1. **ГОСТ Р ИСО/МЭК 12207. Основные процессы и взаимосвязь между документами в информационной системе согласно стандартам.**

**ГОСТ Р ИСО/МЭК 12207. Основные процессы и взаимосвязь между документами в информационной системе согласно стандартам.**

ISO (International Organization for Standardization) –это система мировых стандартов качества, разработанных Международной организацией по стандартизации. Она включает семейство нормативов, которые разрабатываются для разных областей деятельности.

Российские аналоги стандартов *ISO*, т. е. ГОСТы, в названии которых присутствует сочетание "Р ИСО", представляют собой *аутентичные* официальные переводы стандартов *ISO* на русский язык.

Одно из центральных мест занимает стандарт

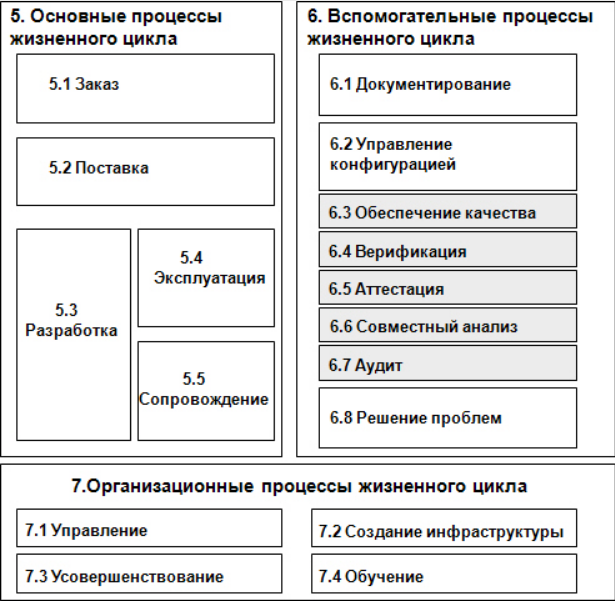
**ГОСТ Р ИСО/МЭК 12207 "Процессы жизненного цикла программных систем"** (ГОСТ 12207, 1999) - один из самых известных и распространенных процессно-ориентированных стандартов в области управления ИТ. Ссылки на него встречаются практически во всех работах и методиках, относящихся к процессам управления ИТ.

**Методологическая основа ГОСТ Р ИСО/МЭК 12207 - разбиение процессов на группы, которых в стандарте вводится три.**

**Основные.** Это процессы, непосредственно относящиеся к *жизненному циклу информационной системы*. Можно считать, что это производственные процессы организации.

**Вспомогательные.** Это процессы, предназначенные для поддержки основных процессов. Сами по себе эти процессы организации не нужны.

**Организационные.** Это общекорпоративные процессы. Эти процессы существуют в организации независимо от того, как организовано производство и как устроены вспомогательные процессы.

**Структура процессов жизненного цикла программных систем по ГОСТ Р ИСО/МЭК 12207**

***Процессы,***Стандарт предусматривает процессы, состоящие из работ, а работы — из задач. Порядок работ и задач в стандарте не строгий, важно соблюдать логические связи между ними.

**Цель стандарта** — установить полный набор процессов, которые могут применяться в рамках проекта по разработке программной системы. Однако, учитывая различия между проектами (например, масштаб, сложность, риски и т. д.), разрешается локально корректировать процессы для каждого проекта, исключая или добавляя отдельные задачи и работы. Этот процесс называется адаптацией в рамках стандарта.

1. **Виды внедрения, план внедрения.**

**Виды внедрения информационных систем**

**Комплексная автоматизация**.

Разработка ИС производится для обеспечения максимального охвата средствами автоматизации всех участков обработки информации и выполняется в сжатые сроки. Наиболее предпочтительный, но и самый дорогостоящий вариант.

**Поэтапная автоматизация.**

втоматизация деятельности отдельных участков с возможностью их дальнейшего объединения в единую систему. Позволяет распределить финансовые затраты во времени и постепенно приспособить производственный цикл к требованиям АСУ.

**Процесс внедрения стратегии организации включает ряд этапов:**

1. **Подготовительный этап**
2. **Внедрение стратегии**
3. **Этап завершения реализации основных этапов стратегии**

**Подготовительный этап:**

* Анализ существующей системы менеджмента компании, определе­ние ее соответствия внедряемой стратегии. Разработка мер по устранению выявленных дефектов.
* Решение проблемы мотивации менеджеров и специалистов, внедряю­щих стратегию.
* Разработка системы управления внедрением стратегии.
* Разработка системы стратегического контроля.
* Подбор управленческого персонала и принятие решений о расстанов­ке менеджеров, связанных с внедрением стратегии.
* Формирование рабочих групп и команд по решению проблем внедре­ния стратегии.
* Изменение организационной структуры и приведение ее в соответ­ствие с новой стратегией.
* Подготовка персонала к внедрению стратегии, определение необ­ходимых изменений в организационной культуре, прежде всего, обуче­ние и повышение квалификации персонала для работы в изменившихся условиях, развитие обучения и самообучения, трансфера (распростра­нения и передачи знаний), применение новых систем мотивации и сти­мулирования.
* Разработка плана внедрения (этапы внедрения, результаты) и форми­рование временных графиков внедрения стратегии, распределение функций власти и ответственности между конкретными менеджерами, участвующи­ми во внедрении стратегий.

1. **Стратегии, цели и сценарии внедрения.**

**Внедрение стратегии:**

* Проведение рабочих совещаний групп и команд по внедрению стра­тегии.
* Установление прямых коммуникаций и взаимодействия между работника­ми, внедряющими стратегию, в том числе по каналам телекоммуникаций.
* Финансирование внедрения стратегии, рабочих групп и команд.
* Контроль результатов внедрения стратегии.
* Определение экономической эффективности и результатов стратегии. Сравнение их с планом.
* Проведение рабочих совещаний групп и команд по внедрению стра­тегии.
* Установление прямых коммуникаций и взаимодействия между работника­ми, внедряющими стратегию, в том числе по каналам телекоммуникаций.
* Финансирование внедрения стратегии, рабочих групп и команд.
* Контроль результатов внедрения стратегии.
* Определение экономической эффективности и результатов стратегии. Сравнение их с планом.

