены [лицензионной](javascript:void(9609)) локализованной версией [операционной системы](javascript:void(17011)). Допускается использование пакета обновления такого-то.

Для проведения испытаний предоставляется инсталляционная (установочная) версия разработанной программы.

**Порядок проведения испытаний**

Испытания должны проводиться поэтапно согласно п. «Перечень этапов испытаний» настоящего документа.

**Условия проведения испытаний**

Испытания должны проводиться в нормальных климатических условиях по ГОСТ 22261-94. Условия проведения испытаний приведены ниже:

* температура окружающего воздуха, °С - 20 ± 5;
* относительная влажность, % - от 30 до 80;
* [атмосферное давление](javascript:void(16249)), кПа - от 84 до 106;
* частота питающей электросети, Гц - 50 ± 0,5;
* напряжение питающей сети переменного тока, В - 220 ± 4,4.

**Условия начала и завершения отдельных этапов испытаний**

Необходимым и достаточным условием завершения 1 этапа испытаний и начала 2 этапа испытаний является успешное завершение проверок, проводимых на 1 этапе (см. п. «Перечень проверок, проводимых на 1 этапе испытаний»).

Условием завершения 2 этапа испытаний является успешное завершение проверок, проводимых на 2 этапе испытаний.

**Ограничения в условиях проведения испытаний**

Климатические [условия эксплуатации](javascript:void(10583)), при которых должны обеспечиваться заданные характеристики, должны удовлетворять требованиям, предъявляемым к техническим средствам в части условий их эксплуатации.

**Требования к техническому обслуживанию**

Требования к [техническому обслуживанию](javascript:void(10887)) не предъявляются.

Бензином, соляркой программу заправлять не надо, масло менять не надо.

**Меры, обеспечивающие безопасность и безаварийность проведения испытаний**

При проведении испытаний заказчик должен обеспечить соблюдение требований безопасности, установленных ГОСТ 12.2.007.0–75, ГОСТ 12.2.007.3-75, «Правилами техники безопасности при эксплуатации электроустановок потребителей», и «Правилами технической эксплуатации электроустановок потребителей».

Поскольку (согласно подразделу «Виды испытаний» технического задания) испытания проводятся на объекте заказчика, заказчик должен обеспечить соблюдение мер безопасности.

**Порядок взаимодействия организаций, участвующих в испытаниях**

Исполнитель письменно извещает заказчика о готовности к проведению приемо-сдаточных испытаний не позднее чем за 14 дней до намеченного срока проведения испытаний.

Заказчик Приказом назначает срок проведения испытаний и приемочную комиссии, которая должна включать в свой состав представителей заказчика и исполнителя.

Заказчик письменно извещает сторонние организации, которые должны принять участие в приемо-сдаточных испытаниях.

Заказчик совместно с исполнителем проводят все подготовительные мероприятия для проведения испытаний на объекте заказчика, а так же проводят испытания в соответствии с настоящей программой и методиками.

Заказчик осуществляет контроль проведения испытаний, а также документирует ход проведения проверок в Протоколе проведения испытаний.

**Требования к персоналу, проводящему испытания**

Персонал, проводящий испытания, должен быть аттестован на II квалификационную группу по электробезопасности (для работы с конторским оборудованием).

**Методы испытаний**

В разделе «Методы испытаний» должны быть приведены описания используемых [методов испытаний](javascript:void(24702)). [Методы](javascript:void(25228)) испытаний рекомендуется по отдельным показателям располагать в последовательности, в которой эти показатели расположены в разделах «Требования к программе» и «Требования к программной документации».

В методах испытаний должны быть приведены описания проверок с указанием [результатов проведения испытаний](javascript:void(24714)) (перечней тестовых примеров, контрольных распечаток тестовых примеров и т. п.) [из п. 2.8 ГОСТ 19.301-79]. Сведения о методах проведения испытаний изложены в документах Приложение А и Приложение Б.

**Приложения**

В приложение к документу могут быть включены тестовые примеры, контрольные распечатки тестовых примеров, таблицы, графики и т. п. [из п. 2.9 ГОСТ 19.301-79]

**Приложение А**

(обязательное)

**Методы проведения проверки комплектности программной документации**

Проверка комплектности программной документации на программное изделие производится визуально представителями заказчика. В ходе проверки сопоставляется состав и комплектность программной документации, представленной исполнителем, с перечнем программной документации.

Проверка считается завершенной в случае соответствия состава и комплектности программной документации, представленной исполнителем, перечню программной документации, приведенному в указанном выше пункте.

По результатам проведения проверки представитель заказчика вносит запись в Протокол испытаний - «Комплектность программной документации соответствует (не соответствует) требованиям п. Перечень документов, предъявляемых на испытания настоящего документа».

Протокол испытаний – п. 7  РД 50-34.698-90.

**Методы проведения проверки комплектности и состава технических и программных средств**

Проверка комплектности и состава технических и программных средств производится визуально представителем заказчика. В ходе проверки сопоставляется состав и комплектность технических и программных средств, представленных исполнителем, с перечнем технических и программных средств, приведенным в Табл Х настоящего документа.

Комплектность системного блока, входящего в состав технических средств, может производиться по бланку заказа, если системный блок опечатан производителем или продавцом. Комплектность программных средств проводится также визуально. Загрузилась операционная система, высветился логотип, версия – соответствует/не соответствует заявленной в техническом задании.

Проверка считается завершенной в случае соответствия состава и комплектности технических и программных средств, представленных исполнителем, с перечнем технических и программных средств.

По результатам проведения проверки представитель заказчика вносит запись в Протокол испытаний - «Комплектность технических и программных средств соответствует (не соответствует) требованиям п. Технические средства, используемые во время испытаний настоящего документа».

**Приложение Б**

(обязательное)

Руководство оператора должно содержать подробные сведения о реализации всех функций программы. Чтобы не копировать указанные сведения в настоящий документ, достаточно ограничиться ссылками на подразделы Руководства оператора.

**Методы проверки выполнения функции создания нового (пустого) файла**

Проверка выполнения указанной функции выполняется согласно п. «Выполнение функции создания нового (безымянного) файла» руководства оператора.

Проверка считается завершенной в случае соответствия состава и последовательности действий оператора при выполнении данной функции указанному выше подразделу руководства оператора.

По результатам проведения проверки представитель заказчика вносит запись в Протокол испытаний - «п. такой-то выполнен».

Формализованное изложение («указанной», «данному» и пр.) позволяет копировать все три абзаца из подраздела в подраздел, меняться будет только номер подраздела руководства оператора.

**Методы проверки выполнения функции открытия (загрузки) существующего файла**

Проверка выполнения указанной функции выполняется согласно п. «Выполнение функции открытия (загрузки) существующего файла» руководства оператора.

Проверка считается завершенной в случае соответствия состава и последовательности действий оператора при выполнении данной функции указанному выше подразделу руководства оператора.

По результатам проведения проверки представитель заказчика вносит запись в Протокол испытаний - «п. такой-то выполнен».

**Методы проверки выполнения функции редактирования открытого (далее - текущего) файла путем ввода, замены, удаления содержимого файла с применением стандартных устройств ввода**

Проверка выполнения указанной функции выполняется согласно п. «Выполнение функции редактирования текущего файла путем ввода, замены, удаления содержимого файла с применением устройств ввода» руководства оператора.

Проверка считается завершенной в случае соответствия состава и последовательности действий оператора при выполнении данной функции указанному выше подразделу руководства оператора.

По результатам проведения проверки представитель заказчика вносит запись в Протокол испытаний - «п. такой-то выполнен».

**Методы проверки выполнения функции редактирования текущего файла с применением буфера обмена операционной системы**

Проверка выполнения указанной функции выполняется согласно п. «Выполнение функции редактирования текущего файла с применением буфера обмена операционной системы» руководства оператора.

Проверка считается завершенной в случае соответствия состава и последовательности действий оператора при выполнении данной функции указанному выше подразделу руководства оператора.

По результатам проведения проверки представитель заказчика вносит запись в Протокол испытаний - «п. такой-то выполнен».

**Методы проверки выполнения функции сохранения файла с исходным именем**

Проверка выполнения указанной функции выполняется согласно п. «Выполнение функции сохранения файла с исходным именем» руководства оператора.

Проверка считается завершенной в случае соответствия состава и последовательности действий оператора при выполнении данной функции указанному выше подразделу руководства оператора.

По результатам проведения проверки представитель заказчика вносит запись в Протокол испытаний - «п. такой-то выполнен».

**Методы проверки выполнение функции сохранения файла с именем, отличным от исходного**

Проверка выполнения указанной функции выполняется согласно п. «Выполнение функции сохранения файла с именем, отличным от исходного» руководства оператора.

Проверка считается завершенной в случае соответствия состава и последовательности действий оператора при выполнении данной функции указанному выше подразделу руководства оператора.

По результатам проведения проверки представитель заказчика вносит запись в Протокол испытаний - «п. такой-то выполнен».

**Методы проверки выполнения функции отправки содержимого текущего файла электронной почтой с помощью внешней клиентской почтовой программы**

Проверка выполнения указанной функции выполняется согласно п. такому-то руководства оператора.

Проверка считается завершенной в случае соответствия состава и последовательности действий оператора при выполнении данной функции указанному выше подразделу руководства оператора.

По результатам проведения проверки представитель заказчика вносит запись в Протокол испытаний - «п. такой-то выполнен».

**Методы проверки выполнения функции вывода оперативных справок в строковом формате (подсказок)**

Проверка выполнения указанной функции выполняется согласно п. такому-то руководства оператора.

Проверка считается завершенной в случае соответствия состава и последовательности действий оператора при выполнении данной функции указанному выше подразделу руководства оператора.

По результатам проведения проверки представитель заказчика вносит запись в Протокол испытаний - «п. такой-то выполнен».

**Методы проверки выполнения функции интерактивной справочной системы**

Проверка выполнения указанной функции выполняется согласно п. такому-то руководства оператора.

Проверка считается завершенной в случае соответствия состава и последовательности действий оператора при выполнении данной функции указанному выше подразделу руководства оператора.

По результатам проведения проверки представитель заказчика вносит запись в Протокол испытаний - «п. такой-то выполнен».

**Методы проверки выполнение функции отображения названия программы, версии программы, копирайта и комментариев разработчика**

Проверка выполнения указанной функции выполняется согласно п. такому-то руководства оператора.

Проверка считается завершенной в случае соответствия состава и последовательности действий оператора при выполнении данной функции указанному выше подразделу руководства оператора.

По результатам проведения проверки представитель заказчика вносит запись в Протокол испытаний - «п. такой-то выполнен».

Вот и вся Программа и методики испытаний.

**Выводы**

Программа и методика испытаний, разработанные согласно требований ГОСТ 19.301-79 – программный документ, достаточный (в целом) для проведения испытаний программных изделий.

В то же время программа и методики испытаний (компонентов, комплексов средств автоматизации, подсистем, систем) согласно п. 2.14.  РД 50-34.698-90. АВТОМАТИЗИРОВАННЫЕ СИСТЕМЫ ТРЕБОВАНИЯ К СОДЕРЖАНИЮ ДОКУМЕНТОВ – документ, который можно считать «всеобъемлющим». Автор(ы), при разработке программы и методики испытаний, рекомендует обращаться именно к указанному выше документу.

# Метрология

Метрология – это наука об измерениях, методах, средствах обеспечения их единства и способах достижения требуемой точности.

Предметом метрологии является извлечение количественной информации о свойствах объектов и процессов с заданной точностью и достоверностью.

Средства метрологии – это совокупность средств измерений и метрологических стандартов, обеспечивающих их рациональное использование.

Измерение – это нахождение значения физической величины опытным путем с помощью специальных технических средств.

Ранжирование – это распределение величин по возрастающим или убывающим показателям, характеризующим те или иные свойства этой величины.

Метрологическое обеспечение – это установление и применение научных и организационных основ, технических средств, правил и норм, необходимых для достижения единства и требуемой точности измерений.

Метрология (от греч. «метро» — мера, «логос» — учение) — наука об измерениях, методах и средствах обеспечения их единства и требуемой точности.

Современная метрология включает три составляющие: законодательную метрологию, фундаментальную (научную) и практическую (прикладную) метрологию.

Метрология как наука и область практической деятельности возникла в древние времена. Основой системы мер в древнерусской практике послужили древнеегипетские единицы измерений, а они в свою очередь были заимствованы в Древней Греции и Риме. Естественно, что каждая система мер отличалась своими особенностями, связанными не только с эпохой, но и с национальным менталитетом.

Наименования единиц и их размеры соответствовали возможности осуществления измерений «подручными» способами, не прибегая к специальным устройствам. Так, на Руси основными единицами длины были пядь и локоть, причем пядь служила основной древнерусской мерой длины и означала расстояние между концами большого и указательного пальца взрослого человека. Позднее, когда появилась другая единица — аршин — пядь (1/4 аршина) постепенно вышла из употребления.

Мера «локоть» пришла к нам из Вавилона и означала расстояние от сгиба локтя до конца среднего пальца руки (иногда — сжатого кулака или большого пальца).

С XVIII в. в России стали применяться дюйм, заимствованный из Англии (назывался он «палец»), а также английский фут. Особой русской мерой была сажень, равная трем локтям (около 152 см), и косая сажень (около 248 см).

Указом Петра I русские меры длины были согласованы с английскими, и это по существу — первая ступень гармонизации российской метрологии с европейской.

Метрическая система мер была введена во Франции в 1840 г. Значимость ее принятия в России подчеркнул Д.И. Менделеев, предсказав большую роль всеобщего распространения метрической системы как средства содействия «будущему желанному сближению народов».

С развитием науки и техники требовались новые измерения и новые единицы измерения, что в свою очередь стимулировало совершенствование фундаментальной и прикладной метрологии.

Первоначально прототип единиц измерения искали в природе, исследуя макрообъекты и их движение. Так, секундой стали считать часть периода обращения Земли вокруг оси. Постепенно поиски переместились на атомный и внутриатомный уровень. В результате уточнялись «старые» единицы (меры) и появились новые. Так, в 1983 г. было принято новое определение метра: это длина пути, проходимого светом в вакууме за 1/299792458 долю секунды. Это стало возможным после того, как скорость света в вакууме (299792458 м/с) метрологи приняли в качестве физической константы. Интересно отметить, что теперь с точки зрения метрологических правил метр зависит от секунды.

В 1988 г. на международном уровне были приняты новые константы в области измерений электрических единиц и величин, а в 1989 г. принята новая Международная практическая температурная шкала МТШ-90.

На этих нескольких примерах видно, что метрология как наука динамично развивается, что, естественно, способствует совершенствованию практики измерений во всех других научных и прикладных областях.

Качеством и точностью измерений определяется возможность разработки принципиально новых приборов, измерительных устройств для любой сферы техники, что говорит в пользу опережающих темпов развития науки и техники измерений, т.е. метрологии. Вместе с развитием фундаментальной и практической метрологии происходило становление законодательной метрологии.

Законодательная метрология — это раздел метрологии, включающий комплексы взаимосвязанных и взаимообусловленных общих правил, а также другие вопросы, нуждающиеся в регламентации и контроле со стороны государства, направленные на обеспечение единства измерений и единообразия средств измерений.

Законодательная метрология служит средством государственного регулирования метрологической деятельности посредством законов и законодательных положений, которые вводятся в практику через Государственную метрологическую службу и метрологические службы государственных органов управления и юридических лиц. К области законодательной метрологии относятся испытания и утверждение типа средств измерений и их поверка и калибровка, сертификация средств измерений, государственный метрологический контроль и надзор за средствами измерений.

Метрологические правила и нормы законодательной метрологии гармонизованы с рекомендациями и документами соответствующих международных организаций. Тем самым законодательная метрология способствует развитию международных экономических и торговых связей и содействует взаимопониманию в международном метрологическом сотрудничестве.

Рассмотрим содержание основных понятий фундаментальной и практической метрологии.

Измерения как основной объект метрологии связаны как с физическими величинами, так и с величинами, относящимися к другим наукам (математике, психологии, медицине, общественным наукам и др.). Далее будут рассматриваться понятия, относящиеся к физическим величинам.

Физической величиной называют одно из свойств физического объекта (явления, процесса), которое является общим в качественном отношении для многих физических объектов, отличаясь при этом количественным значением. Так, свойство «прочность» в качественном отношении характеризует такие материалы, как сталь, дерево, ткань, стекло и многие другие, в то время как степень (количественное значение) прочности — величина для каждого из них совершенно разная.

Измерением называют совокупность операций, выполняемых с помощью технического средства, хранящего единицу величины и позволяющего сопоставить с нею измеряемую величину. Полученное значение величины и есть результат измерений. Интересно отметить соответствие в целом этой современной трактовки с толкованием данного термина философом П.А. Флоренским, которое вошло в «Техническую энциклопедию» издания 1931 г.: «Измерение — основной познавательный процесс науки и техники, посредством которого неизвестная величина количественно сравнивается с другою, однородною с ней и считаемою известной».