**Этап завершения реализации основных этапов стратегии:**

* Оценка общих результатов внедрения стратегии в плановых периодах (месяцы, кварталы, годы).
* Определение причин отклонений и принятие корректирующих ре­шений.
* Анализ перспектив развития стратегических процессов

**Управление в процессе реализации стратегии проблем значительно уп­рощается, если используются механизмы политик, процедур и правил.**

**Политика** определяет общие ориентиры действий.

**Процедуры** указыва­ют обязательную последовательность действий при возникновении проблем.

**Правила** определяют обязательные действия менеджеров в условиях стан­дартных и часто возникающих ситуаций.

**виды контроля:**

**административный контроль** выполнения управленческих решений и регламентов, соблюдения правовых норм и законодательства, расстановки кадров, выполнения планов и заданий, взаимодействия организации с ок­ружающей средой (экологический контроль, соблюдения прав потребите­лей, этическое поведение по отношению к конкурентам);

**финансовый контроль**—учет и анализ расходования финансовых ресурсов, обеспечение финансовых потребностей организации. В современных систе­мах управления нередко принципы финансового аудита распространяются на сферу маркетинга, контроля качества, процессы производства;

**административный контроль** выполнения управленческих решений и регламентов, соблюдения правовых норм и законодательства, расстановки кадров, выполнения планов и заданий, взаимодействия организации с ок­ружающей средой (экологический контроль, соблюдения прав потребите­лей, этическое поведение по отношению к конкурентам);

**финансовый контроль**—учет и анализ расходования финансовых ресурсов, обеспечение финансовых потребностей организации. В современных систе­мах управления нередко принципы финансового аудита распространяются на сферу маркетинга, контроля качества, процессы производства;

**Профессиональные функции современного менеджера целесообразно разделить на следующие виды**

— базовые функции общего менеджмента

— интегральные функции менеджмента

— специальные (объектно-ориентированные) функции менеджмента.

**базовые функции** общего менеджмента, универсальные для всех менед­жеров и всех видов деятельности, независимо от подразделения, в котором менеджеры работают, образующие каркас содержания труда каждого менед­жера. К ним относятся планирование, организация, контроль, координа­ции и активизация персонала;

**интегральные функции** менеджмента. Эти функции образуют об­щую платформу для всех других функций, тесно связаны со всеми други­ми функциями и друг с другом**.**

**Специальные функции** менеджмента - функции, которые реализуются в связи со спецификой деятельности конкретной компании**.**

1. **Функции менеджера сопровождения и менеджера развертывания.**

***Соп­ро­вож­де­ние ПО*** — это про­цесс из­ме­нения прог­рам­мной сис­те­мы пос­ле ее пос­тавки за­каз­чи­ку.

На этом эта­пе ис­прав­ля­ют­ся нез­на­чительные ошиб­ки прог­рамми­рова­ния и  кор­ректи­ру­ют­ся от­дельные не­дора­бот­ки, не свя­зан­ные с из­ме­нени­ем об­щей ар­хи­тек­ту­ры сис­те­мы.

При этом ли­бо из­ме­ня­ют­ся су­щес­тву­ющие, ли­бо до­бав­ля­ют­ся но­вые ком­по­нен­ты ПО.

Соп­ро­вож­де­ние вы­пол­ня­ет­ся соп­ро­вож­да­ющей ор­га­низа­ци­ей. Со­дер­жа­ние действий по соп­ро­вож­де­нию пред­по­лага­ет вы­пол­не­ние под­го­тови­тельной ра­боты, в про­цес­се ко­торой раз­ра­баты­ва­ют­ся про­цеду­ры ло­кали­зации и ус­тра­нения проб­лем, воз­ни­ка­ющих при соп­ро­вож­де­нии. При по­ложи­тельном ре­шении вы­пол­ня­ет­ся мо­дифи­кация ПО, в про­цес­се ко­торой в прог­рам­мные ком­по­нен­ты и до­кумен­та­цию вно­сят­ся из­ме­нения, про­веря­ет­ся це­лост­ность мо­дифи­циру­емо­го ПО и ПО пе­рено­сит­ся в дру­гую сре­ду. Да­лее сле­ду­ет пе­ри­од па­рал­лельной экс­плу­ата­ции ПО в ста­рой и но­вой сре­дах, по окон­ча­нии ко­торо­го ста­рое ПО сни­ма­ет­ся с экс­плу­ата­ции и ар­хи­виру­ет­ся. Соп­ро­вож­де­ние сис­те­мы мо­жет вы­пол­няться ли­бо для ис­прав­ле­ния оши­бок, ли­бо для адап­та­ции, ли­бо в це­лях из­ме­нения фун­кци­ональных воз­можнос­тей сис­те­мы.

***Соп­ро­вож­де­ние в це­лях ис­прав­ле­ния оши­бок*** вы­пол­ня­ет­ся в слу­чае по­яв­ле­ния в ПО оши­бок в прог­рамми­рова­нии, ли­бо оши­бок про­ек­ти­рова­ния, ли­бо оши­бок в сис­темных тре­бова­ни­ях.

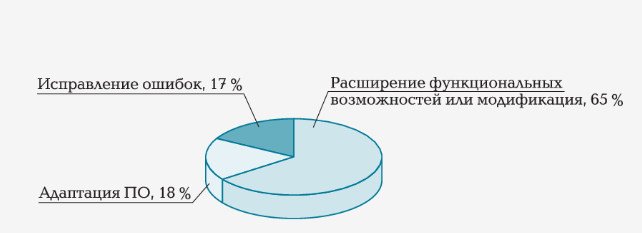
***Оши­бки в прог­рамми­рова­нии*** — дос­та­точ­но лег­ко ус­тра­нимые ошиб­ки.

Ис­прав­ле­ния ***оши­бок про­ек­ти­рова­ния*** об­хо­дят­ся весьма до­рого, пос­кольку тре­бу­ют кор­ректи­ров­ки и (или) пе­реп­рограм­ми­рова­ния от­дельных ком­по­нен­тов.

***Ошиб­ки в сис­темных тре­бова­ни­ях*** тре­бу­ют пе­реп­ро­ек­ти­рова­ния всей сис­те­мы и по­это­му яв­ля­ют­ся са­мыми до­рогос­то­ящи­ми.

***Соп­ро­вож­де­ние в це­лях адап­та­ции ПО к ус­ло­ви­ям экс­плу­ата­ции*** вы­пол­ня­ет­ся при из­ме­нении ого­ворен­ных в тех­ни­чес­ком за­дании ус­ло­вий и зак­лю­ча­ет­ся в до­пол­ни­тельной мо­дифи­кации ПО.

***Соп­ро­вож­де­ние в це­лях из­ме­нения фун­кци­ональных воз­можнос­тей сис­те­мы*** про­водит­ся, нап­ри­мер, для под­дер­жки ап­па­рат­ных средств, опе­раци­он­ной сис­те­мы и  ПО.



Со­дер­жа­ние про­цес­сов соп­ро­вож­де­ния за­висит от ти­па ПО, тех­но­логии его раз­ра­бот­ки, уров­ня спе­ци­алис­тов, ко­торые не­пос­редс­твен­но вов­ле­чены в про­цесс, и не­кото­рых дру­гих фак­то­ров.

Мож­но вы­делить сов­па­да­ющие фак­то­ры и эта­пы в этих про­цес­сах:

* ана­лиз из­ме­нений;
* пла­ниро­вание но­вой вер­сии ПО;
* ре­али­зация но­вой вер­сии сис­те­мы;
* пос­тавка сис­те­мы за­каз­чи­ку.
* Про­цесс мо­дер­ни­зации соп­ро­вож­да­ет­ся из­ме­нени­ем па­рамет­ров сис­те­мы, ар­хи­тек­ту­ры и собс­твен­но ПО, в ко­торое до­бав­ля­ют­ся но­вые ком­по­нен­ты.
* Пос­ле вне­сен­ных из­ме­нений вы­пол­ня­ет­ся пов­торное тес­ти­рова­ние ПО и про­водит­ся под­робный ана­лиз из­ме­нений, при ко­тором мо­гут вы­явиться те пос­ледс­твия мо­дер­ни­зации, ко­торые не бы­ли за­мече­ны при на­чальном ана­лизе из­ме­нений.

**Менеджер сопровождения** – это сотрудник, который замыкает на себе основные функции работы с клиентом и оказывает ему поддержку – от момента первого визита и взаимодействия с администратором и на протяжении всего периода использования клиентом услуг салона или центра.

Менеджер сопровождения старается как можно дольше продлить сотрудничество своей компании с клиентом, постоянно расширяя и углубляя диапазон услуг

Одна из задач менеджера сопровождения – повышение рентабельности обслуживания клиентов и повышение прибыли предприятия.

***Раз­верты­вание ПО***яв­ля­ет­ся частью ЖЦ ПО и под­ра­зуме­ва­ет все действия, ко­торые де­ла­ют прог­рам­мную сис­те­му го­товой к ис­пользо­ванию.

Раз­верты­вание мо­жет осу­щест­вляться в про­цес­се раз­ра­бот­ки ПО са­мим прог­раммис­том.

При пла­ниро­вании про­цес­са оп­ре­деля­ют­ся *кон­трольные от­метки* — ве­хи, от­ме­ча­ющие окон­ча­ние оп­ре­делен­но­го эта­па ра­бот.

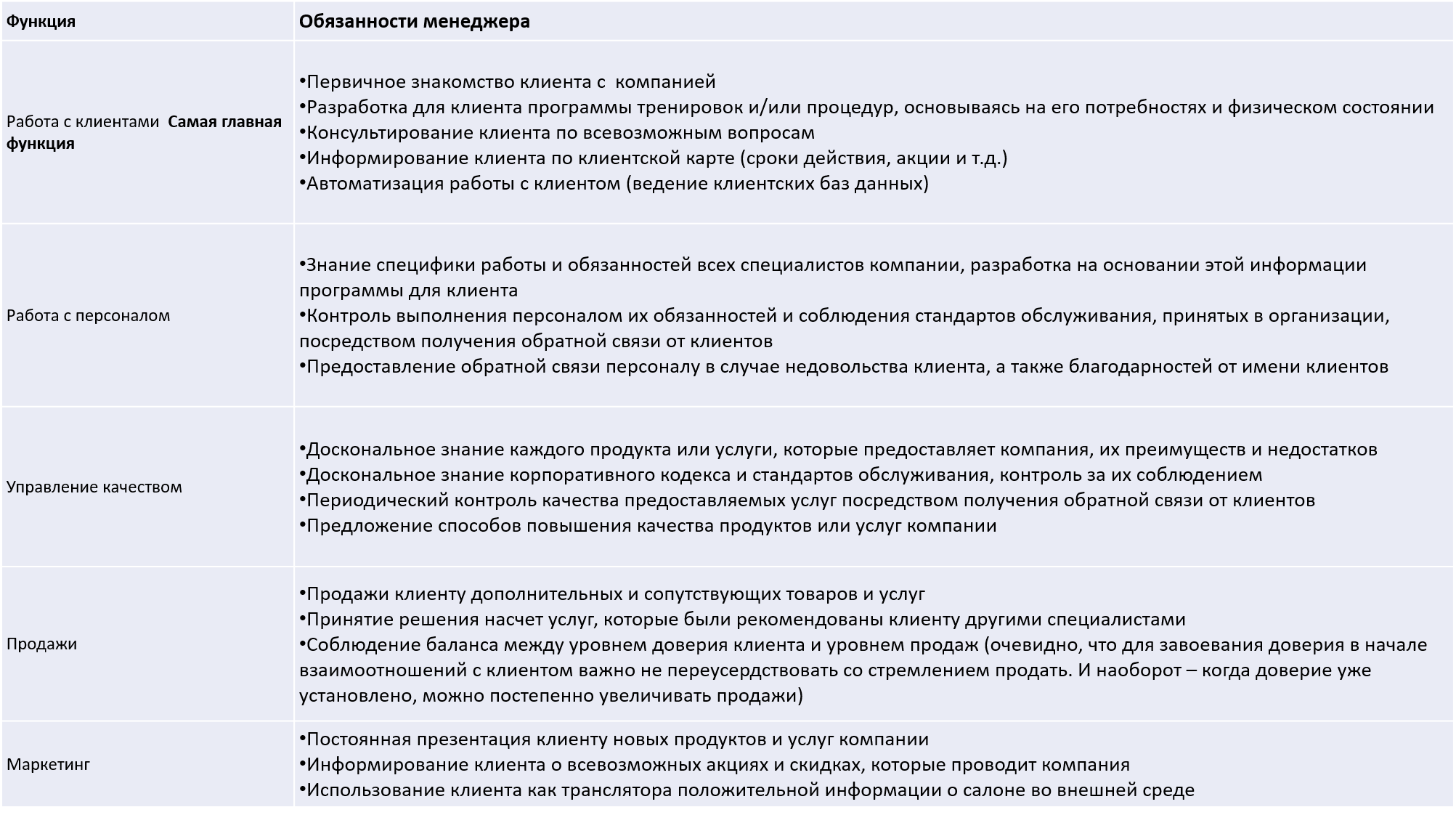
Для каж­дой кон­трольной от­метки соз­да­ет­ся от­чет, ко­торый пре­дос­тавля­ет­ся ру­ководс­тву про­ек­та.