Одна из главных задач метрологии — обеспечение единства измерений — может быть решена при соблюдении двух условий, которые можно назвать основополагающими:

выражение результатов измерений в единых узаконенных единицах;

установление допустимых ошибок (погрешностей) результатов измерений и пределов, за которые они не должны выходить при заданной вероятности.

Погрешностью называют отклонение результата измерений от действительного (истинного) значения измеряемой величины. При этом следует иметь в виду, что истинное значение физической величины считается неизвестным и применяется в теоретических исследованиях; действительное значение физической величины устанавливается экспериментальным путем в предположении, что результат эксперимента (измерения) в максимальной степени приближается к истинному значению. Погрешности измерений приводятся обычно в технической документации на средства измерений или в нормативных документах. Правда, если учесть, что погрешность зависит еще и от условий, в которых проводится само измерение, от экспериментальной ошибки методики и субъективных особенностей человека в случаях, где он непосредственно участвует в измерениях, то можно говорить о нескольких составляющих погрешности измерений либо о суммарной погрешности.

Единство измерений, однако, не может быть обеспечено лишь совпадением погрешностей. Требуется еще и достоверность измерений, которая говорит о том, что погрешность не выходит за пределы отклонений, заданных в соответствии с поставленной целью измерений. Есть еще и понятие точности измерений, которое характеризует степень приближения погрешности измерений к нулю, т.е. полученного при измерении значения к истинному значению измеряемой величины.

Обобщает все эти положения современное определение понятия единство измерений — состояние измерений, при котором их результаты выражены в узаконенных единицах, а погрешности известны с заданной вероятностью и не выходят за установленные пределы.

Измерения различают по способу получения информации, по характеру изменений измеряемой величины в процессе измерений, по количеству измерительной информации, по отношению к основным единицам.

По способу получения информации измерения разделяют на:

* прямые,
* косвенные,
* совокупные,
* совместные.

Прямые измерения — это непосредственное сравнение физической величины с ее мерой. Например, при определении длины предмета линейкой происходит сравнение искомой величины (количественного выражения значения длины) с мерой, т.е. линейкой.

Косвенные измерения отличаются от прямых тем, что искомое значение величины устанавливают по результатам прямых измерений таких величин, которые связаны с искомой определенной зависимостью. Так, если измерить силу тока амперметром, а напряжение вольтметром, то по известной функциональной взаимосвязи всех трех величин можно рассчитать мощность электрической цепи.

Совокупные измерения сопряжены с решением системы уравнений, составляемых по результатам одновременных измерений нескольких однородных величин. Решение системы уравнений дает возможность вычислить искомую величину. (Пример: нахождение масс отдельных гирь набора по известной массе одной из них и по результатам прямых сравнений масс различных сочетаний гирь.)

Совместные измерения — это измерения двух или более неоднородных физических величин для определения зависимости между ними. (Пример: определение температурного коэффициента измерительного резистора по данным прямых измерений его сопротивления при различных температурах.)

Совокупные и совместные измерения часто применяют в измерениях различных параметров и характеристик в области электротехники.

По характеру изменения измеряемой величины в процессе измерений бывают статистические, динамические и статические измерения.

Статистические измерения связаны с определением характеристик случайных процессов, звуковых сигналов, уровня шумов и т.д.

Статические измерения имеют место тогда, когда измеряемая величина практически постоянна.

Динамические измерения связаны с такими величинами, которые в процессе измерений претерпевают те или иные изменения.

Статические и динамические измерения в идеальном виде на практике редки.

По количеству измерительной информации различают однократные и многократные измерения.

Однократные измерения — это одно измерение одной величины, т.е. число измерений равно числу измеряемых величин. Практическое применение такого вида измерений всегда сопряжено с большими погрешностями, поэтому следует проводить не менее трех однократных измерений и находить конечный результат как среднее арифметическое значение.

Многократные измерения характеризуются превышением числа измерений количества измеряемых величин. Обычно минимальное число измерений в данном случае больше трех. Преимущество многократных измерений — в значительном снижении влияний случайных факторов на погрешность измерения.

По отношению к основным единицам измерения делят на абсолютные и относительные.

Абсолютными измерениями называют такие, при которых используются прямое измерение одной (иногда нескольких) основной величины и физическая константа. Так, в известной формуле Эйнштейна Е=mс2 масса (m) — основная физическая величина, которая может быть измерена прямым путем (взвешиванием), а скорость света (с) — физическая константа.

Относительные измерения базируются на установлении отношения измеряемой величины к однородной, применяемой в качестве единицы. Естественно, что искомое значение зависит от используемой единицы измерений.

С измерениями связаны такие понятия, как «шкала измерений», «принцип измерений», «метод измерений».

Шкала измерений — это упорядоченная совокупность значений физической величины, которая служит основой для ее измерения. Поясним это понятие на примере температурных шкал.

В шкале Цельсия за начало отсчета принята температура таяния льда, а в качестве основного интервала (опорной точки) — температура кипения воды. Одна сотая часть этого интервала является единицей температуры (градус Цельсия). В температурной шкале Фаренгейта за начало отсчета принята температура таяния смеси льда и нашатырного спирта (либо поваренной соли), а в качестве опорной точки взята нормальная температура тела здорового человека. За единицу температуры (градус Фаренгейта) принята одна девяносто шестая часть основного интервала. По этой шкале температура таяния льда равна + 32°F, а температура кипения воды + 212°F. Таким образом, если по шкале Цельсия разность между температурой кипения воды и таяния льда составляет 100°С, то по Фаренгейту она равна 180°F. На этом примере мы видим роль принятой шкалы как в количественном значении измеряемой величины, так и в аспекте обеспечения единства измерений. В данном случае требуется находить отношение размеров единиц, чтобы можно было сравнить результаты измерений, т.е. t°F/t°C.

В метрологической практике известны несколько разновидностей шкал: шкала наименований, шкала порядка, шкала интервалов, шкала отношений и др.

Шкала наименований — это своего рода качественная, а не количественная шкала, она не содержит нуля и единиц измерений. Примером может служить атлас цветов (шкала цветов). Процесс измерения заключается в визуальном сравнении окрашенного предмета с образцами цветов (эталонными образцами атласа цветов). Поскольку каждый цвет имеет немало вариантов, такое сравнение под силу опытному эксперту, который обладает не только практическим опытом, но и соответствующими особыми характеристиками зрительных возможностей.

Шкала порядка характеризует значение измеряемой величины в баллах (шкала землетрясений, силы ветра, твердости физических тел и т.п.).

Шкала интервалов (разностей) имеет условные нулевые значения, а интервалы устанавливаются по согласованию. Такими шкалами являются шкала времени, шкала длины.

Шкала отношений имеет естественное нулевое значение, а единица измерений устанавливается по согласованию. Например, шкала массы (обычно мы говорим «веса»), начинаясь от нуля, может быть градуирована по-разному в зависимости от требуемой точности взвешивания (сравните бытовые и аналитические весы).

**Вопросы для самопроверки**

1. В чем заключаются основные задачи метрологии?
2. На какие виды подразделяется метрология?
3. Что понимают под измерением?
4. Какие измерения называют прямыми, косвенными, совокупными, совместными?
5. Какие Вы знаете средства измерений?
6. Что такое поверка средств измерений?
7. Дайте характеристику существующих методов измерений?
8. Что понимают под метрологической службой?
9. Кто был одним и основоположников метрологии в России?
10. Какова роль метрологии в повышении качества ПС?
11. Основы метрологического обеспечения программных средств.
12. История развития метрологии.
13. Метрология в зарубежных странах и международные метрологические организации.
14. Государственная метрологическая служба в РФ.
15. Закон « Об обеспечении единства измерений».

# Качество программных средств.

Как проверить, что требования определены достаточно полно и описывают все, что ожидается от будущей программной системы? Это можно сделать, проследив, все ли необходимые аспекты качества ПО отражены в них. Именно понятие качественного ПО соответствует представлению о том, что программа достаточно успешно справляется со всеми возложенными на нее задачами и не приносит проблем ни конечным пользователям, ни их начальству, ни службе поддержки, ни специалистам по продажам. Да и самим разработчикам создание качественной программы приносит гораздо больше удовольствия.

Если попросить группу людей высказать свое мнение по поводу того, что такое качественное ПО, можно получить следующие варианты ответов:

* Его легко использовать.
* Оно демонстрирует хорошую производительность.
* В нем нет ошибок.
* Оно не портит пользовательские данные при сбоях.
* Его можно использовать на разных платформах.
* Оно может работать 24 часа в сутки и 7 дней в неделю.
* В него легко добавлять новые возможности.
* Оно удовлетворяет потребности пользователей.
* Оно хорошо документировано.

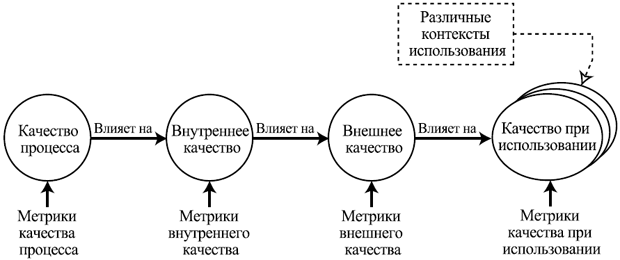
Все это действительно имеет непосредственное отношение к качеству ПО. Но эти ответы выделяют характеристики, важные для конкретного пользователя, разработчика или группы таких лиц. Для того чтобы удовлетворить потребности всех сторон (конечных пользователей, заказчиков, разработчиков, администраторов систем, в которых оно будет работать, регулирующих организаций и пр.), для достижения прочного положения разрабатываемого ПО на рынке и повышения потенциала его развития необходим учет всей совокупности характеристик ПО, важных для всех заинтересованных лиц.

Приведенные выше ответы показывают, что качество ПО может быть описано большим набором разнородных характеристик. Такой подход к описанию сложных понятий называется холистическим (от греческого слова ολοσ, целое). Он не дает единой концептуальной основы для рассмотрения затрагиваемых вопросов, какую дает целостная система представлений (например, Ньтоновская механика в физике или классическая теория вычислимости на основе машин Тьюринга), но позволяет, по крайней мере, не упустить ничего существенного.

Качество программного обеспечения определяется в стандарте ISO 9126 [[1]](http://www.intuit.ru/studies/courses/64/64/literature#literature.5.1) как вся совокупность его характеристик, относящихся к возможности удовлетворять высказанные или подразумеваемые потребности всех заинтересованных лиц.

Тот же стандарт ISO 9126 дает следующее представление качества.

Различаются понятия внутреннего качества, связанного с характеристиками ПО самого по себе, без учета его поведения; внешнего качества, характеризующего ПО с точки зрения его поведения; и качества ПО при использовании в различных контекстах — того качества, которое ощущается пользователями при конкретных сценариях работы ПО. Для всех этих аспектов качества введены метрики, позволяющие оценить их. Кроме того, для создания добротного ПО существенно качество технологических процессов его разработки. Взаимоотношения между этими аспектами качества по схеме, принятой ISO 9126, показано на рис.

[](http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/145/objects/5/files/5_1.gif)

**рис.** Основные аспекты качества ПО по ISO 9126

Общие принципы обеспечения качества процессов производства во всех отраслях экономики регулируются набором стандартов ISO 9000. Наиболее важные для разработки ПО стандарты в его составе следующие:

* **ISO 9000:2000 Quality management systems — Fundamentals and vocabulary**.

Системы управления качеством — Основы и словарь. (Аналог — ГОСТ Р-2001).

* **ISO 9001:2000 Quality management systems — Requirements. Models for quality assurance in design, development, production, installation, and servicing**.

Системы управления качеством — Требования. Модели для обеспечения качества при проектировании, разработке, коммерциализации, установке и обслуживании.

Определяет общие правила обеспечения качества результатов во всех процессах жизненного цикла. (Аналог — ГОСТ Р-2001).

Этот стандарт выделяет следующие процессы:

* Управление качеством.
* Управление ресурсами.
* Развитие системы управления.
* Исследования рынка.
* Проектирование продуктов.
* Приобретения.
* Производство.
* Оказание услуг.
* Защита продуктов.
* Оценка потребностей заказчиков.
* Поддержка коммуникаций с заказчиками.
* Поддержка внутренних коммуникаций.
* Управление документацией.
* Ведение записей о деятельности.
* Планирование.
* Обучение персонала.
* Внутренние аудиты.
* Оценки управления.
* Мониторинг и измерения.
* Управление несоответствиями.
* Постоянное совершенствование.
* Управление и развитие системы в целом.

Для каждого процесса требуется иметь планы развития процесса, состоящие как минимум из следующих разделов:

* Проектирование процесса.
* Документирование процесса.
* Реализация процесса.
* Поддержка процесса.
* Мониторинг процесса.
* Управление процессом.
* Усовершенствование процесса.

Помимо поддержки и развития системы процессов, нацеленных на удовлетворение нужд заказчиков и пользователей, ISO 9001 требует:

* Определить, документировать и развивать собственную систему качества на основе измеримых показателей.
* Использовать эту систему качества как средство управления процессами, нацеливая их на большее удовлетворение нужд заказчиков, планируя и постоянно отслеживая качество результатов всех видов деятельности, в том числе и самого управления.
* Обеспечить использование качественных ресурсов, качественного (компетентного, профессионального) персонала, качественной инфраструктуры и качественного окружения.
* Постоянно контролировать соблюдение требований к качеству на практике, во всех процессах проектирования, производства, предоставления услуг и при приобретениях.
* Предусмотреть процесс устранения дефектов, определить и контролировать качество результатов этого процесса.

Ранее использовавшиеся стандарты ISO 9002:1994 Quality systems — Model for quality assurance in production, installation and servicing и ISO 9003:1994 Quality systems — Model for quality assurance in final inspection and test в 2000 году были заменены соответствующими им частями ISO 9001.

* **ISO 9004:2000 Quality management systems — Guidelines for performance improvements**.

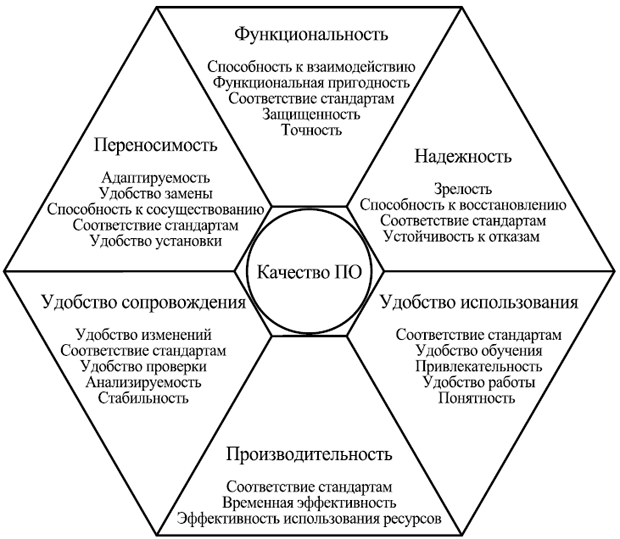
Системы управления качеством. Руководство по улучшению деятельности. (Аналог — ГОСТ Р-2001).

* **ISO/IEC 90003:2004 Software engineering — Guidelines for the application of ISO 9001:2000 to computer software**.

Руководящие положения по применению стандарта ISO 9001 при разработке, поставке и обслуживании программного обеспечения.

Этот стандарт конкретизирует положения ISO 9001 для разработки программных систем, с упором на обеспечение качества при процессе проектирования. Он также определяет некоторый набор техник и процедур, которые рекомендуется применять для контроля и обеспечения качества разрабатываемых программ.

Стандарт ISO 9126 предлагает использовать для описания внутреннего и внешнего качества ПО многоуровневую модель. На верхнем уровне выделено 6 основных характеристик качества ПО. Каждая характеристика описывается при помощи нескольких входящих в нее атрибутов. Для каждого атрибута определяется набор метрик, позволяющих его оценить. Множество характеристик и атрибутов качества согласно ISO 9126 показано на рис

[](http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/145/objects/5/files/5_2.gif)

**Рис.** Характеристики и атрибуты качества ПО по ISO 9126

Ниже приведены определения этих характеристик и атрибутов по стандарту ISO 9126:2001:

Функциональность (functionality) - способность ПО в определенных условиях решать задачи, нужные пользователям. Определяет, что именно делает ПО, какие задачи оно решает.

* + **Функциональная пригодность (suitability)**.

Способность решать нужный набор задач.

* + **Точность (accuracy)**.

Способность выдавать нужные результаты.

* + **Способность к взаимодействию (interoperability)**.

Способность взаимодействовать с нужным набором других систем.

* + **Соответствие стандартам и правилам (compliance)**.

Соответствие ПО имеющимся индустриальным стандартам, нормативным и законодательным актам, другим регулирующим нормам.

* + **Защищенность (security)**.

Способность предотвращать неавторизированный, т.е. без указания лица, пытающегося его осуществить, и неразрешенный доступ к данным и программам.