Эти от­че­ты не дол­жны быть больши­ми объем­ны­ми до­кумен­та­ми; они дол­жны под­во­дить крат­кие ито­ги окон­ча­ния от­дельно­го ло­гичес­ки за­вер­шенно­го эта­па про­ек­та**.**

Раз­верты­вание ПО в ор­га­низа­ции яв­ля­ет­ся тру­до­ем­ким и кро­пот­ли­вым про­цес­сом, сос­то­ящим из ря­да вза­имос­вя­зан­ных эта­пов ко­торые на­зыва­ют­ся *кон­трольны­ми про­ек­тны­ми эле­мен­та­ми*.

* план раз­верты­вания ПО
* раз­ра­бот­ка спе­цифи­каций (мо­тиви­рован­ных спис­ков) тре­бова­ний
* ус­та­нов­ка/ти­ражи­рова­ние от­дельной под­систе­мы.

На це­левом пред­при­ятии весь этап раз­верты­вания раз­би­ва­ет­ся на ряд по­дэта­пов, за­вер­ша­ющих­ся пред­став­ле­ни­ем ад­ми­нис­тра­тору раз­верты­вания внут­ренних про­ек­тных ре­зульта­тов (кон­трольны­ми от­метка­ми).

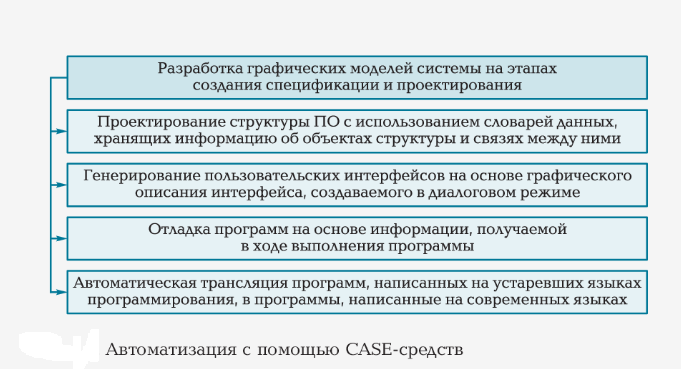


(к менеджеру сопровождения)

1. **Типовые функции инструментария для автоматизации процесса внедрения информационной системы.**

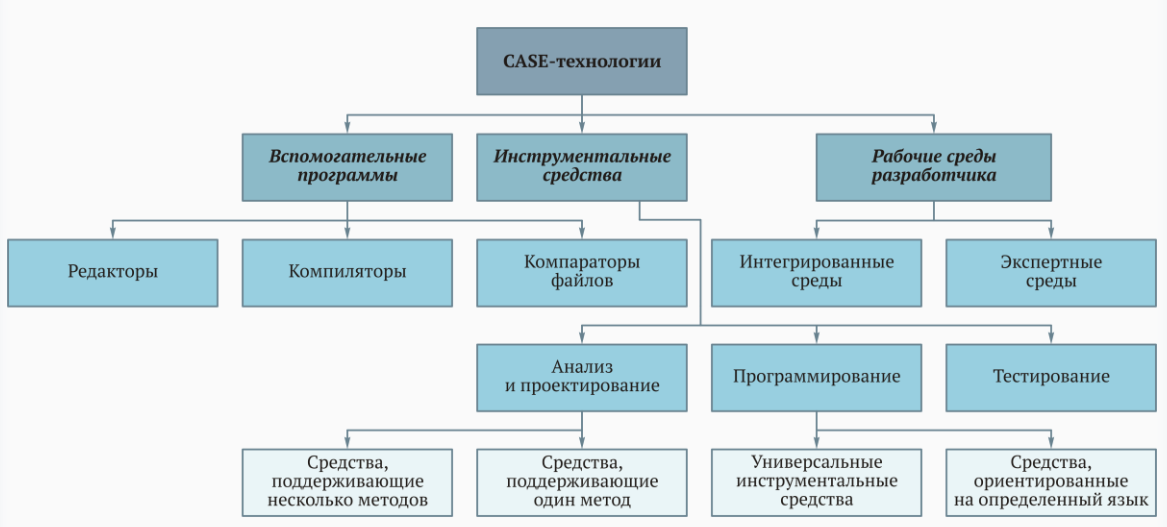
***Ав­то­мати­зиро­ван­ные средс­тва раз­ра­бот­ки ПО***— это спе­ци­альный тип ПО, пред­назна­чен­но­го для под­дер­жки про­цес­сов соз­да­ния са­мих прог­рам­мных средств, та­ких как раз­ра­бот­ка тре­бова­ний к ПО, про­ек­ти­рова­ние ПО, ко­диро­вание и тес­ти­рова­ние прог­рамм и т. д.

Тех­но­логия CASE пред­ла­га­ет под­дер­жку про­цес­са соз­да­ния ПО пу­тем ав­то­мати­зации по­дав­ля­юще­го большинс­тва эта­пов его раз­ра­бот­ки. При­меры CASE-про­цес­сов, или про­цес­сов, ко­торые мож­но ав­то­мати­зиро­вать с по­мощью CASE-средств



Кро­ме улуч­ше­ния ка­чес­тва соз­да­ва­емых прог­рамм при­мене­ние тех­но­логии CASE поз­во­ля­ет на по­рядок по­высить про­из­во­дительность тру­да раз­ра­бот­чи­ков ПО.

Од­на­ко дальнейшее рас­ши­рение при­мене­ния CASE-тех­но­логии ог­ра­ничи­ва­ет сле­ду­ющий фак­тор. На до­лю средств CASE при­ходит­ся ав­то­мати­зация не­ин­теллек­ту­альных про­цес­сов: тран­сля­ция и от­ладка ПО, ге­нера­ция ин­терфейсов и про­чих, по­казан­ных на следующем рисунке.



**Спи­сок клю­чевых по­нятий CASE-средств.**

***Про­цесс соз­да­ния ПО*** — это со­вокуп­ность про­цес­сов, вы­пол­ня­емых при его раз­ра­бот­ке.

***Мо­дели про­цес­са соз­да­ния ПО*** — абс­трак­тные пред­став­ле­ния этих про­цес­сов. Лю­бой про­цесс соз­да­ния ПО вклю­ча­ет в се­бя эта­пы раз­ра­бот­ки сис­темной спе­цифи­кации, про­ек­ти­рова­ния и ре­али­зации, ат­теста­ции и мо­дер­ни­зации ПО.

***Обоб­щенные мо­дели соз­да­ния ПО*** опи­сыва­ют ор­га­низа­цию про­цес­са раз­ра­бот­ки про­грам­мных сис­тем. К та­ким мо­делям от­но­сят­ся кас­кадная мо­дель, эво­люци­он­ная мо­дель раз­ра­бот­ки, мо­дель фор­мальной раз­ра­бот­ки сис­тем и мо­дель раз­ра­бот­ки ПО на ос­но­ве ра­нее соз­данных ком­по­нен­тов.

***Оп­ре­деле­ние тре­бова­ний*** — это про­цесс раз­ра­бот­ки сис­темной спе­цифи­кации.

***Про­ек­ти­рова­ние и ре­али­зация*** — это про­цес­сы пре­об­ра­зова­ния сис­темной спе­цифи­кации в сис­те­му ис­полня­емых прог­рамм.

***Ат­теста­ция ПО*** — это про­цесс про­вер­ки со­от­ветс­твия раз­ра­ботан­ной сис­те­мы ее спе­цифи­кации и пот­ребнос­тям пользо­вате­лей.

***Эво­люция ПО*** — это мо­дер­ни­зация су­щес­тву­ющих прог­рам­мных сис­тем в со­от­ветс­твии с но­выми тре­бова­ни­ями.

В нас­то­ящий пе­ри­од на­ибо­лее раз­ви­тым ком­плек­сом CASE-средств в Рос­сии яв­ля­ет­ся ком­плекс тех­но­логий и инс­тру­мен­тальных средств соз­да­ния ин­тегральных средств, ос­но­ван­ный на ме­тодо­логии и тех­но­логии DATARUN.

**В сос­тав ком­плек­са вхо­дят сле­ду­ющие инс­тру­мен­тальные средс­тва:**

* CASE-средс­тво Silverrun и средс­тво раз­ра­бот­ки при­ложе­ний JAM;
* сог­ла­су­ющий мост Silverrun JAM;
* ком­плекс средств тес­ти­рова­ния и ме­нед­жер тран­закций;
* ком­плекс средств пла­ниро­вания и уп­равле­ния про­ек­том;
* ком­плекс средств кон­фи­гура­ци­он­но­го уп­равле­ния;
* CASE-средс­тво про­ек­ти­рова­ния и раз­ра­бот­ки ПО Rational Rose;
* средс­тво до­кумен­ти­рова­ния.

1. **Оценка качества функционирования информационной системы.**

Сог­ласно стан­дарту ISO/IEC 25000:2014 «Про­ек­ти­рова­ние сис­тем и раз­ра­бот­ка про­грам­мно­го обес­пе­чения. Тре­бова­ния к ка­чес­тву сис­тем и прог­рам­мно­го обес­пе­чения и их оцен­ка . Ру­ководс­тво» **под ка­чес­твом ПО по­нима­ет­ся его спо­соб­ность при за­дан­ных ус­ло­ви­ях удов­летво­рять ус­та­нов­ленным или пред­по­лага­емым пот­ребнос­тям.**

***Фун­кци­ональность***- ха­рак­те­рис­ти­ка от­ве­ча­ет за то, что ПО ра­бота­ет ис­прав­но и точ­но, фун­кци­онально сов­мести­мо, со­от­ветс­тву­ет стан­дартам от­расли и за­щище­но от не­сан­кци­они­рован­но­го дос­ту­па.

***На­деж­ность —*** спо­соб­ность ПО вы­пол­нять тре­бу­емые за­дачи в обоз­на­чен­ных ус­ло­ви­ях на про­тяже­нии за­дан­но­го про­межут­ка вре­мени или ука­зан­ное ко­личес­тво опе­раций.

Ат­ри­буты дан­ной ха­рак­те­рис­ти­ки — это за­вер­шенность и це­лос­тность всей сис­те­мы, спо­соб­ность са­мос­то­ятельно и кор­рек­тно вос­ста­нав­ли­ваться пос­ле сбо­ев в ра­боте, от­ка­зо­ус­тойчи­вость***.***

***Удобс­тво ис­пользо­вания*** — воз­можность ком­фор­тно­го изу­чения ПО пользо­вате­лем.

***Эф­фектив­ность*** — спо­соб­ность ПО обес­пе­чивать тре­бу­емый уро­вень про­из­во­дительнос­ти в со­от­ветс­твии с вы­делен­ны­ми ре­сур­са­ми, вре­менем и дру­гими обоз­на­чен­ны­ми ус­ло­ви­ями.

***Удобс­тво соп­ро­вож­де­ния*** — лег­кость, с ко­торой ПО мо­жет ана­лизи­роваться, тес­ти­роваться, из­ме­няться для ис­прав­ле­ния де­фек­тов, для ре­али­зации но­вых тре­бова­ний, для об­легче­ния дальнейше­го об­слу­жива­ния и адап­ти­роваться к име­юще­муся ок­ру­жению.

***Пор­та­тив­ность***— ха­рак­те­ризу­ет ПО с точ­ки зре­ния мо­бильнос­ти его пе­рено­са из од­но­го ок­ру­жения в дру­гое.