* Надежность (reliability).

Способность ПО поддерживать определенную работоспособность в заданных условиях.

* + **Зрелость, завершенность (maturity)**.

Величина, обратная частоте отказов ПО. Обычно измеряется средним временем работы без сбоев и величиной, обратной вероятности возникновения отказа за данный период времени.

* + **Устойчивость к отказам (fault tolerance)**.

Способность поддерживать заданный уровень работоспособности при отказах и нарушениях правил взаимодействия с окружением.

* + **Способность к восстановлению (recoverability)**.

Способность восстанавливать определенный уровень работоспособности и целостность данных после отказа, необходимые для этого время и ресурсы.

* + **Соответствие стандартам** надежности **(reliability compliance)**.

Этот атрибут добавлен в 2001 году.

* Удобство использования **(usability) или практичность**.

Способность ПО быть удобным в обучении и использовании, а также привлекательным для пользователей.

* + **Понятность (understandability)**.

Показатель, обратный к усилиям, которые затрачиваются пользователями на восприятие основных понятий ПО и осознание их применимости для решения своих задач.

* + **Удобство обучения (learnability)**.

Показатель, обратный усилиям, затрачиваемым пользователями на обучение работе с ПО.

* + **Удобство работы (operability)**.

Показатель, обратный усилиям, предпринимаемым пользователями для решения своих задач с помощью ПО.

* + **Привлекательность (attractiveness)**.

Способность ПО быть привлекательным для пользователей. Этот атрибут добавлен в 2001 году.

* + **Соответствие стандартам** удобства использования **(usability compliance)**.

Этот атрибут добавлен в 2001 году.

* Производительность **(efficiency) или эффективность**.

Способность ПО при заданных условиях обеспечивать необходимую работоспособность по отношению к выделяемым для этого ресурсам. Можно определить ее и как отношение получаемых с помощью ПО результатов к затрачиваемым на это ресурсам всех типов.

* + **Временная эффективность (time behaviour)**.

Способность ПО выдавать ожидаемые результаты, а также обеспечивать передачу необходимого объема данных за отведенное время.

* + **Эффективность использования ресурсов (resource utilisation)**.

Способность решать нужные задачи с использованием определенных объемов ресурсов определенных видов. Имеются в виду такие ресурсы, как оперативная и долговременная память, сетевые соединения, устройства ввода и вывода и пр.

* + **Соответствие стандартам** производительности **(efficiency compliance)**.

Этот атрибут добавлен в 2001 году.

* Удобство сопровождения (maintainability).

Удобство проведения всех видов деятельности, связанных с сопровождение программ.

* + **Анализируемость (analyzability) или удобство проведения анализа**.

Удобство проведения анализа ошибок, дефектов и недостатков, а также удобство анализа необходимости изменений и их возможных последствий.

* + **Удобство внесения изменений (changeability)**.

Показатель, обратный трудозатратам на выполнение необходимых изменений.

* + **Стабильность (stability)**.

Показатель, обратный риску возникновения неожиданных эффектов при внесении необходимых изменений.

* + **Удобство проверки (testability)**.

Показатель, обратный трудозатратам на проведение тестирования и других видов проверки того, что внесенные изменения привели к нужным результатам.

* + **Соответствие стандартам** удобства сопровождения **(maintainability compliance)**.

Этот атрибут добавлен в 2001 году.

* Переносимость (portability).

Способность ПО сохранять работоспособность при переносе из одного окружения в другое, включая организационные, аппаратные и программные аспекты окружения.

Иногда эта характеристика называется в русскоязычной литературе мобильностью. Однако термин "мобильность" стоит зарезервировать для перевода "mobility" — способности ПО и компьютерной системы в целом сохранять работоспособность при ее физическом перемещении в пространстве.

* + **Адаптируемость (adaptability)**.

Способность ПО приспосабливаться различным окружениям без проведения для этого действий, помимо заранее предусмотренных.

* + **Удобство установки (installability)**.

Способность ПО быть установленным или развернутым в определенном окружении.

* + **Способность к сосуществованию (coexistence)**.

Способность ПО сосуществовать с другими программами в общем окружении, деля с ними ресурсы.

* + **Удобство замены (replaceability) другого ПО данным**.

Возможность применения данного ПО вместо других программных систем для решения тех же задач в определенном окружении.

* + **Соответствие стандартам** переносимости **(portability compliance)**.

Этот атрибут добавлен в 2001 году.

Перечисленные атрибуты относятся к внутреннему и внешнему качеству ПО согласно ISO 9126. Для описания качества ПО при использовании стандарт ISO 9126-4 предлагает другой, более узкий набор характеристик.

* **Эффективность (effectiveness)**.

Способность ПО предоставлять пользователям возможность решать их задачи с необходимой точностью при использовании в заданном контексте.

* **Продуктивность (productivity)**.

Способность ПО предоставлять пользователям определенные результаты в рамках ожидаемых затрат ресурсов.

* **Безопасность (safety)**.

Способность ПО обеспечивать необходимо низкий уровень риска нанесения ущерба жизни и здоровью людей, бизнесу, собственности или окружающей среде.

* **Удовлетворение пользователей (satisfaction)**.

Способность ПО приносить удовлетворение пользователям при использовании в заданном контексте.

Помимо перечисленных характеристик и атрибутов качества, стандарт ISO 9126:2001 определяет наборы метрик для оценки каждого атрибута. Приведем следующие примеры таких метрик.

* **Полнота реализации функций** — процент реализованных функций по отношению к перечисленным в требованиях. Используется для измерения функциональной пригодности.
* **Корректность реализации функций** — правильность их реализации по отношению к требованиям. Используется для измерения функциональной пригодности.
* **Отношение числа обнаруженных дефектов к прогнозируемому**. Используется для определения зрелости.
* **Отношение числа проведенных тестов к общему их числу**. Используется для определения зрелости.
* **Отношение числа доступных проектных документов к указанному в их списке**. Используется для измерения удобства проведения анализа.
* **Наглядность и полнота документации**. Используется для оценки понятности.

Перечисленные характеристики и атрибуты качества ПО позволяют систематически описывать требования к нему, определяя, какие свойства ПО по данной характеристике хотят видеть заинтересованные стороны. Таким образом, требования должны определять следующее.

* Что ПО должно делать, например:
  + позволять клиенту оформить заказы и обеспечить их доставку;
  + обеспечивать контроль качества строительства и отслеживать проблемные места;
  + поддерживать нужные характеристики автоматизированного процесса производства, предотвращая аварии и оптимальным образом используя имеющиеся ресурсы.
* Насколько оно должно быть надежно, например:
  + работать 7 дней в неделю и 24 часа в сутки;
  + допускается неработоспособность в течение не более 3 часов в год;
  + никакие введенные пользователями данные при отказе не должны теряться.
* Насколько им должно быть удобно пользоваться, например:
  + покупатель должен, зная название товара и имея средние навыки работы в Интернет, находить нужный ему товар за не более чем 2 минуты;
  + инженер по специальности "строительство мостов" должен в течение одного дня уметь разобраться в 80% функций системы.
* Насколько оно должно быть эффективно, например:
  + поддерживать обслуживание до 10000 запросов в секунду;
  + время отклика на запрос при максимальной загрузке не должно превышать 3 с;
  + время реакции на изменение параметров процесса производства не должно превышать 0.1 с;
  + на обработку одного запроса не должно тратиться более 1 MB оперативной памяти.
* Насколько удобно должно быть его сопровождение, например:
  + добавление в систему нового вида запросов не должно требовать более 3 человеко-дней;
  + добавление поддержки нового этапа процесса производства не должно стоить более $20000.
* Насколько оно должно быть переносимо, например:
  + ПО должно работать на операционных системах Linux, Windows XP и MacOS X;
  + ПО должно работать с документами в форматах MS Word 97 и HTML;
  + ПО должно сохранять файлы отчетов в форматах MS Word 2000, MS Excel 2000, HTML, RTF и в виде обычного текста;
  + ПО должно сопрягаться с существующей системой записи данных о заказах.

Приведенные атрибуты качества закреплены в стандартах, но это не значит, что они вполне исчерпывают понятие качества ПО. Так, в стандарте ISO 9126 отсутствуют характеристики, связанные с **мобильностью ПО (mobility)**, т.е. способностью программы работать при физических перемещениях машины, на которой она работает. Вместо надежности многие исследователи предпочитают рассматривать более общее понятие **добротности (dependability)**, описывающее способность ПО поддерживать определенные показатели качества по основным характеристикам ( функциональности, производительности, удобству использования ) с заданными вероятностями выхода за их рамки и определенным максимальным ущербом от возможных нарушений. Кроме того, активно исследуются понятия удобства использования, безопасности и защищенности ПО, — они кажутся большинству специалистов гораздо более сложными, чем это описывается данным стандартом.

**Методы контроля качества**

Как контролировать качество системы? Как точно узнать, что программа делает именно то, что нужно, и ничего другого? Как определить, что она достаточно надежна, переносима, удобна в использовании? Ответы на эти вопросы можно получить с помощью процессов верификации и валидации.

* **Верификация** обозначает проверку того, что ПО разработано в соответствии со всеми требованиями к нему, или что результаты очередного этапа разработки соответствуют ограничениям, сформулированным на предшествующих этапах.
* **Валидация** — это проверка того, что сам продукт правилен, т.е. подтверждение того, что он действительно удовлетворяет потребностям и ожиданиям пользователей, заказчиков и других заинтересованных сторон.

Эффективность верификации и валидации, как и эффективность разработки ПО в целом, зависит от полноты и корректности формулировки требований к программному продукту.

Основой любой системы обеспечения качества являются методы его обеспечения и контроля. Методы обеспечения качества представляют собой техники, гарантирующие достижение определенных показателей качества при их применении. Мы будем рассматривать подобные методы на протяжении всего курса.

Методы контроля качества позволяют убедиться, что определенные характеристики качества ПО достигнуты. Сами по себе они не могут помочь их достижению, они лишь помогают определить, удалось ли получить в результате то, что хотелось, или нет, а также найти ошибки, дефекты и отклонения от требований. Методы контроля качества ПО можно классифицировать следующим образом:

* Методы и техники, связанные с выяснением свойств ПО во время его работы.

Это, прежде всего, все виды тестирования, а также **профилирование** и измерение количественных показателей качества, которые можно определить по результатам работы ПО — эффективности по времени и другим ресурсам, надежности, доступности и пр.

* Методы и техники определения показателей качества на основе симуляции работы ПО с помощью моделей разного рода.

К этому виду относятся **проверка на моделях (model checking)**, а также **прототипирование (макетирование)**, используемое для оценки качества принимаемых решений.

* Методы и техники, нацеленные на выявление нарушений формализованных правил построения исходного кода ПО, проектных моделей и документации.

К методам такого рода относится **инспектирование кода**, заключающееся в целенаправленном поиске определенных дефектов и нарушений требований в коде на основе набора шаблонов, автоматизированные методы поиска ошибок в коде, не основанные на его выполнении, методы проверки документации на согласованность и соответствие стандартам.

* Методы и техники обычного или формализованного анализа проектной документации и исходного кода для выявления их свойств.

К этой группе относятся многочисленные методы анализа архитектуры ПО, о которых пойдет речь в следующей лекции, методы формального доказательства свойств ПО и формального анализа эффективности применяемых алгоритмов.

Далее мы несколько подробнее рассмотрим тестирование и проверку на моделях как примеры методов контроля качества.

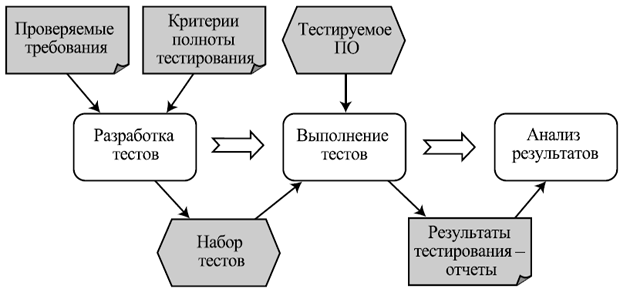
**Тестирование**

Тестирование — это проверка соответствия ПО требованиям, осуществляемая с помощью наблюдения за его работой в специальных, искусственно построенных ситуациях. Такого рода ситуации называют тестовыми или просто тестами.

Тестирование — конечная процедура. Набор ситуаций, в которых будет проверяться тестируемое ПО, всегда конечен. Более того, он должен быть настолько мал, чтобы тестирование можно было провести в рамках проекта разработки ПО, не слишком увеличивая его бюджет и сроки. Это означает, что при тестировании всегда проверяется очень небольшая доля всех возможных ситуаций. По этому поводу Дейкстра (Dijkstra) заметил, что тестирование позволяет точно определить, что в программе есть ошибка, но не позволяет утверждать, что там нет ошибок.

Тем не менее, тестирование может использоваться для достаточно уверенного вынесения оценок о качестве ПО. Для этого необходимо иметь критерии полноты тестирования, описывающие важность различных ситуаций для оценки качества, а также эквивалентности и зависимости между ними. Этот критерий может утверждать, что все равно в какой из ситуаций, A или B, проверять правильность работы ПО, или, если программа правильно работает в ситуации A, то, скорее всего, в ситуации B все тоже будет правильно. Часто критерий полноты тестирования задается при помощи определения эквивалентности ситуаций, дающей конечный набор классов ситуаций. В этом случае считают, что все равно, какую из ситуаций одного класса использовать в качестве теста. Такой критерий называют критерием тестового покрытия, а процент классов эквивалентности ситуаций, случившихся во вр емя тестирования, — достигнутым тестовым покрытием.

Таким образом, основные задачи тестирования: построить такой набор ситуаций, который был бы достаточно представителен и позволял бы завершить тестирование с достаточной степенью уверенности в правильности ПО вообще, и убедиться, что в конкретной ситуации ПО работает правильно, в соответствии с требованиями.

[](http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/145/objects/5/files/5_3.gif)

**Рис.** Схема процесса тестирования

Тестирование — наиболее широко применяемый метод контроля качества. Для оценки многих атрибутов качества не существует других эффективных способов, кроме тестирования.

Организация тестирования ПО регулируется следующими стандартами:

* IEEE 829-1998 Standard for Software Test Documentation.

Описывает виды документов, служащих для подготовки тестов.

* IEEE 1008-1987 (R1993, R2002) Standard for Software Unit Testing.

Описывает организацию модульного тестирования (см. ниже).

* ISO/IEC 12119:1994 (аналог AS/NZS 4366:1996 и ГОСТ Р-2000, также принят IEEE под номером IEEE 1465-1998) Information Technology. Software packages — Quality requirements and testing.

Описывает требования к процедурам тестирования программных систем.

Тестировать можно соблюдение любых требований, соответствие которым выявляется во время работы ПО. Из характеристик качества по ISO 9126 этим свойством не обладают только атрибуты удобства сопровождения. Поэтому выделяют виды тестирования, связанные с проверкой определенных характеристик и атрибутов качества — тестирование функциональности, надежности, удобства использования, переносимости и производительности, а также тестирование защищенности, функциональной пригодности и пр. Кроме того, особо выделяют нагрузочное или стрессовое тестирование, проверяющее работоспособность ПО и показатели его производительности в условиях повышенных нагрузок — при большом количестве пользователей, интенсивном обмене данными с другими системами, большом объеме передаваемых или используемых данных и пр.

На основе исходных данных, используемых для построения тестов, тестирование делят на следующие виды:

* Тестирование черного ящика, нацеленное на проверку требований. Тесты для него и критерий полноты тестирования строятся на основе требований и ограничений, четко зафиксированных в спецификациях, стандартах, внутренних нормативных документах. Часто такое тестирование называется тестированием на соответствие (conformance testing). Частным случаем его является функциональное тестирование — тесты для него, а также используемые критерии полноты проведенного тестирования определяют на основе требований к функциональности.

Еще одним примером тестирования на соответствие является аттестационное или квалификационное тестирование, по результатам которого программная система получает (или не получает) официальный документ, подтверждающий ее соответствие определенным требованиям и стандартам.

* Тестирование белого ящика, оно же структурное тестирование — тесты создаются на основе знаний о структуре самой системы и о том, как она работает. Критерии полноты основаны на проценте элементов кода, которые отработали в ходе выполнения тестов. Для оценки степени соответствия требованиям могут привлекаться дополнительные знания о связи требований с определенными ограничениями на значения внутренних данных системы (например, на значения параметров вызовов, результатов и локальных переменных).
* Тестирование, нацеленное на определенные ошибки. Тесты для такого тестирования строятся так, чтобы гарантированно выявлять определенные виды ошибок. Полнота тестирования определяется на основе количества проверенных ситуаций по отношению к общему числу ситуаций, которые мы пытались проверить. К этому виду относится, например, тестирование на отказ (smoke testing), в ходе которого просто пытаются вывести систему из строя, давая ей на вход как обычные данные, так и некорректные, с нарочно внесенными ошибками.

Другим примером служит метод оценки полноты тестирования при помощи набора мутантов — программ, совпадающих с тестируемой всюду, кроме нескольких мест, где специально внесены некоторые ошибки. Чем больше мутантов не проходит успешно через данный набор тестов, тем полнее и качественнее проводимое с его помощью тестирование.

Еще одна классификация видов тестирования основана на том уровне, на который оно нацелено. Эти же разновидности тестирования можно связать с фазой жизненного цикла, на которой они выполняются.

* Модульное тестирование (unit testing) предназначено для проверки правильности отдельных модулей, вне зависимости от их окружения. При этом проверяется, что если модуль получает на вход данные, удовлетворяющие определенным критериям корректности, то и результаты его корректны. Для описания критериев корректности входных и выходных данных часто используют программные контракты — предусловия, описывающие для каждой операции, на каких входных данных она предназначена работать, постусловия, описывающие для каждой операции, как должны соотноситься входные данные с возвращаемыми ею результатами, и инварианты, определяющие критерии целостности внутренних данных модуля.

Модульное тестирование является важной составной частью отладочного тестирования, выполняемого разработчиками для отладки написанного ими кода.

* Интеграционное тестирование (integration testing) предназначено для проверки правильности взаимодействия модулей некоторого набора друг с другом. При этом проверяется, что в ходе совместной работы модули обмениваются данными и вызовами операций, не нарушая взаимных ограничений на такое взаимодействие, например, предусловий вызываемых операций. Интеграционное тестирование также используется при отладке, но на более позднем этапе разработки.
* Системное тестирование (system testing) предназначено для проверки правильности работы системы в целом, ее способности правильно решать поставленные пользователями задачи в различных ситуациях.

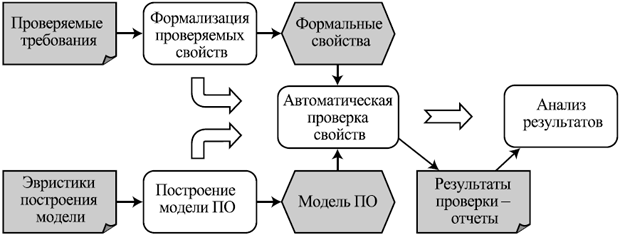
Системное тестирование выполняется через внешние интерфейсы ПО и тесно связано с тестированием пользовательского интерфейса (или через пользовательский интерфейс), проводимым при помощи имитации действий пользователей над элементами этого интерфейса. Частными случаями этого вида тестирования являются тестирование графического пользовательского интерфейса (Graphical User Interface, GUI) и пользовательского интерфейса Web-приложений (WebUI).

Если интеграционное и модульное тестирование чаще всего проводят, воздействуя на компоненты системы при помощи операций предоставляемого ими программного интерфейса (Application Programming Interface, API), то на системном уровне без использования пользовательского интерфейса не обойтись, хотя тестирование через API в этом случае также вполне возможно.

Основной недостаток тестирования состоит в том, что проводить его можно, только когда проверяемый элемент программы уже разработан. Снизить влияние этого ограничения можно, подготавливая тесты (а это — наиболее трудоемкая часть тестирования ) на основе требований заранее, когда исходного кода еще нет. Подход опережающей разработки тестов с успехом используется, например, в рамках XP.

**Проверка на моделях**

Проверка свойств на моделях (model checking) — проверка соответствия ПО требованиям при помощи формализации проверяемых свойств, построения формальных моделей проверяемого ПО (чаще всего в виде автоматов различных видов) и автоматической проверки выполнения этих свойств на построенных моделях. Проверка свойств на моделях позволяет проверять достаточно сложные свойства автоматически, при минимальном участии человека. Однако она оставляет открытым вопрос о том, насколько выявленные свойства модели можно переносить на само ПО.

[](http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/145/objects/5/files/5_4.gif)

**Рис.** Схема процесса проверки свойств ПО на моделях

Обычно при помощи проверки свойств на моделях анализируют два вида свойств алгоритмов, использованных при построении ПО. **Свойства безопасности (safety properties)** утверждают, что нечто нежелательное никогда не случится в ходе работы ПО. **Свойства живучести (liveness properties)** утверждают, наоборот, что нечто желательное при любом развитии событий произойдет в ходе его работы.

Примером свойства первого типа служит отсутствие **взаимных блокировок (deadlocks)**. Взаимная блокировка возникает, если каждый из группы параллельно работающих в проверяемом ПО процессов или потоков ожидает прибытия данных или снятия блокировки ресурса от одного из других, а тот не может продолжить выполнение, ожидая того же от первого или от третьего процесса, и т.д.

Примером свойства живости служит гарантированная доставка сообщения, обеспечиваемая некоторыми протоколами — как бы ни развивались события, если сетевое соединение между машинами будет работать, посланное с одной стороны (процессом на первой машине) сообщение будет доставлено другой стороне (процессу на второй машине).

В классическом подходе к проверке на моделях проверяемые свойства формализуются в виде формул так называемых **временных логик**. Их общей чертой является наличие операторов "всегда в будущем" и "когда-то в будущем". Заметим, что второй оператор может быть выражен с помощью первого и отрицания — то, что некоторое свойство когда-то будет выполнено, эквивалентно тому, что отрицание этого свойства не будет выполнено всегда. Свойства безопасности легко записываются в виде "всегда будет выполнено отрицание нежелательного свойства", а свойства живости — в виде "когда-то обязательно будет выполнено желаемое".

Проверяемая программа в классическом подходе моделируется при помощи конечного автомата. Проверка, выполняемая автоматически, состоит в том, что для всех достижимых при работе системы состояний этого автомата проверяется нужное свойство. Если оно оказывается выполненным, выдается сообщение об успешности проверки, если нет — выдается трасса, последовательность выполнения отдельных шагов программы, моделируемых переходами автомата, приводящая из начального состояния в такое, в котором нужное свойство нарушается. Эта трасса используется для анализа происходящего и исправления либо программы, либо модели, если ошибка находится в ней.

Основная проблема этого подхода — огромное, а часто и бесконечное, количество состояний в моделях, достаточно хорошо отражающих поведение реальных программ. Для борьбы с комбинаторным взрывом состояний применяются различные методы оптимизации представления автомата, выделения и поиска состояний, существенных для выполнения проверяемого свойства.

**Ошибки в программах**

Ошибками в ПО, вообще говоря, являются все возможные несоответствия между демонстрируемыми характеристиками его качества и сформулированными или подразумеваемыми требованиями и ожиданиями пользователей.

В англоязычной литературе используется несколько терминов, часто одинаково переводящихся как "ошибка" на русский язык:

* **defect** — самое общее нарушение каких-либо требований или ожиданий, не обязательно проявляющееся вовне (к дефектам относятся нарушения стандартов кодирования, недостаточная гибкость системы и пр.).
* **failure** — наблюдаемое нарушение требований, проявляющееся при каком-то реальном сценарии работы ПО. Это можно назвать проявлением ошибки.
* **fault** — ошибка в коде программы, вызывающая нарушения требований при работе (failures), то место, которое надо исправить. Хотя это понятие используется довольно часто, оно, вообще говоря, не вполне четкое, поскольку для устранения нарушения можно исправить программу в нескольких местах. Что именно надо исправлять, зависит от дополнительных условий, выполнение которых мы хотим при этом обеспечить, хотя в некоторых ситуациях наложение дополнительных ограничений не устраняет неоднозначность.
* **error** — используется в двух смыслах.

Первый — это ошибка в ментальной модели программиста, ошибка в его рассуждениях о программе, которая заставляет его делать ошибки в коде (faults). Это, собственно, ошибка, которую сделал человек в своем понимании свойств программы.

Второй смысл — это некорректные значения данных (выходных или внутренних), которые возникают при ошибках в работе программы.

Эти понятия некоторым образом связаны с основными источниками ошибок. Поскольку при разработке программ необходимо сначала понять задачу, затем придумать ее решение и закодировать его в виде программы, то, соответственно, основных источников ошибок три:

* Неправильное понимание задач.

Очень часто люди не понимают, что им пытаются сказать другие. Так же и разработчики ПО не всегда понимают, что именно нужно сделать. Другим источником непонимания служит отсутствие его у самих пользователей и заказчиков — достаточно часто они могут просить сделать несколько не то, что им действительно нужно.

Для предотвращения неправильного понимания задач программной системы служит анализ предметной области.

* Неправильное решение задач.

Зачастую, даже правильно поняв, что именно нужно сделать, разработчики выбирают неправильный подход к тому, как это делать. Выбираемые решения могут обеспечивать лишь некоторые из требуемых свойств, они могут хорошо подходить для данной задачи в теории, но плохо работать на практике, в конкретных обстоятельствах, в которых должно будет работать ПО.

Помочь в выборе правильного решения может сопоставление альтернативных решений и тщательный анализ их на предмет соответствия всем требованиям, поддержание постоянной связи с пользователями и заказчиками, предоставление им необходимой информации о выбранных решениях, демонстрация прототипов, анализ пригодности выбираемых решений для работы в том контексте, в котором они будут использоваться.

* Неправильный перенос решений в код.

Имея правильное решение правильно понятой задачи, люди, тем не менее, способны сделать достаточно много ошибок при воплощении этих решений. Корректному представлению решений в коде могут помешать как обычные опечатки, так и забывчивость программиста или его нежелание отказаться от привычных приемов, которые не дают возможности аккуратно записать принятое решение.

С ошибками такого рода можно справиться при помощи инспектирования кода, взаимного контроля, при котором разработчики внимательно читают код друг друга, опережающей разработки модульных тестов и тестирования.

Первое место в неформальном состязании за место "самой дорого обошедшейся ошибки в ПО " долгое время удерживала ошибка, приведшая к неудаче первого запуска ракеты Ариан-5 4 июня 1996 года, стоившая около $500 000 000. После произошедшего 14 августа 2003 года обширного отключения электричества на северо-востоке Северной Америки, стоившего экономике США и Канады от 4 до 10 миллиардов долларов, это место можно отдать спровоцировавшей его ошибке в системе управления электростанцией. Широко известны также примеры ошибок в системах управления космическими аппаратами, приведшие к их потере или разрушению. Менее известны, но не менее трагичны, ошибки в ПО, управлявшем медицинским и военным оборудованием, некоторые из которых привели к гибели людей.

Стоит отметить, что в большинстве примеров ошибок, имевших тяжелые последствия, нельзя однозначно приписать всю вину за случившееся ровно одному недочету, одному месту в коде. Ошибки очень часто "охотятся стаями". К тяжелым последствиям чаще всего приводят ошибки системного характера, затрагивающие многие аспекты и элементы системы в целом. Это значит, что при анализе такого происшествия обычно выявляется множество частных ошибок, нарушений действующих правил, недочетов в инструкциях и требованиях, которые совместно привели к создавшейся ситуации.

Даже если ограничиться рассмотрением только ПО, часто одно проявление ошибки (failure) может выявить несколько дефектов, находящихся в разных местах. Такие ошибки возникают, как показывает практика, в тех ситуациях, поведение в рамках которых неоднозначно или недостаточно четко определяется требованиями (а иногда и вообще никак не определяется — признак неполного понимания задачи). Поэтому разработчики различных модулей ПО имеют возможность по-разному интерпретировать те части требований, которые относятся непосредственно к их модулям, а также иметь разные мнения по поводу области ответственности каждого из взаимодействующих модулей в данной ситуации. Если различия в их понимании не выявляются достаточно рано, при разработке системы, то становятся "минами замедленного действия" в ее коде.

Например, анализ катастрофы Ариан-5 показал следующее.

* Ариан-5 была способна летать при более высоких значениях ускорений и скоростей, чем это могла делать ракета предыдущей серии, Ариан-4.

Однако большое количество процедур контроля и управления движением по траектории в коде управляющей системы было унаследовано от Ариан-4. Большинство таких процедур не были специально проверены на работоспособность в новой ситуации, как в силу большого размера кода, который надо было проанализировать, так и потому, что этот код раньше не вызывал проблем, а соотнести его со специфическими характеристиками полета ракет вовремя никто не сумел.

* В одной из таких процедур производилась обработка горизонтальной скорости ракеты. При выходе этой величины за границы, допустимые для Ариан-4, создавалась исключительная ситуация переполнения.

Надо отметить, что обработка нескольких достаточно однородных величин производилась по-разному — семь переменных могли вызвать исключительную ситуацию данного вида, обработка четырех из них была защищена от этого, а три оставшихся, включая горизонтальную скорость, оставлены без защиты. Аргументом для этого послужило выдвинутое при разработке требование поддерживать загрузку процессора не выше 80%. "Нагружающие" процессор защитные действия для этих переменных не были использованы, поскольку предполагалось, что эти величины будут находиться в нужных пределах в силу физических ограничений на параметры движения ракеты. Обоснований для поддержки именно такой загрузки процессора и того, что отсутствие обработки переполнения выбранных величин будет способствовать этому, найдено не было.

* Когда такая ситуация действительно случилась, т.е. горизонтальная скорость ракеты превысила определенное значение, она не была обработана соответствующим образом, и в результате ею вынужден был заняться модуль, обеспечивающим отказоустойчивость программной системы в целом.
* Этот модуль, в силу отсутствия у него какой-либо возможности обрабатывать такие ошибки специальным образом, применил обычный прием — остановил процесс, в котором возникла ошибка, и запустил другой процесс с теми же исходными данными. Как легко догадаться, эта же ошибка повторилась и во втором процессе.
* Не в силах получить какие-либо осмысленные данные о текущем состоянии полета, система управления использовала ранее полученные, которые уже не соответствовали действительности. При этом были ошибочно включены боковые двигатели "для корректировки траектории", ракета начала болтаться, угол между нею и траекторией движения стал увеличиваться и достиг 20 градусов. В результате она стала испытывать чрезмерные аэродинамические нагрузки и была автоматически уничтожена.

# Методы оценки технико-экономических показателей ПС

**Технико-экономический анализ разработки проектов программных средств**

Технико-экономический анализ разработки проектов программных средств - это выбор и прогнозирование наиболее адекватных экономических и функциональных критериев для обобщенного описания эффективности, стоимости создания и использования проектов программных средств в зависимости от их назначения, области применения и прочих факторов.

Если разрабатываемое программное средство в дальнейшем будет представлено в качестве продукции на рынке, то технико-экономический анализ его разработки имеет еще большую актуальность.

Методы технико-экономического анализа в зависимости от дальнейшего применения программного средства можно разделить на две группы:

1) Программное средство ориентировано на массовое тиражирование и продажу на рынке. Конкретная область применения и заказчик неизвестны. В данном случае методы технико-экономического анализа базируются на данных, полученных при исследовании рынка сбыта программного средства. Изучаются характеристики программных продуктов разработчиков-конкурентов: эффективность, стоимость, трудозатраты; большое внимание уделяется показателям конкурентоспособности продуктов программных средств. На основании полученной информации оптимизируются технико-экономические показатели разрабатываемого программного средства.

2) Программное средство предназначено для конкретного пользователя и ориентировано на заранее известную область применения. При этом заказчик выбирает на рынке услуг конкурентоспособного разработчика, которого оценивает на возможность реализовать проект с необходимым качеством с учетом ограничения требуемых бюджета, сроков и других ресурсов. Заказчик заинтересован в получении программного средства высокого качества при минимальных затратах и в приемлемые сроки, а разработчик стремится к получению достаточных ресурсов на реализацию программного средства и максимальной оплаты за свою работу.

Цели технико-экономического обоснования:

1. Прогнозирование реальных затрат: изучается процесс разработки программ, происходит определение метрик технико-экономических показателей. На основе обобщения этих метрик выявляется трудоемкость и производительность труда, а также факторы, влияющие на эти показатели. Разрабатываются и внедряются методики сбора первичных данных, с помощью которых определяется длительность всего процесса разработки.

2. Создание методов и методик прогнозирования затрат и длительности разработки: методики базируются на анализе аналогов - прототипов и должны учитывать полученные значения технико-экономических показателей, основные характеристики создаваемых программных средств, а также технологию, оснащенность и организацию их разработки.