Об­щим меж­ду­народ­ным стан­дартом, ко­торый лю­бая ком­па­ния в сфе­рах про­из­водс­тва мо­жет при­нять за ос­но­ву раз­ви­тия сис­те­мы уп­равле­ния ка­чес­твом, мож­но наз­вать ISO 9000, *раз­ра­ботан­ный Меж­ду­народ­ной ор­га­низа­ци­ей по стан­дарти­зации (ISO).*

Этот до­кумент со­дер­жит ряд все­воз­можных стан­дартов, при­мени­мых как в про­мыш­леннос­ти, так и в сфе­ре ус­луг.

Нап­ри­мер, стан­дарт ISO 9001 рег­ла­мен­ти­ру­ет де­ятельность ор­га­низа­ций, за­нима­ющих­ся раз­ра­бот­кой, про­из­водс­твом и соп­ро­вож­де­ни­ем раз­личных то­варов. Раз­дел ISO 9000-3 под­держи­ва­ет ве­дение до­кумен­та­ции, адап­ти­ру­ющей стан­дарт ISO 9000 к раз­ра­бот­ке прог­рам­мных про­дук­тов.

***Уп­равле­ние ка­чес­твом ПО*** вклю­ча­ет в се­бя сле­ду­ющие про­цес­сы:

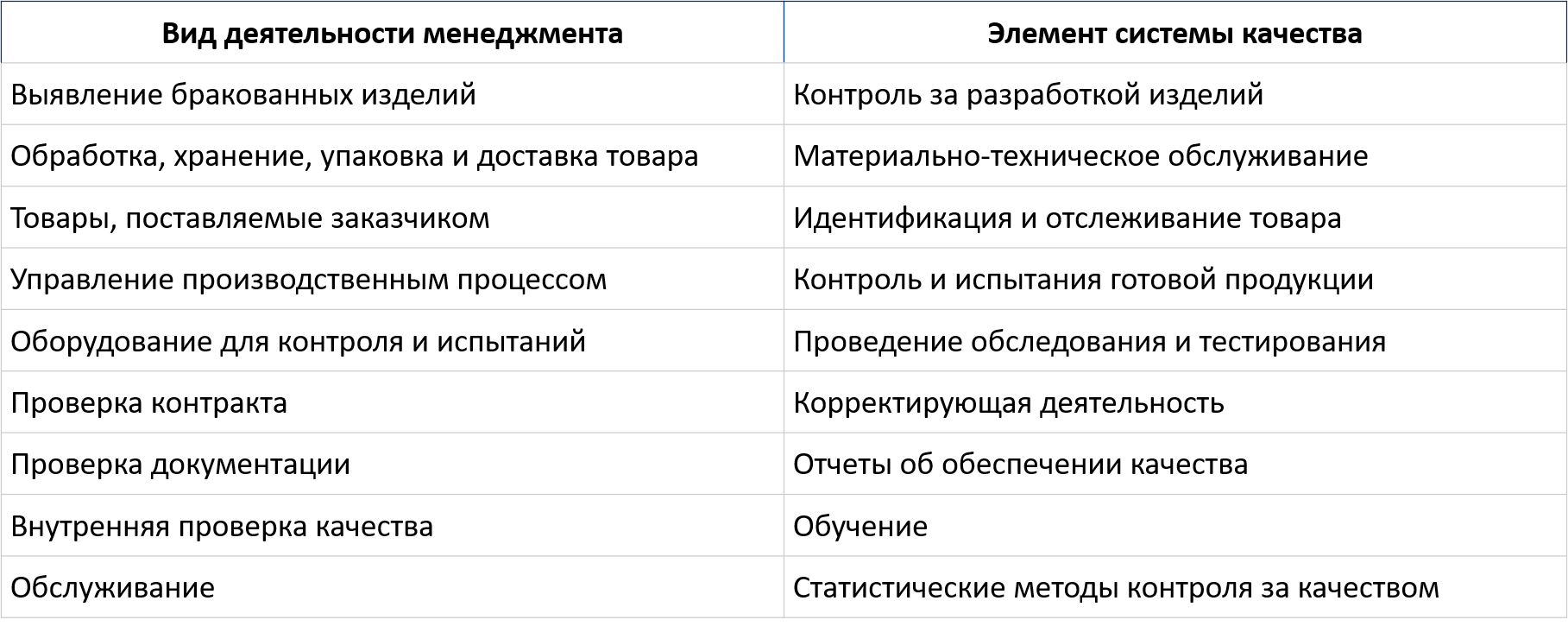
1) оп­ре­деле­ние стан­дартов, рег­ла­мен­ти­ру­ющих про­цесс раз­ра­бот­ки ПО;

2) кон­троль за про­цес­сом раз­ра­бот­ки (для обес­пе­чения вы­пол­не­ния стан­дартов);

3) соз­да­ние от­четнос­ти о хо­де про­цес­са раз­ра­бот­ки для ме­нед­же­ра про­ек­та и за­каз­чи­ка ПО.

**Су­щес­тву­ет два вза­имо­допол­ня­ющих под­хо­да к про­цес­су кон­тро­ля ка­чес­тва**:1) *про­вер­ки ка­чес­тва*, ког­да прог­рам­мный про­дукт, соп­ро­вож­да­ющая до­кумен­та­ция и про­цесс раз­ра­бот­ки ана­лизи­ру­ют­ся груп­пой про­веря­ющих.

2) *ав­то­мати­зиро­ван­ная оцен­ка* *ПО*, ког­да прог­рам­мный про­дукт и его до­кумен­та­ция про­веря­ют­ся спе­ци­альной компьютер­ной прог­раммой, ко­торая со­пос­тавля­ет их со стан­дарта­ми дан­но­го про­ек­та.



1. **CALS-технологии.**

CALS-технологии - это процесс создания единого информационного пространства в отдельно взятой системе обеспечения жизненного цикла продукции. Когда производственные системы стали более сложными, появилась потребность в создании инструментов и методов для быстрого обмена информацией между участниками производственных процессов на разных стадиях использования товаров. CALS-технологии помогают объединить промышленные автоматизированные системы в многофункциональную систему.