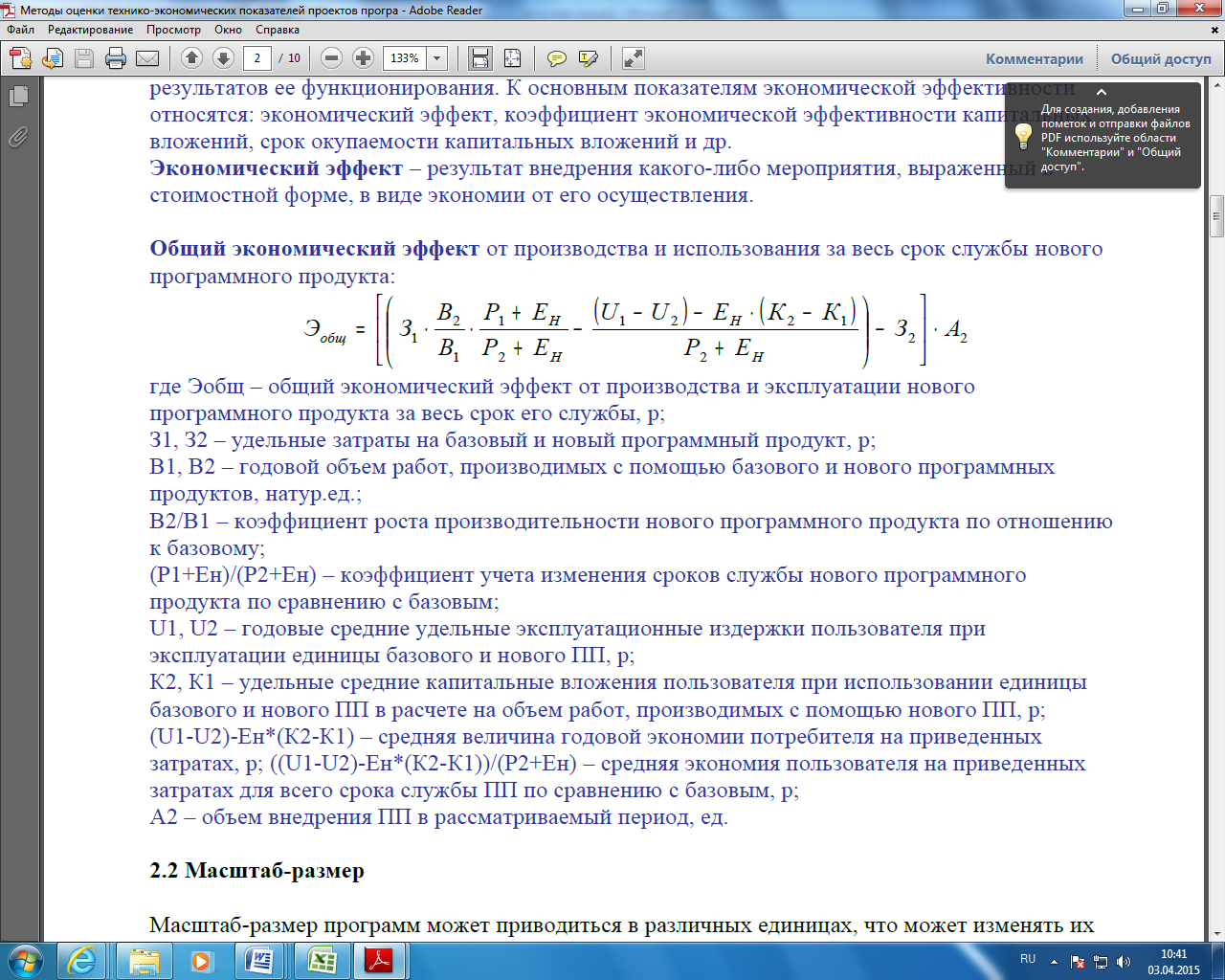
3. Обоснование и создание методов и средств снижения совокупных затрат и сроков разработки сложных программных средств: решаются задачи эффективного распределения трудовых ресурсов, повышение уровня автоматизации технологий разработки, выбор методов и средств, позволяющих снизить длительность разработки и проч.

4. Создание методических и нормативных документов: появляется возможность управления затратами на разработку, количеством и качеством создаваемых программных средств и их компонентов.

**Перечень основных технико-экономических показателей**

**Эффективность** – это характеристика системы с точки зрения соотношения затрат и результатов ее функционирования. К основным показателям экономической эффективности относятся: экономический эффект, коэффициент экономической эффективности капитальных вложений, срок окупаемости капитальных вложений и др.

**Экономический эффект** – результат внедрения какого-либо мероприятия, выраженный в стоимостной форме, в виде экономии от его осуществления. Общий экономический эффект от производства и использования за весь срок службы нового программного продукта:



где Эобщ – общий экономический эффект от производства и эксплуатации нового программного продукта за весь срок его службы, р;

З1, З2 – удельные затраты на базовый и новый программный продукт, р;

В1, В2 – годовой объем работ, производимых с помощью базового и нового программных продуктов, натур.ед.;

В2/В1 – коэффициент роста производительности нового программного продукта по отношению к базовому;

(Р1+Ен)/(Р2+Ен) – коэффициент учета изменения сроков службы нового программного

продукта по сравнению с базовым;

U1, U2 – годовые средние удельные эксплуатационные издержки пользователя при

эксплуатации единицы базового и нового ПП, р;

К2, К1 – удельные средние капитальные вложения пользователя при использовании единицы базового и нового ПП в расчете на объем работ, производимых с помощью нового ПП, р;

(U1-U2)-Ен\*(К2-К1) – средняя величина годовой экономии потребителя на приведенных затратах, р; ((U1-U2)-Ен\*(К2-К1))/(Р2+Ен) – средняя экономия пользователя на приведенных затратах для всего срока службы ПП по сравнению с базовым, р;

А2 – объем внедрения ПП в рассматриваемый период, ед.

**Масштаб-размер** программ может приводиться в различных единицах, что может изменять их численные значения для одних и тех же программ в несколько раз. Размер исходных текстов программ, прежде всего, отражает трудоемкость и длительность их разработки и позволяет оценивать относительные характеристики производительности труда специалистов разработчиков.

Единицы измерения масштаба-размера программных средств можно разделить на две группы:

1. Единицы измерения, которые разрабатываются и анализируются человеком (отражают сложность и трудоемкость создания программных средств и их компонентов). При этом различаются следующие типы текста программы:

* новый код текстов программ, разработанный полностью для нового приложения, который не включает фрагменты и процедуры ранее написанного и испытанного кода;
* модифицируемый код, разработанный для предыдущих проектов программных средств, который становится пригодным для использования в новом проекте, после внесения умеренного объема изменений;
* повторно используемый код, разработанный для предыдущих программных средств, который будет полностью пригодным для новых приложений без внесения каких-либо изменений;
* наследственный код, разработанный для предыдущих проектов, использование которого ожидается новым комплексом программ.

2. Единицы измерения, которые размещаются в реализующем компьютере (характеризуют объем памяти и производительность компьютера, необходимые для нормального функционирования разрабатываемого программного средства)

**Затраты, связанные с проектом**

К основным затратам можно отнести:

* допустимые трудозатраты на разработку программного средства с требуемым качеством
* время (длительность полного цикла создания программного продукта)
* необходимое и доступное число специалистов соответствующей квалификации

Трудоемкость разработки программных средств наиболее сильно зависит от масштаба-размера программ, выраженного числом операторов на ассемблере или строк на языке программирования высокого уровня.

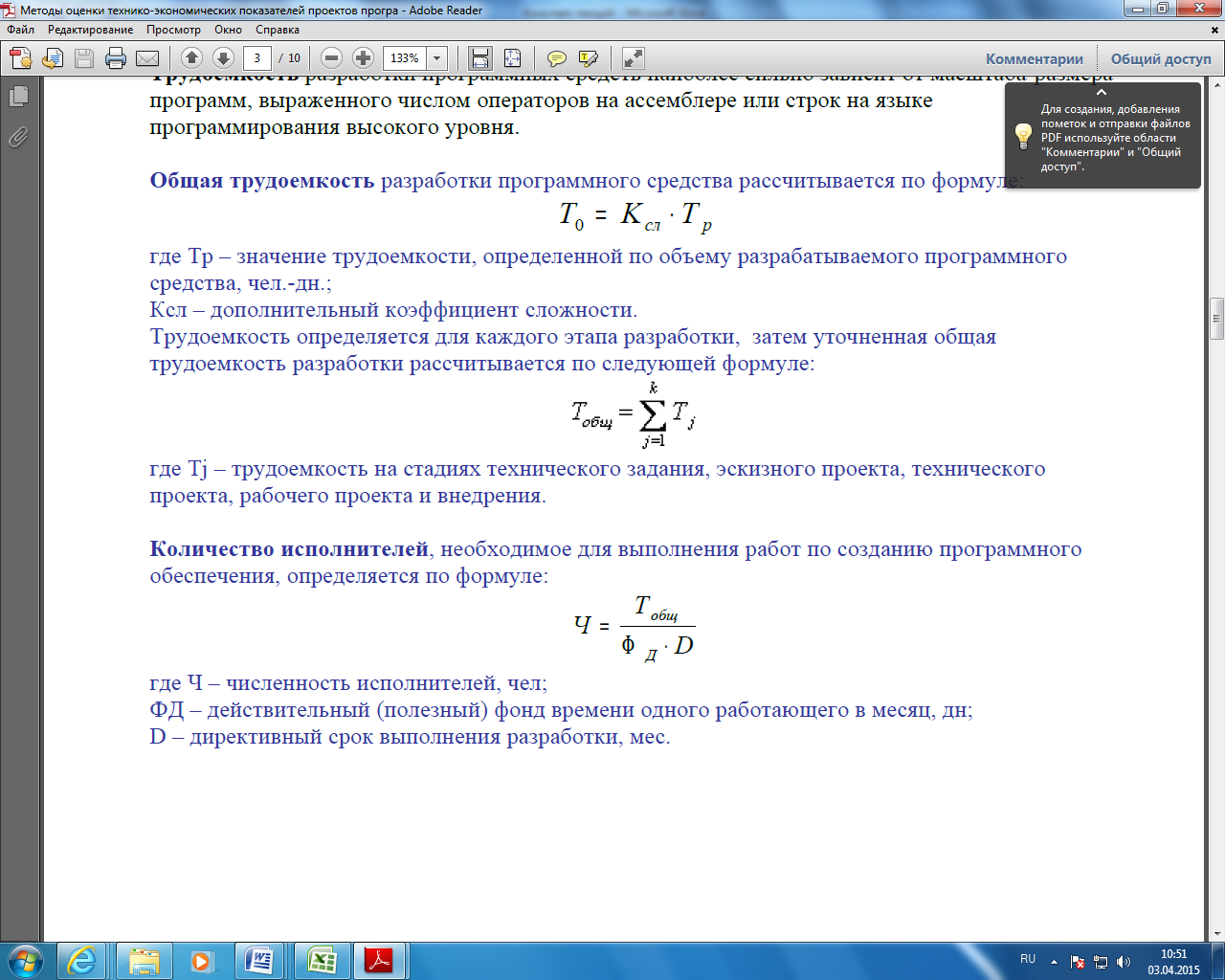
Общая трудоемкость разработки программного средства рассчитывается по формуле:

Т0=Ксл\*Тр

где Тр – значение трудоемкости, определенной по объему разрабатываемого программного средства, чел.-дн.;

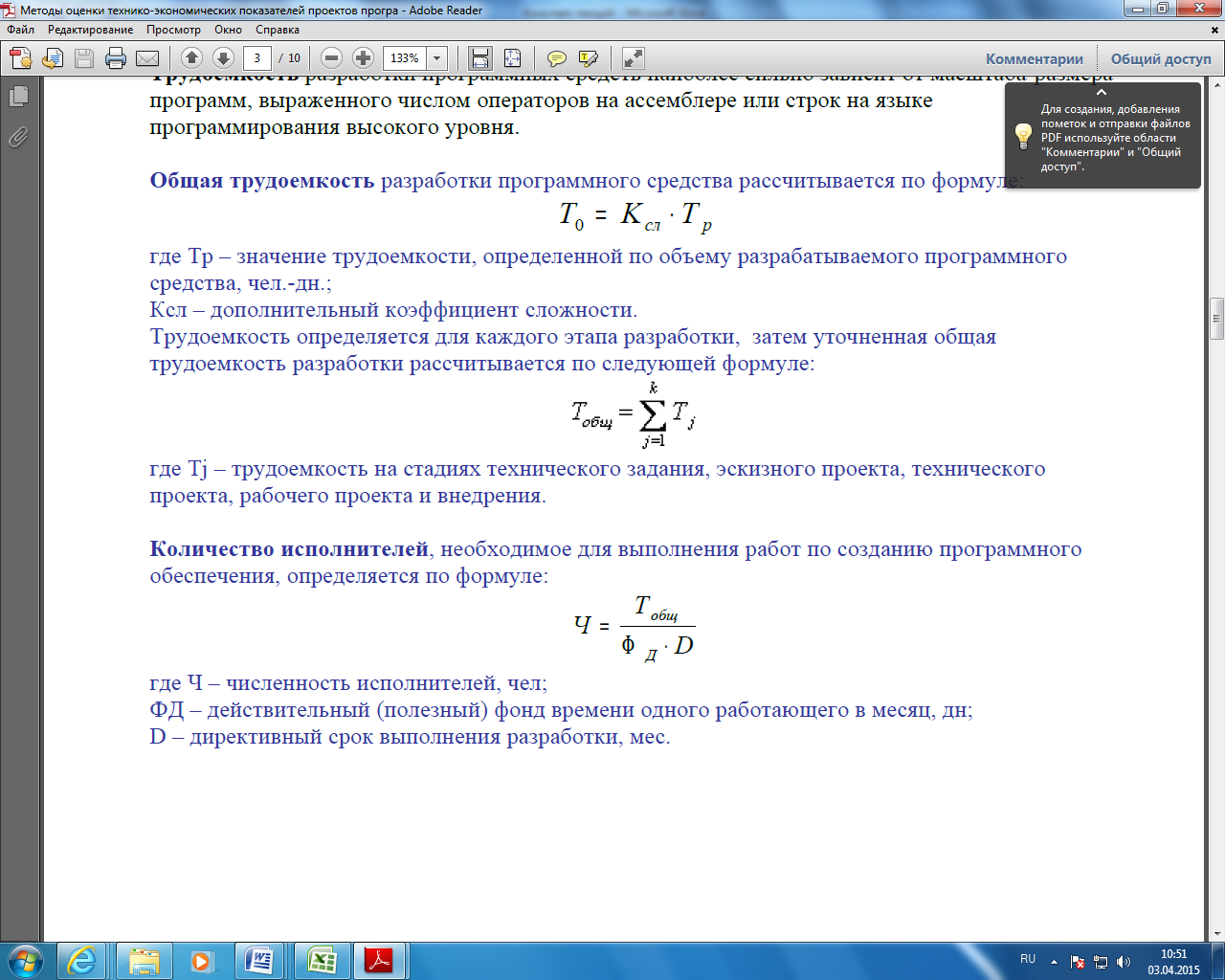
Ксл – дополнительный коэффициент сложности.

Трудоемкость определяется для каждого этапа разработки, затем уточненная общая трудоемкость разработки рассчитывается по следующей формуле:



где Tj – трудоемкость на стадиях технического задания, эскизного проекта, технического проекта, рабочего проекта и внедрения.

Количество исполнителей, необходимое для выполнения работ по созданию программного обеспечения, определяется по формуле:



где Ч – численность исполнителей, чел;

ФД – действительный (полезный) фонд времени одного работающего в месяц, дн;

D – директивный срок выполнения разработки, мес.

**Конкурентоспособность проекта**

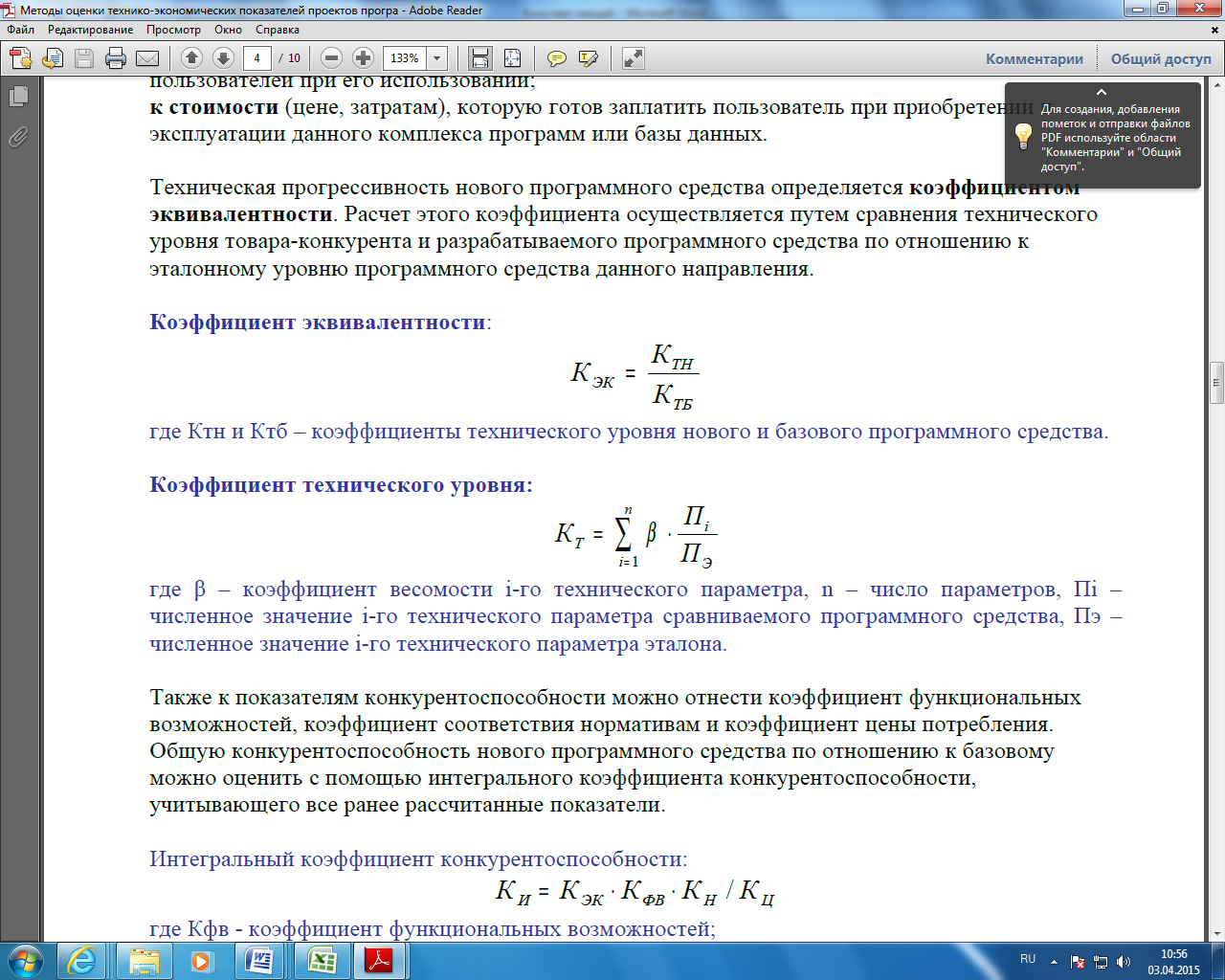
Конкурентоспособность товара – это степень его соответствия выбранному рынку по коммерческим, техническим и экономическим показателям, обеспечивающим возможность сбыта товара на этом рынке. Это те характеристики, которые выгодно отличают данный товар от товаров-конкурентов.

Конкурентоспособность определяется как соотношение возможной эффективности (ценности, достоинств) последующего применения ПС и способности удовлетворить потребности пользователей при его использовании;

к стоимости (цене, затратам), которую готов заплатить пользователь при приобретении и эксплуатации данного комплекса программ или базы данных.

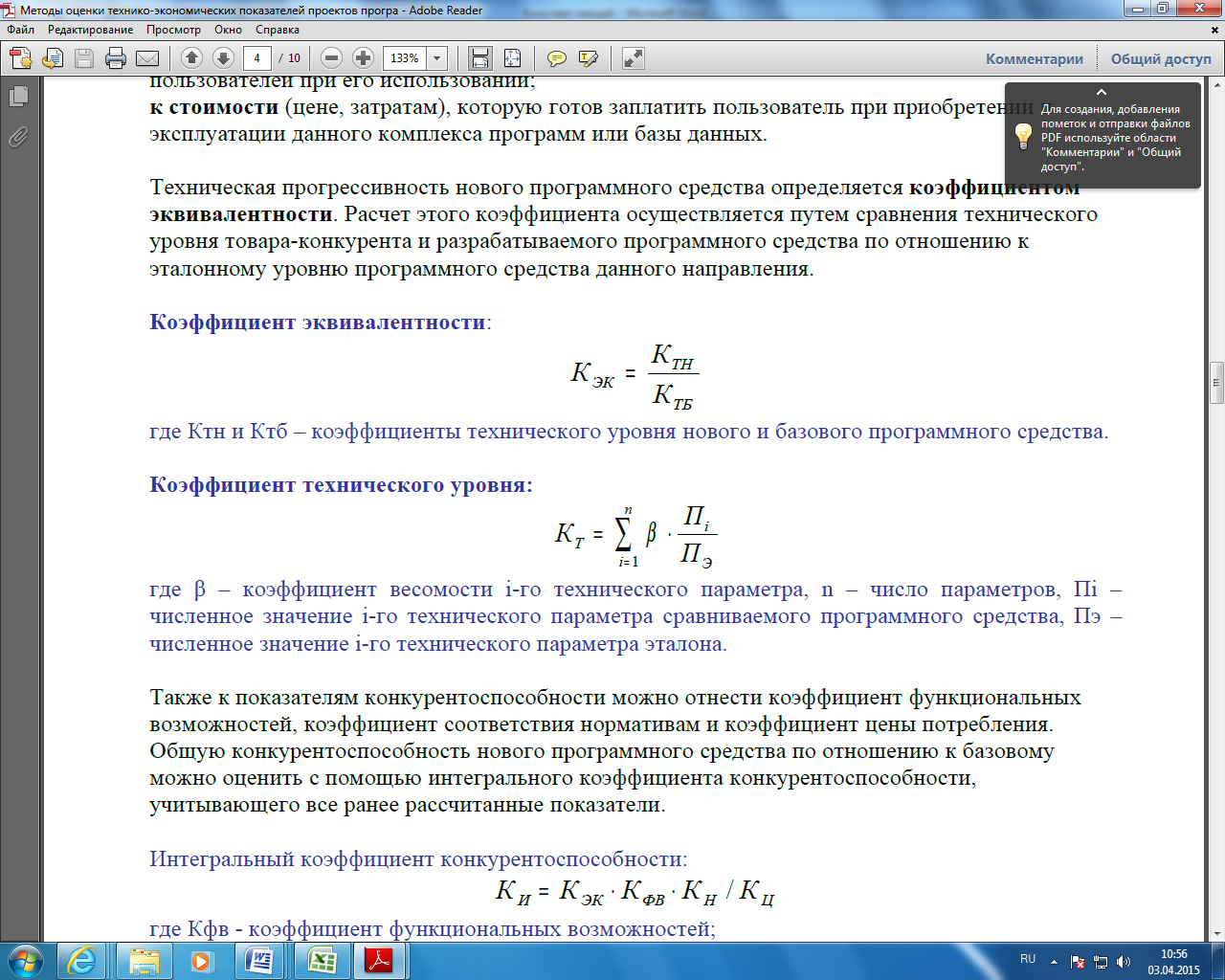
Техническая прогрессивность нового программного средства определяется коэффициентом эквивалентности. Расчет этого коэффициента осуществляется путем сравнения технического уровня товара-конкурента и разрабатываемого программного средства по отношению к эталонному уровню программного средства данного направления.

Коэффициент эквивалентности:



где Ктн и Ктб – коэффициенты технического уровня нового и базового программного средства.

Коэффициент технического уровня:



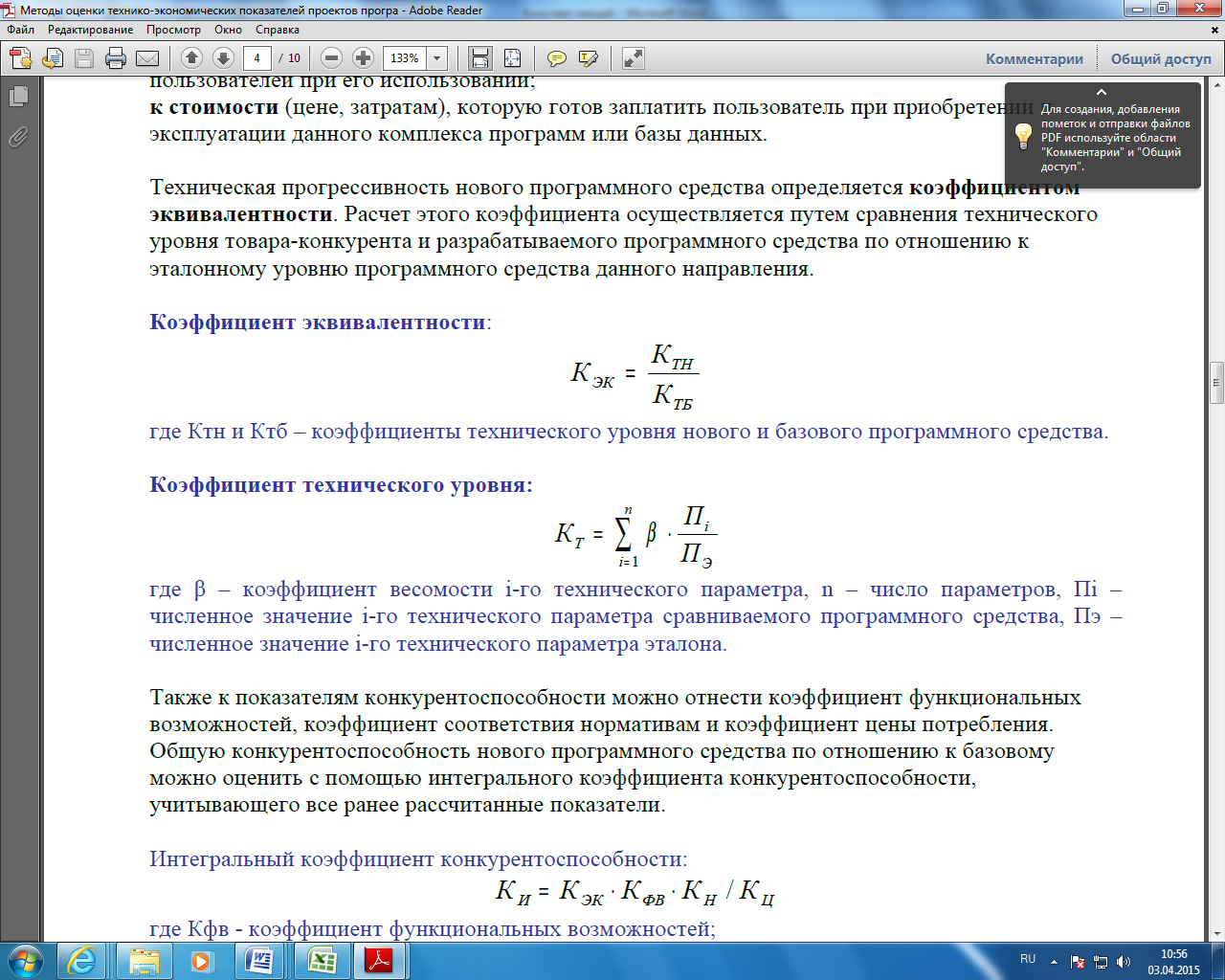
где β – коэффициент весомости i-го технического параметра, n – число параметров, Пi –

численное значение i-го технического параметра сравниваемого программного средства, Пэ – численное значение i-го технического параметра эталона.

Также к показателям конкурентоспособности можно отнести коэффициент функциональных возможностей, коэффициент соответствия нормативам и коэффициент цены потребления.

Общую конкурентоспособность нового программного средства по отношению к базовому можно оценить с помощью интегрального коэффициента конкурентоспособности, учитывающего все ранее рассчитанные показатели.

Интегральный коэффициент конкурентоспособности:



где Кфв - коэффициент функциональных возможностей;

Кн – коэффициент соответствия нового программного средства нормативам;

Кц – коэффициент цены потребления.

**Сложность создания программного средства**

Оценка таких характеристик качества, как надежность и сопровождаемость не может быть выполнена до тех пор, пока не будет получена хотя бы первая версия программного средства. Еще до конца разработки программного средства метрика сложности позволяет прогнозировать, какие компоненты программного средства будут способны к отказу, сложнее тестироваться и негативно реагировать на изменение настроек или конфигурации.

Метрики сложности непосредственно определяют трудоемкость разработки комплекса

программ. Метрики сложности можно разделить на две группы:

· метрики сложности потока управления программ (метрика Маккейба, метрика Майерса, метрика подсчета точек пересечения, метрика Джилба, метрика граничных значений) С помощью этих метрик оперируют либо плотностью управляющих переходов внутри программ, либо взаимосвязями этих переходов.

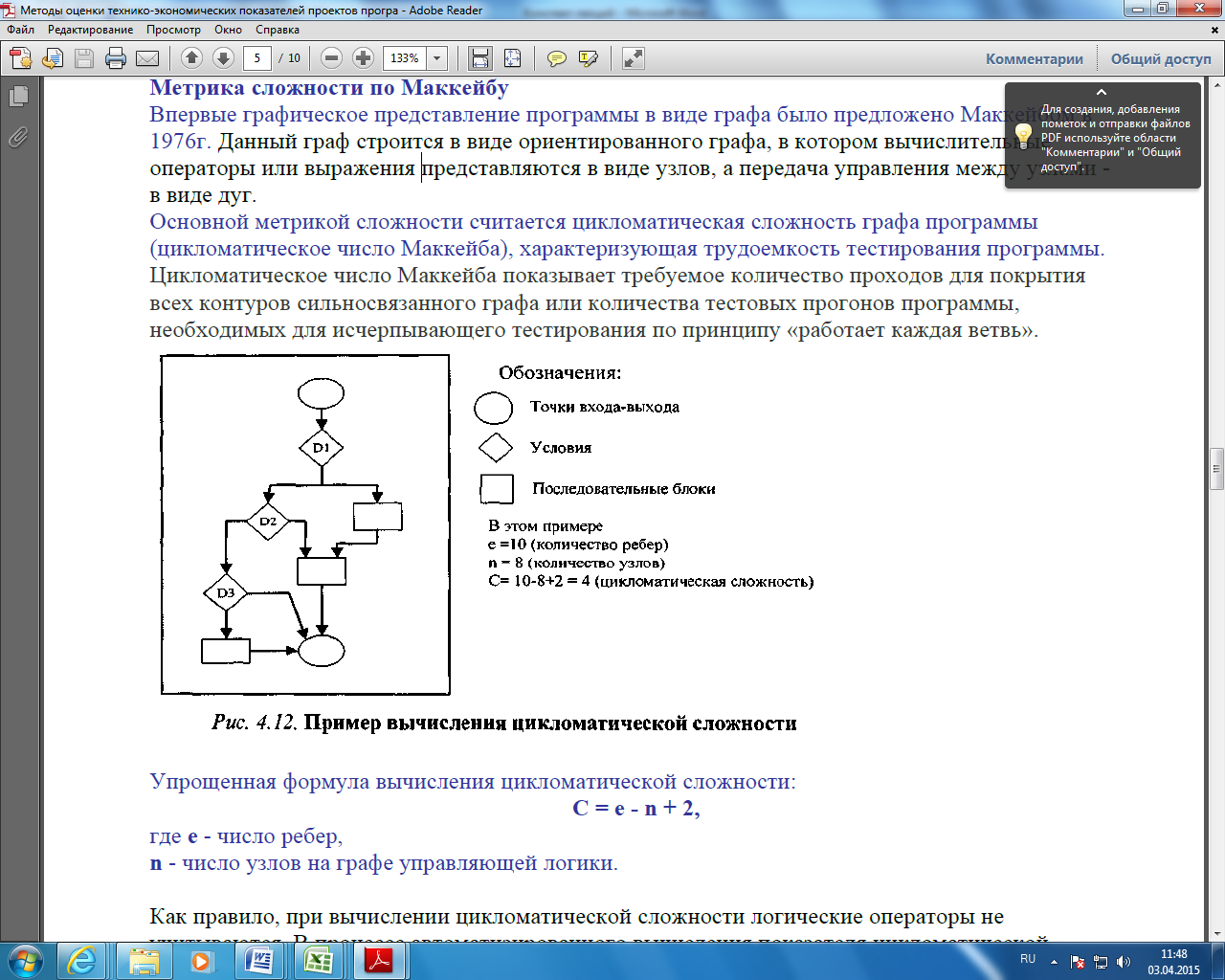
метрики сложности потока данных (метрика обращения к глобальным переменным, метрика спена, метрика Чепина) Метрики сложности потока данных определяют использование, конфигурацию и размещение данных в программе.

**Метрика сложности по Маккейбу**

Впервые графическое представление программы в виде графа было предложено Маккейбом в 1976г. Данный граф строится в виде ориентированного графа, в котором вычислительные операторы или выражения представляются в виде узлов, а передача управления между узлами - в виде дуг.

Основной метрикой сложности считается цикломатическая сложность графа программы (цикломатическое число Маккейба), характеризующая трудоемкость тестирования программы.

Цикломатическое число Маккейба показывает требуемое количество проходов для покрытия всех контуров сильносвязанного графа или количества тестовых прогонов программы, необходимых для исчерпывающего тестирования по принципу «работает каждая ветвь»

.

Пример вычисления цикломатической сложности

Упрощенная формула вычисления цикломатической сложности:

C = e - n + 2,

где e - число ребер,

n - число узлов на графе управляющей логики.

Как правило, при вычислении цикломатической сложности логические операторы не учитываются. В процессе автоматизированного вычисления показателя цикломатической сложности построение графа не осуществляется. Вычисление показателя производится на основании подсчета числа операторов управляющей логики (if, switch и т.д.) и возможного количества путей исполнения программы.

**Методы измерения масштаба-размера программных средств**

Ранее уже упоминалось, что единицы измерения масштаба-размера программных средств делятся на единицы измерения, которые разрабатываются и анализируются человеком; и единицы измерения, которые размещаются в реализующем компьютере.