Целью ин­тегра­ции яв­ля­ет­ся по­выше­ние эф­фектив­ности соз­да­ния и ис­пользо­вания слож­ных прог­рам­мных сис­тем. При­мени­тельно к раз­ра­бот­ке ПО по­выше­ние ее эф­фектив­ности обес­пе­чива­ет­ся за счет вы­пол­не­ния трех ме­роп­ри­ятий: **Мероприятия по повышению эффективности ПО**



Основные принципы CALS-технологий базируются на контроле и организации этапов существования продукции. К ним относят:

* обеспечение системного управления минимизацию затрат на всех стадиях;
* использование стандартных механизмов описания управляемых объектов дифференциацию программных элементов на основе использования общих стандартов (данных и интерфейсов доступа) и применение платформ на коммерческой основе;
* представление информации на безбумажной основе с приоритетом использования электронной подписи;
* сопутствующий инжиниринг всех процессов;
* непрерывное корректирование и усовершенствование с целью создания оптимальной модели управления.

**Примеры использования**

CALS-технологии в России используются на многих отечественных предприятиях, как гражданского, так и военного сектора:

Электронная документация используется для многих изделий. К примеру, в авиации для самолетов, вертолетов, авиационных двигателей и комплектующих.

Помимо этого, ведутся разработки систем навигации, телефонной и радио связи, управления.

Применяются при проектировании и разработке автотракторной техники.

Элементы системы используются на Воронежском механическом заводе, в государственной корпорации "Росатом", НПП "Аэросила", ОАО "Российские железные дороги" и др.

1. **Организация процесса обновления в информационной системе.**

Обыч­но под ***об­новле­ни­ем ПО*** по­нима­ют до­пол­не­ния к прог­рам­мно­му обес­пе­чению, пре­дот­вра­ща­ющие или ус­тра­ня­ющие не­полад­ки, по­выша­ющие бе­зопас­ность ли­бо улуч­ша­ющие про­из­во­дительность компьютер­ной сис­те­мы.

На­боры об­новле­ний, ис­прав­ле­ний и улуч­ше­ний компьютер­ной прог­раммы, по­став­ля­емые в ви­де еди­ного ус­та­новоч­но­го дис­три­бути­ва, при­нято на­зывать ***па­кета­ми об­новле­ний***. Па­кеты об­новле­ния обыч­но ну­меру­ют­ся и крат­ко ука­зыва­ют­ся как SP1, SP2, SP3 и т. д.

Пат­чи. Мно­гие ком­па­нии, нап­ри­мер Microsoft или Autodesk, вы­пус­ка­ют па­кет об­новле­ний, ког­да чис­ло от­дельных пат­чей для кон­крет­ной прог­раммы дос­ти­га­ет не­кото­рого пре­дела.

***Патч*(или «зап­латка»)** — это не­большой прог­рам­мный код, пред­назна­чен­ный для за­мены оши­боч­ной и (или) не­оп­ти­мизи­рован­ной час­ти дру­гой прог­раммы.

Ис­прав­ле­ние с по­мощью пат­чей мо­жет при­меняться к уже ус­та­нов­ленной прог­рамме ли­бо к ее ис­ходным ко­дам. Сю­да вхо­дят ис­прав­ле­ние оши­бок, из­ме­нение внеш­не­го ви­да, улуч­ше­ние эр­го­номич­ности или про­из­во­дительнос­ти прог­рамм, а так­же лю­бые дру­гие из­ме­нения, ко­торые раз­ра­бот­чик по­желал сде­лать.

***Хот­фикс*** — тер­мин, при­меня­ющийся для пат­чей, ко­торые ус­та­нав­ли­ва­ют­ся на ра­бота­ющую сис­те­му без пе­реза­пус­ка.

Час­то хот­фиксы пред­назна­чены для ре­шения кон­крет­ных проб­лем той или иной ор­га­низа­ции или кон­крет­ных пользо­вате­лей и не вы­ходят за ее пре­делы.

Так­же хот­фиксы мо­гут пре­дос­тавляться для кон­крет­ной кон­фи­гура­ции обо­рудо­вания и не ра­ботать с дру­гими сис­те­мами.

Важ­ность об­новле­ний и их свое­вре­мен­ную ус­та­нов­ку для под­держа­ния над­ле­жаще­го уров­ня ин­форма­ци­он­ной бе­зопас­ности труд­но пе­ре­оце­нить.

***Ме­тод уп­равле­ния об­новле­ни­ями*** яв­ля­ет­ся ком­би­наци­ей под­хо­да к тес­ти­рова­нию об­новле­ний и под­хо­да к раз­верты­ванию ре­лизов с об­новле­ни­ями.

**Раз­ли­ча­ют два под­хо­да к тес­ти­рова­нию об­новле­ний:**

1) на ло­кальных вир­ту­альных ма­шинах;

2) в пол­но­цен­ной тес­то­вой сре­де.

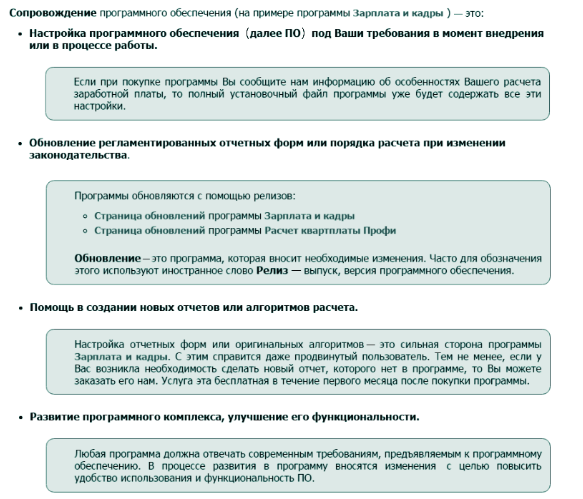
В пер­вом слу­чае для соз­да­ния вир­ту­альных ма­шин и се­тей ис­пользу­ют­ся ли­бо про­дукт VMware, ли­бо про­дукт Oracle VirtualBox. Как пра­вило, тех­но­логия вир­ту­али­зации ис­пользу­ет­ся для не­больших се­тей с чис­лом ра­бочих стан­ций 45…70, не бо­лее.