Единицы измерения масштаба-размера программных средств, разрабатываемые человеком, выражаются в:

* символах исходного текста программы на любых языках программирования;
* операторах языка программирования уровня ассемблера;
* строках текста программы на языке программирования высокого уровня;
* строках кода в терминах Lines of code (LOC);
* функциональных точках Function Point (FP).

Число символов в тексте программы достаточно адекватно отражает сложность разработки ПС. При возрастании уровня языка программирования число символов, имеющих непосредственное смысловое содержание, убывает за счет объединения их в группы, отражающие обобщенные понятия языка программирования. Допустимое группирование обычно ограничено 6-8 символами. Первичная оценка сложности текста на языках высокого уровня может проводиться прямым подсчетом числа символов в тексте.

Число операторов на ассемблере широко применяется на практике для оценки размера текста программ. Макрокоманды ассемблера могут значительно изменять размер текста программы. Целесообразно оценку размера программ на ассемблере проводить с учетом раскрытия макрокоманд. При этом условии число операторов в текстах программ на разных ассемблерах является наиболее стабильной характеристикой их размера.

Число строк на языках высокого уровня в значительной степени зависит от конкретных особенностей языка. Языки программирования высокого уровня ориентированы на определенные классы задач. Однако применение их для других классов приводит к расширению текста и не всегда рационально.

При оценке показателя LOC на первоначальном этапе производится детализация структуры программного средства с помощью его разделения на логические подсистемы. Далее производятся процессы измерения и суммирования. Величина размера каждого компонента может быть получена путем опроса экспертов, разрабатывавших подобные системы, либо путем использования данных опроса потенциальных разработчиков подобных систем. В результате становится возможной оценка размеров каждого компонента на нижних уровнях структуры.

После сложения результатов измерений полученный итог будет называться оценкой размера «снизу-вверх». При этом каждая учитываемая строка исходного кода содержит лишь один оператор. Определения данных учитывается лишь один раз, не учитываются строки, которые содержат комментарии и проч.

Под строкой понимается любой оператор программы, поскольку именно оператор, а не отдельно взятая строка является тем интеллектуальным "квантом" программы, опираясь на который можно строить метрики сложности ее создания. Для измерения размера программы в терминах LOC можно использовать метрику Холстеда.

Основу метрики Холстеда составляют четыре измеряемых характеристики программы:

n1 - число уникальных операторов программы, включая символы-разделители, имена процедур и знаки операций (словарь операторов);

n2 - число уникальных операндов программы (словарь операндов);

N1 - общее число операторов в программе;

N2 - общее число операндов в программе.

Вводятся следующие оценки:

словарь программы

n1=n1+n2,

длину программы

N=N1+N2,

объем программы

V=N\*log2(n) (бит).

Под битом подразумевается логическая единица информации - символ, оператор, операнд.

Использование функциональных точек в качестве единиц измерения основывается на том, что размер программного средства можно оценивать в терминах количества и сложности функций, реализованных в данном программном коде, а не только посредством количества строк кода. Подход с использованием функциональных точек учитывает размер и функциональные свойства (сложность) компонентов.

Единицы измерения масштаба-размера программных средств, размещаемых в

реализующей ЭВМ, выражаются в:

* байтах, занятых текстом программы в объектном коде и переменными базы данных;
* командах (операциях) реализующей ЭВМ, составляющих текст исполняемой программы в объектном коде;
* словах памяти, обусловленных структурой данной реализующей ЭВМ, используемых для хранения исполняемых программ и базы данных при функционировании программных средств.

**Методы оценки технико-экономических показателей**

***Линейный подход***

Стоимость разработки определяется из количественной оценки трудозатрат Т (в человеко-месяцах или человеко-часах) и их удельной стоимости Ц:

С = Т × Ц

Трудозатраты в этом случае определяются как:

Т = Р × П,

где Р – размер исходного кода; П – временная производительность

Методы оценки с использованием эмпирических данных

***Использование функциональных точек***

Функциональная точка - некоторый блок, модуль программы, который выполняет определённую функцию и может рассматриваться отдельно от других.

Функциональные точки можно охарактеризовать с помощью:

* внутренней сложности (насколько трудно реализовывать)
* транзакционной сложности (насколько часто и сложно взаимодействие с внешними модулями)
* сложность реализации
* сложность отладки
* сложность, связанная с объёмом и плотностью кода (оцениваются после создания прототипа системы)

Типы функциональных точек:

Internal Logical Files (ILF) – понятная для пользователя группа логически-связанных данных, которые находятся полностью внутри приложения и обслуживаются через внешние входы системы.

External Interface Files (EIF) – понятная для пользователя группа логически-связанных данных, которые используются только для целей ссылки. Сюда относятся так же данные, которые использует разрабатываемое приложение, но которые управляются другим приложением.

External Inputs (EI) - элементарный процесс, в котором данные вводятся в систему снаружи. Эти данные могут поступать от экранов ввода данных, электронных устройств или другого приложения.

External Outputs (EO) - элементарный процесс, в котором данные выводятся из системы. Данные создают отчеты или внешние файлы, которые пересылаются другим приложениям. Они могут создаваться одним или несколькими внутренними логическими файлами (ILF) и внешним файлом интерфейса (EIF).

External Inquiry (EQ) - элементарный процесс, в котором данные выводятся из сисиемы в результате выполнения комплексного запроса и процесса обработки внутренних логических файлов (ILF) и внешних интерфейсных файлов (EIF).

**COCOMO**

Модель для оценки стоимости разработок ПО COCOMO (COnstructive COst MOdel) была представлена в 1981 г. Барри Боэмом. В COCOMO используются три уровня детализации: базовый, промежуточный и подробный. Также предусмотрено три режима использования модели в зависимости от размеров команды и проекта. Модель COCOMO предназначена только для каскадной модели жизненного цикла.

Режимы модели COCOMO

|  |  |  |
| --- | --- | --- |
| Название режима | Размер проекта | Описание |
| Органичный | До 50  KLOC | Некрупный проект разрабатывается небольшой командой, для которой нехарактерны нововведения, и среда остается стабильной |
| Сблокированный | 50–300  KLOC | Относительно небольшая команда занимается проектом среднего размера, в процессе разработки необходимы определенные инновации, среда характеризуется незначительной нестабильностью |
| Внедренный | Более 300  KLOC | Большая команда разработчиков трудится над крупным проектом, необходим значительный объем инноваций, среда состоит из множества элементов, которые не характеризуются стабильностью |

Для оценки трудозатрат применяется формула:

Т = a × Р b,

где a и b – константы, которые зависят от режима использования модели.

В соответствии с этой формулой трудозатраты вообще нелинейно зависят от размера проекта и скачкообразно изменяются при смене режима.

Длительность (F) выполнения проекта вычисляется по формуле:

F = 2,5 × Т k,

При этом значение константы k изменяется в зависимости от размера проекта:

Значения коэффициентов модели COCOMO в зависимости от режима использования

|  |  |  |  |
| --- | --- | --- | --- |
| Название режима | Значение коэффициента a | Значение коэффициента b | Значение коэффициента k |
| Органичный | 2,4 | 1,05 | 0,38 |
| Сблокированный | 3,0 | 1,12 | 0,35 |
| Внедренный | 3,6 | 1,20 | 0,32 |

**COCOMOII**

Модель COCOMOII была представлена в 1997 г и предназначалась не только для каскадной модели жизненного цикла, но и для спиральной и итерационной. В этой модели допускается измерять размер проекта не только числом строк кода, но и функциональными точками.

COCOMO II фактически объединяет три различные подмодели:

* Композиционная прикладная: ориентирована на проекты, создаваемые с применением современных инструментальных средств и UML, использует в качестве метрики объектные точки
* Ранней разработки проекта: применяется для получения приближенных оценок по проекту до определения его архитектуры, использует в качестве метрик количество строк кода или функциональные точки
* Постархитектурная модель: наиболее детализированная модель, используется после разработки архитектуры проекта и позволяет получить самые точные оценки, применяет в качестве метрик количество строк кода или функциональные точки

Трудоемкость для модели COCOMOII определяется как:

T=C1\*EAF\*(Размер)P1

Продолжительность работ:

F=C2\*(Работа)P2,

где T - трудоемкость, человеко-месяцах;

C1 - масштабирующий коэффициент;

EAF - уточняющий фактор;

Размер - размер конечного продукта (кода, созданного человеком) в SLOC;

P1 - показатель степени, характеризующий экономию при больших масштабах;

F - общая длительность разработки в месяцах;

C2 - масштабирующий коэффициент для сроков исполнения;

P2 - показатель степени, который характеризует инерцию и распараллеливание, присущие управлению разработкой ПО.

# Сертификация

Программный продукт, как и любой другой, должен быть сертифицирован, т.е. пройти проверку и иметь документ, удостоверяющий его соответствие требованиям стандартов и других нормативных документов. Имеется два сертификационных документа: сертификат соответствия и знак соответствия.

Определения:

**Сертификат соответствия** – это документ, выданный по правилам системы сертификации для подтверждения соответствия сертифицированной продукции установленным требованиям.

**Знак соответствия** - это зарегистрированный в установленном порядке знак, которым по правилам определенной системы сертификации подтверждается соответствие маркированной им продукции установленным требованиям.

**Система сертификации** – совокупность участников сертификации, которые проводят сертификацию продукции по устанавливаемым в этой системе определенным правилам в соответствии с законом. Система сертификации создается федеральными органами исполнительной власти.

Сертификация включает следующие этапы :

1) подача заявки на сертификацию;

2) рассмотрение и принятие решения по заявке;

3) проведение необходимых проверок( анализ документов, испытание, проверка и т.п.);

4) анализ полученных результатов и принятие решения о возможности выдачи сертификата соответствия;

5) выдача сертификата и лицензии (разрешения) на применение знака соответствия;

6) инспекционный контроль за сертифицированным объектом в соответствии со схемой сертификации.

Схема сертификации – определенная совокупность действий, официально принимаемая в качестве доказательства соответствия продукции заданным требованиям.

Под **сертификацией** понимается действие третьей стороны, доказывающее, что обеспечивается необходимая уверенность в том, что должным образом идентифицированная продукция, процесс или услуга соответствует конкретному стандарту или другому нормативному документу.

Цели сертификации ПС

Защита интересов пользователей

Выдача сертификата

Контроль качества

Обеспечение высоких потребительских свойств

Повышение эффективности затрат

Полнота, точность эталонных данных

Адекватные показатели качества ПС

Методологии интерпретации данных

формальная

основная

Решение о выдаче сертификата на ПС основывается на оценке степени его соответствия действующим и/или специально разработанным документам.

Решение о выдаче сертификата на ПС основывается на оценке степени его соответствия действующим и/или специально разработанным документам:

В Законе "О сертификации продукции и услуг" определены два вида сертификации: обязательная и добровольная.

Обязательной сертификации подлежит продукция, включенная в отраслевые перечни, определяемые соответствующими нормативными документами. В соответствии с законодательством обязательной сертификации подлежит используемое программное обеспечение и базы данных программно-аппаратных комплексов, обеспечивающее защиту государственных информационных ресурсов и конфиденциальность информации, составляющей государственную тайну.

Объектами, подлежащими добровольной сертификации, являются:

* сертификация программного обеспечения средств измерений как автономного, так и встроенного;
* сертификация программного обеспечения измерительных, информационно-измерительных и информационных систем;
* сертификация программного обеспечения контроллеров и вычислительных блоков;
* сертификация программного обеспечения систем управления, в том числе автоматизированных систем управления, функционирующих с использованием измерительного оборудования или элементов измерительных систем;
* сертификация программного обеспечения тренажеров и иных имитационных систем;
* сертификация программного обеспечения, используемого для моделирования технологических процессов, математического и иного моделирования;
* сертификация программного обеспечения для передачи, хранения, актуализации, защиты, обеспечения доступа и использования измерительной, вычислительной и иной информации;
* сертификация программного обеспечения баз данных;
* сертификация программного обеспечения устройств с измерительными функциями, в том числе игровых автоматов, включая аттракционы и игровые автоматы с денежным выигрышем, тотализаторов, виртуальных игр, платежных терминалов, а также лотерейного оборудования и т.п.;
* сертификация аппаратно-программных комплексов, представляющих собой нераздельную совокупность технических и программных средств, осуществляющих автоматизированное выполнение поставленных задач и/или обеспечивающих функционирование электронных информационных ресурсов информационных систем.

**Сертификация**

**Обязательная**

**Добровольная**

Программные средства, выполненяющие особо ответственные функции, в которых недостаточное качество, ошибки или отказы могут нанести большой ущерб или опасны для жизни и здоровья людей (авиация, атомная энергетика, системы управления органами власти, банковские системы…)

Для удостоверения качества ПС с целью повышения их конкурентоспособности, расширения сферы использования и получения дополнительных экономических преимуществ. Таким сертификационным испытаниям подвергаются компоненты ОС и ПП широкого применения, повышение качества которых выгодно как для поставщиков, так и для пользователей ПС

Необходимость проведения тестирования программного обеспечения средств измерений на соответствие требованиям настоятельно подчеркивается в таких международных рекомендациях, как WELMEC 7.2 "Руководство по программному обеспечению", и в ГОСТ Р 8.596 "ГСИ. Метрологическое обеспечение измерительных систем. Основные положения"

Сертификация программного обеспечения предназначена для производителей программного обеспечения и представляет собой проверку ПО с целью оценки степени пригодности к автоматизации процессов управления информационными ресурсами и включает следующие работы:

* Проверка по функциональным критериям функциональных возможностей ПО (не проверяются: удобство использования, быстродействие, надежность защитных механизмов, и т.п.)
* Оценка состава документации;
* Оценка объема программирования, необходимого для настройки продукта по требованиям технологических процессов;
* Конфигурирование продукта для его проверки выполняется производителем ПО, за счет настройки профессионалом для обеспечения объективности проверки.
* Проверка по любому количеству процессов;
* Подробный отчет, включая детали реализации.

**Организации, осуществляющие сертификацию программного обеспечения**

ГОССТАНДАРТ РФ – государственный орган по сертификации:

* организует ведение обязательной сертификации продукции;
* организует и финансирует разработку;
* утверждает основополагающие системы сертификации

Ведомственные органы по управлению сертификацией продукции:

* Те же функции, но в ограниченном объеме и для конкретных классов продукции

Испытательные лаборатории сертификации (ИЛС):

* проводят испытания согласно действующим государственным нормативным документам;
* испытывают ПС по поручению органов госнадзора России, заказчиков или разработчиков ПС;
* оформляют в установленном порядке протоколы испытаний

В соответствии с Законом РФ "О техническом регулировании" Федеральным агенством по промышленности созданы и зарегистрированы организации осуществляющих сертификацию программного обеспечения с целью установления отсутствия в нем недекларированных возможностей:

* ФГУП ВНИИМС "Государственный испытательный сертификационный центр программных средств вычислительной техники" , - орган по сертификации в системе сертификации ГОСТ Р и РОСИНТЕХСЕРТ (регистрационный номер РОСС RU.В317.04И300 от 16 мая 2006 г.);
* "Евро-азиатская ассоциация производителей товаров и услуг в области безопасности" - орган по сертификации в системе сертификации АЙТИСЕРТИФИКА N РОСС.RU.М089ИТ00;
* ОАО "Институт проблем конфиденциальности информации" - орган по сертификации в системе сертификации РОСИНТЕХСЕРТ N РОСС.RU.В263.04РГ00).
* Автономная некоммерческая организация "Межрегиональный испытательный центр" (АНО "МИЦ").

В качестве испытательных лабораторий системы в настоящее время аккредитованы такие Центры стандартизации, метрологии и сертификации (ЦСМС), как Томский, Пермский, Татарстанский, Республики Башкортостан, Ульяновский, Пензенский и Краснодарский, а также ряд отраслевых организаций таких, как ООО Отраслевой метрологический центр (ОМЦ) "Газметрология", ООО Метрологический центр "Контрольно - измерительные технологии" (ООО "МЦ КИТ"), ФГУП Государственный научно-исследовательский институт гражданской авиации (ФГУП ГосНИИ ГА).

Проводится сертификация и выпускаются сертификаты соответствия ГОСТ Р на следующее программное обеспечение:

* операционные системы и средства их расширения,
* системы программирования и утилиты,
* сетевое программное обеспечение,
* системы управления базами данных (СУБД),
* редакционно-издательское программное обеспечение,
* программное обеспечение для деловой и презентационной графики,
* электронные таблицы,
* программное обеспечения для систем «электронных сделок»,
* программное обеспечения для обработки документов,
* программное обеспечения для систем искусственного интеллекта,
* программное обеспечения для моделирования,
* программное обеспечения для научных исследований,
* системы автоматизированного проектирования (САПР),
* программное обеспечения для технологической подготовки производства,
* автоматизированные системы управления технологическими процессами (АСУ ТП),
* программное обеспечения для автоматизации управления отраслями,
* педагогическое программное обеспечение,
* игры, лотереи, аукционы, средства развлечения и пр.,
* базы данных и информационно-справочные системы,
* электронные архивы,
* электронные издания,
* мультимедиа-приложения.

В результате сертификации ГОСТ Р подтверждаются показатели надежности, эффективности, сопровождаемости, мобильности, корректности и заявленные потребительские свойства программного обеспечения. При сертификации программного обеспечения подтверждается совместимость с различными операционными системами, устойчивость работы, возможность восстановления после различных сбоев, таких как отключение электропитания, программные сбои, потеря сети и пр. При сертификации программного обеспечения подтверждаются требования, заявленные в Технических условиях на программное обеспечение. При сертификации ГОСТ Р также проводится оценка качества сопроводительной документации.

**Сертификационные испытания программных средств**

Сертификация программного обеспечения представляет собой процесс испытаний программ в нагруженных режимах применения, подтверждающий соответствие показателей качества программного изделия требованиям установленным в нормативно-технических документах на него и обеспечивающий документальную гарантию использования программного средства при соблюдении заданных ограничений.

Сертификация программного обеспечения возможна при выполнении следующих условий:

* разработке шкалы показателей качества с учетом специфики целевого использования программных средств и набора их функциональных характеристик;
* каталогизации программ, как объекта сертификации на основе опыта их эксплуатации;
* создании специализированного центра сертификации, выполняющего роль "третейской" организации контроля качества;
* разработке нормативно-технической базы, регламентирующей сертификацию программного обеспечения;
* разработке эталонов программных средств, которые удовлетворяют требованиям технических заданий на разработку разнотипных программных комплексов;
* разработке специальной технологии испытаний, определяющей объем и содержание сертификационных испытаний;
* реализации комплекса тестового программного обеспечения для широкого спектра программных изделий.

В процессе сертификации сложного ПО следует выделить два аспекта: методический и технологический. Методический аспект связан с разработкой комплекса методик сертификации программного обеспечения с учетом специфики его применения, а технологический с автоматизацией процесса применения методического аппарата.

Следует отметить, что по некоторым оценкам до 70% общих затрат на создание и внедрение сложных программных комплексов приходится на реализацию процесса их сертификации. Причем значительная доля этих затрат относится к организации аппаратно-программной платформы, моделирующих средств и тестового обеспечения стенда сертификации.

Кроме того, важнейшим вопросом создания качественных программных изделий является обеспечение технологической безопасности ПО на этапе сертификационных стендовых испытаний. Недостаточный уровень развития современных информационных технологий разработки ПО, доминирующее использование зарубежных инструментальных средств и применение разработчиками программ лишь средств защиты от непреднамеренных дефектов обуславливают существенные, принципиально новые изменения технологии создания программ в этих условиях. Поэтому, одной из задач сертификации на современном уровне развития информационных технологий становится выявление преднамеренных программных дефектов.

Технологическая безопасность на этапе сертификационных испытаний характеризуется усилением мер контроля, так как в настоящее время предполагается, что вероятность внедрения закладок на окончательных фазах разработки программ выше, чем на начальных фазах в связи со снижением вероятности их обнаружения при последовательном технологическом контроле. В связи с этим завершающей процедурой тестового контроля и испытаний программ должна стать сертификация ПО по требованиям безопасности с выпуском сертификата на соответствие этого изделия требованиям технического задания.

В условиях существующих технологий создания ПО сертификация программ является наиболее дешевым и быстро реализуемым способом "фильтрации" компьютерных систем от низкокачественных, не отвечающих условиям безопасности программных средств.

Сертификационные испытания программных средств, в том числе защищенных программных средств и программных средств контроля защищенности проводятся в государственных и отраслевых сертификационных центрах.

Право на проведение сертификационных испытаний защищенных средств вычислительной техники, в том числе программных средств предоставляется Гостехкомиссией России по согласованию с Госстандартом России предприятиям-разработчикам защищенных СВТ, специализированным организациям ведомств, разрабатывающих защищенные СВТ, в том числе программные средства.

В соответствии с "Положением о сертификации..." по результатам сертификационных испытаний оформляется акт, а разработчику выдается сертификат, заверенный в Гостехкомиссии России и дающий право на использование и распространение этих средств как защищенных.

Разработанные программные средства после их приемки представляются для регистрации в специализированный фонд Государственного фонда алгоритмов и программ.

**Приемка и эксплуатация программного обеспечения**

Установка ПО осуществляется разработчиком в соответствии с планом в той среде и на том оборудовании, которые предусмотрены договором. В процессе установки проверяется работоспособность ПО и баз данных. Если устанавливаемое программное обеспечение заменяет существующую систему, разработчик должен обеспечить их параллельное функционирование в соответствии с договором.

Приемка ПО предусматривает оценку результатов квалификационного тестирования ПО и системы и документирование результатов оценки, которые проводятся заказчиком с помощью разработчика. Разработчик выполняет окончательную передачу ПО заказчику в соответствии с договором, обеспечивая при этом необходимое обучение и поддержку.

Процесс эксплуатации охватывает действия и задачи оператора - организации, эксплуатирующей систему и включает действия:

Управление конфигурацией позволяет организовать, систематически учитывать и контролировать внесение изменений в ПО на всех стадиях его жизненного цикла. Общие принципы и рекомендации по управлению конфигурацией ПО отражены в проекте стандарта ISO/IEC 12207-2: 1995 “Information Technology - Software Life Cycle Processes. Part2. Configuration Management for Software”.

Процесс обеспечения качества обеспечивает соответствующие гарантии того, что ПО и процессы его жизненные циклы соответствуют заданным требованиям и утвержденным планам. Под качеством ПО понимается совокупность свойств, которые характеризуют способность ПО удовлетворять заданным требованиям.

Для получения достоверных оценок создаваемого ПО процесс обеспечения его качества должен происходить независимо от субъектов, непосредственно связанных с разработкой ПО. При этом могут использоваться результаты других вспомогательных процессов, таких, как верификация, аттестация, совместная оценка, аудит и разрешение проблем. Процесс обеспечения качества программного обеспечения включает:

· подготовительная работа (заключается в координации с другими вспомогательными процессами и планировании самого процесса обеспечения качества с учетом используемых стандартов, методов, процедур и средств);

· обеспечение качества продукта подразумевает гарантирование полного соответствия программных продуктов и их документации требованиям заказчика, предусмотренным в договоре;

· обеспечение качества процесса предполагает гарантирование соответствия процессов жизненного цикла ПО, методов разработки, среды разработки и квалификации персонала условиям договора, установленным стандартам и процедурам;

· обеспечение прочих показателей качества системы осуществляется в соответствии с условиями договора и стандартом ISO 9001.

Процесс **верификации** состоит в определении того, что программные продукты, являющиеся результатами некоторого действия, полностью удовлетворяют требованиям или условиям, обусловленным предшествующими действиями (верификация в узком смысле означает формальное доказательство правильности ПО). В процесс верификации проверяются следующие условия:

* непротиворечивость требований к системе и степень учета потребностей пользователей;
* возможности поставщика выполнять заданные требования;
* соответствие выбранных процессов жизненного цикла программного обеспечения условиям договора;
* адекватность стандартов, процедур и среды разработки процесса ПО;
* соответствие проектных спецификаций ПО заданным требованиям;
* корректность описания в проектных спецификациях входных и выходных данных, последовательности событий, интерфейсов, логики;
* соответствие кода проектным спецификациям и требованиям;
* тестируемость и корректность кода, его соответствие принятым стандартам кодирования;
* корректность интеграции компонентов ПО в систему;
* адекватность, полнота и непротиворечивость документации.

**Аттестация**, так же как и верификация, может осуществляться с различными степенями независимости. Если процесс аттестации выполняется организацией, не зависящей от поставщика, разработчика, оператора или службы сопровождения, то он называется процессом независимой аттестации. Процесс аттестации предусматривает определение полноты соответствия заданных требований и созданной системы или программного продукта их конечному функциональному назначению. Под аттестацией обычно понимается подтверждение и оценка достоверности проведенного тестирования. Аттестация должно гарантировать полное соответствие ПО спецификациям, требованиям и документации, а также возможность его безопасного и надежного применения пользователем. Аттестацию рекомендуется выполнять путем тестирования во всех возможных ситуациях и использовать при этом независимых специалистов. Аттестация может проводиться на начальных стадиях жизненного цикла ПО или как часть работы по приемке ПО.

**Процесс совместной оценки** предназначен для оценки состояния работ по проекту и ПО. Он сосредоточен в основном на контроле планирования и управления ресурсами, персоналом, аппаратурой и инструментальными средствами проекта. Оценка применяется как на уровне управления проектом, так и на уровне технической реализации проекта и проводится в течение всего срока договора. Данный процесс может выполняться двумя любыми сторонами, участвующими в договоре, при этом одна сторона проверяет другую. Процесс совместной оценки включает действия:

* подготовительную работу;
* оценку управления проектом;
* техническую оценку.

**Аудит** - это ревизия (проверка), проводимая компетентным органом (лицом) в целях обеспечения независимой оценки степени соответствия ПО или процессов установленным требованиям. Аудит служит для установления соответствия реальных работ и отчетов требованиям, планам и контракту. Аудиторы не должны иметь прямой зависимости от разработчиков ПО. Они определяют состояние работ, использование ресурсов, соответствие документации требованиям и стандартам, корректность тестирования. Процесс аудита представляет собой определение соответствия требованиям, планам и условиям договора. Аудит может выполняться двумя любыми сторонами, участвующими в договоре, когда одна сторона проверяет другую.

**Процесс разрешения проблем** предусматривает анализ и решение проблем (включая обнаруженные несоответствия) независимо от их происхождения или источника, которые обнаружены в ходе разработки, эксплуатации, сопровождения или других процессов. Каждая обнаруженная проблема должна быть идентифицирована, описана, проанализирована и разрешена.

**Заключение**

В заключении хотелось бы отметить какой эффект дает сертификация программного обеспечения для разработчиков, заказчика и конечных пользователей программных средств.

Благодаря тестированию, экспертизам, различным видам испытаний и соответствующим доработкам программного обеспечения пользователь получает программный продукт, а не полуфабрикат. По эксплуатационной документации пользователь легко может установить программное обеспечение и быстро освоить приемы работы с ним. Ошибки, выявленные испытательным центром при тестировании и испытаниях, позволяют разработчику повысить надежность и качество программного обеспечения.

Таким образом, заказчик получает уверенность в том, что в его распоряжении находится качественный, законченный программный продукт, который соответствует потребностям, уверенно сопровождается и в случае необходимости может быть легко и без потерь восстановлен.

# Интеллектуальная собственность и авторские права

Любая программа для ЭВМ с точки зрения закона представляет собой определённый программный код, набранный известными символами. Поэтому программы для ЭВМ охраняются как литературные произведения (п.1 ст. 1259 Гражданского кодекса Российской Федерации).

Для начала следует понять, что именно может охраняться авторским правом. К примеру, у Вас есть задумка создать программу для ЭВМ. В своём сознании Вы представляете её в деталях, постоянно совершенствуете. До тех пор, пока это произведение находится «в Вашем сознании», авторское право не возникает.

И лишь в тот момент, когда своей задумке Вы придали законченную объективную форму — записали соответствующий программный код, Вы становитесь Автором, возникает Результат интеллектуальной деятельности, и у него появляется авторско-правовая защита.

К примеру, Вы рассказали идею об интересной программе случайному прохожему, и он придал идее объективную форму раньше Вас. В этом случае он будет являться автором. Поэтому не стоит торопиться делиться блестящими идеями с окружающими.

В силу прямого указания закона (п. 5 ст. 1259 ГК РФ) авторские права не распространяются на идеи, концепции, открытия, факты, языки программирования.

Но вот произведение создано — программа для ЭВМ мелькает с экрана монитора. Авторское право возникло. Что дальше, что делать, как получать заслуженное вознаграждение и уважение других к Вашей интеллектуальной собственности.

Сразу оговоримся, в силу прямого указания п.4 ст. 1259 Гражданского кодекса «Для возникновения, осуществления и защиты авторских прав не требуется регистрация произведения или соблюдение каких-либо иных формальностей». Статья 1262 Гражданского кодекса Российской Федерации предоставляет правообладателю программы для ЭВМ право зарегистрировать такую программу в федеральном органе исполнительной власти по интеллектуальной собственности. Такая регистрация является добровольной. Её отсутствие не влечёт опровержение презумпции Вашего авторства.

Формально Вы — Автор, говорит Закон, но жизнь так устроена, что с документом, подтверждающим Ваше авторство, живётся спокойнее. Необходимость в таких бумажках, как правило, возникает в самый неподходящий момент. Принципиальным правилом является возникновение авторского права в силу факта создания произведения без регистрации авторского права или соблюдения каких-либо иных формальностей. Такой порядок возникновения авторских прав предусмотрен Бернской конвенцией и применяется повсеместно. Тем не мене, некоторые страны обходят это положение, предусматривая регистрацию авторских прав, которая формально не имеет правоустанавливающего значения, но на практике авторское право без нее мало стоит. Так, в США с нарушителя авторских прав на незарегистрированное произведение нельзя взыскать денежную компенсацию.

Наше правительство также выступило инициатором создания системы международной регистрации авторских прав, но такие предложения не принимаются мировым сообществом в короткие сроки. На сегодняшний день в России осуществляется добровольная регистрация в Федеральной службе по интеллектуальной собственности авторских прав программ для ЭВМ и базы данных, которая имеет только доказательственное значение. Таким образом, сегодня отсутствует какой-либо государственный орган, который сможет обеспечить публичную достоверность информации о Вашем авторстве на программу для ЭВМ.

Следует начать с того, что на каждом экземпляре произведения должно быть указание на Ваше авторство — указание имени или творческого псевдонима.

Регистрации программы для ЭВМ в Федеральной службе по интеллектуальной собственности. Для этого необходимо уплатить государственную пошлину, экземпляр программы в электронном виде, распечатать программу не более чем на 70 листах, написать анкету с данными о заявителе, реферат (описание программы).

Необходимость в подтверждении своих прав возникает в самые сложные моменты. Последний совет: сохраняйте наброски, черновики исходные файлы с Вашими работами — все это может сыграть роль при доказывании Вашего авторства.

Если же авторские права были нарушены, следует обратиться к юристу, специализирующимся в области авторского права, который даст профессиональную оценку данной ситуации и при положительной перспективе дела будет представлять Ваши интересы в суде.

# Использованная литература

# Стандарты

1. ГОСТ 28195 – 89. Оценка качества программных средств. Общие положения. // М.: Издательство стандартов, 1989
2. ГОСТ 19.102-77. ЕСПД. Стадии разработки.
3. ГОСТ 19.001-77 ЕСПД. Общие положения.
4. ГОСТ 19.101-77 ЕСПД. Виды программ и программных документов.
5. ГОСТ 19.102-77 ЕСПД. Стадии разработки.
6. ГОСТ 19.105-78 ЕСПД. Общие требования к программным документам.
7. ГОСТ 19.106-78 ЕСПД. Требования к программным документам, выполненным печатным способом.
8. ГОСТ 19.201-78 ЕСПД. Техническое задание. Требования к содержанию и оформлению.
9. ГОСТ 19.202-78 ЕСПД. Спецификация. Требования к содержанию и оформлению.
10. ГОСТ 19.301-79 ЕСПД. Порядок и методика испытаний.
11. ГОСТ 19.401-78 ЕСПД. Текст программы. Требования к содержанию и оформлению.
12. ГОСТ 19.402-78 ЕСПД. Описание программы.
13. ГОСТ 19.404-79 ЕСПД. Пояснительная записка. Требования к содержанию и оформлению.
14. ГОСТ 19.504-79 ЕСПД. Руководство программиста.
15. ГОСТ 19.505-79 ЕСПД. Руководство оператора.
16. ГОСТ 19.701-90 ЕСПД. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения.
17. ГОСТ 19.781-90. Обеспечение систем обработки информации программное.
18. ГОСТ Р ИСО/МЭК 12207. Процессы жизненного цикла программных средств.
19. ГОСТ Р ИСО/МЭК 9126-93 Информационная технология. Оценка программной продукции. Характеристики качества и руководства по их применению.

# Книги

1. Ковалевская Е.В. Метрология, качество и сертификация программного обеспечения. // М.,МЭСИ,2008
2. В.В. Липаев Сертификация программных средств // М., Синтег, 2010
3. В.А. Благодатских, В.А. Волнин, К.Ф. Поскакалов Стандартизация разработки программных средств. Учебное пособие. // М., Финансы и статистика, 2006
4. Л.В.Брянский, А.С.Дойников Краткий справочник метролога. // М., Издательство стандартов, 1991

# Интернет ресурсы

1. <http://www.intuit.ru/studies/courses/64/64/lecture/1874?page=1>
2. <http://tdocs.su>
3. <http://www.raai.org/about/persons/karpov/pages/ofdoc/ofdoc.html>
4. <http://www.philosoft.ru/users-manual.zhtml>