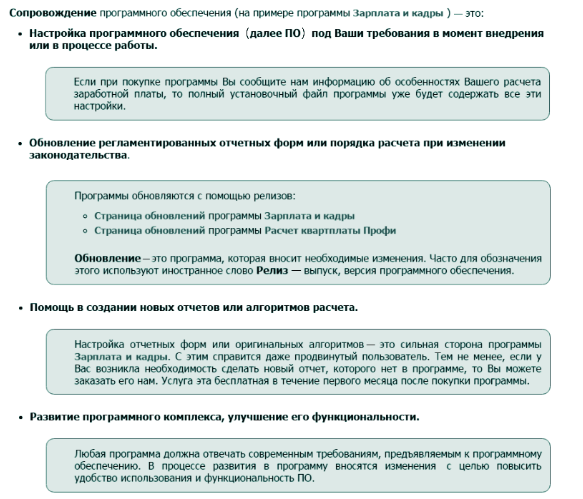
Тестирование обновлений в полноценной тестовой среде — это подход, который используется для больших промышленных сетей и гарантирует высокую чистоту тестирования.

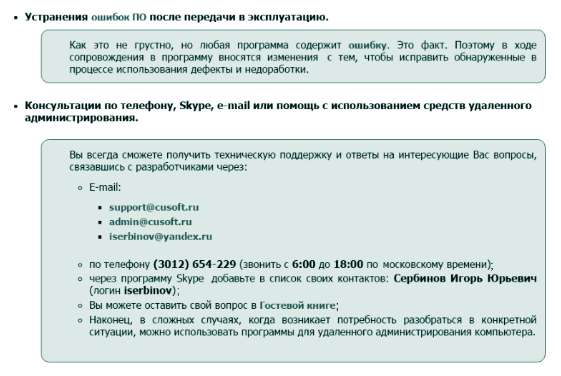
**Раз­ли­ча­ют об­новле­ния ОС и прик­ладно­го ПО.**

Об­новле­ния для ОС и сер­верно­го ПО при­меня­ют­ся в це­лях под­дер­жки над­ле­жаще­го уров­ня бе­зопас­ности и ус­тра­нения «дыр» в за­щите.

Об­новле­ния прик­ладно­го ПО (нап­ри­мер, Microsoft Office, Adobe Acrobat, кли­ент­ские час­ти биз­нес-при­ложе­ний) не­об­хо­димы для ре­шения воз­никших проб­лем с час­то ис­пользу­емы­ми или важ­ны­ми биб­ли­оте­ками и дру­гими час­тя­ми ис­ходно­го ко­да.

1. **Регламенты обновления.** 





1. **Тестирование программного обеспечения в процессе внедрения и эксплуатации.**

Тес­ти­рова­ние ис­пользу­ет­ся для об­на­руже­ния оши­бок и де­фек­тов в ПО. Целью тес­ти­рова­ния яв­ля­ет­ся вы­яв­ле­ние в ПО скры­тых де­фек­тов до то­го, как оно бу­дет сда­но за­каз­чи­ку.



Обыч­но тес­ти­рова­ние ба­зиру­ет­ся на не­кото­ром под­мно­жес­тве все­воз­можных тес­то­вых сце­нари­ев, пос­кольку про­верить все воз­можные пос­ле­дова­тельнос­ти вы­пол­не­ния прог­раммы не­ре­ально.

**Тестирование сборки**

Пос­ле то­го как про­тес­ти­рова­ны все от­дельные прог­рам­мные мо­дули, вы­пол­ня­ет­ся сбор­ка сис­те­мы, в ре­зульта­те че­го соз­да­ет­ся час­тичная или пол­ная прог­рам­мная сис­те­ма.

При ее тес­ти­рова­нии вы­яв­ля­ют­ся проб­ле­мы, воз­ни­ка­ющие при вза­имо­действии прог­рам­мных мо­дулей сис­те­мы.

**Тестирование интерфейса**

Дан­ное тес­ти­рова­ние вы­пол­ня­ет­ся в тех слу­ча­ях, ког­да мо­дули или под­систе­мы ин­тегри­ру­ют­ся в большие сис­те­мы.

Цель тес­ти­рова­ния ин­терфейса — вы­явить ошиб­ки, воз­ни­ка­ющие в сис­те­ме вследс­твие оши­бок в ин­терфейсах.

Типы интерфейсов

* ***па­рамет­ри­чес­кие ин­терфейсы***, в ко­торых ссыл­ки на дан­ные и фун­кции пе­реда­ют­ся в ви­де па­рамет­ров от од­но­го ком­по­нен­та к дру­гому;
* ***ин­терфейсы раз­де­ля­емой па­мяти***, в ко­торых ка­кой-ли­бо блок па­мяти сов­мес­тно ис­пользу­ет­ся раз­ны­ми под­систе­мами. Од­на под­систе­ма по­меща­ет дан­ные в па­мять, а дру­гие под­систе­мы ис­пользу­ют эти дан­ные;
* ***про­цедур­ные ин­терфейсы***, в ко­торых од­на под­систе­ма ин­капсу­лиру­ет (вклю­ча­ет в се­бя) на­бор про­цедур, вы­зыва­емых из дру­гих под­систем. Та­кой тип ин­терфейса име­ют объек­ты и абс­трак­тные ти­пы дан­ных;
* ***ин­терфейсы пе­реда­чи со­об­ще­ний*,** в ко­торых од­на под­систе­ма зап­ра­шива­ет сер­вис у дру­гой под­систе­мы пос­редс­твом пе­реда­чи ей со­об­ще­ния, при­чем от­ветное со­об­ще­ние со­дер­жит ре­зульта­ты вы­пол­не­ния сер­ви­са (при­мер: так ра­бота­ют сис­те­мы «кли­ент-сер­вер»).

Классы ошибок

***Не­кор­рек­тное ис­пользо­вание ин­терфейса*** име­ет мес­то, ког­да при вы­зове мо­дуля вы­зыва­ющий мо­дуль пе­реда­ет вы­зыва­емо­му па­рамет­ры, на­рушая по­рядок их сле­дова­ния или ука­зывая не­вер­ное их чис­ло и (или) тип пе­реда­ва­емых дан­ных.

***Неп­ра­вильное по­нима­ние ин­терфейсов*** име­ет мес­то, ког­да по­веде­ние вы­зыва­емо­го мо­дуля не сов­па­да­ет с ожи­да­емым. В этом слу­чае по­веде­ние вы­зыва­юще­го мо­дуля ста­новит­ся неп­ред­ска­зу­емым. Нап­ри­мер, ес­ли прог­рамма би­нар­но­го по­ис­ка вы­зыва­ет­ся для по­ис­ка за­дан­но­го эле­мен­та в не­упо­рядо­чен­ном мас­си­ве, то в ра­боте прог­раммы про­изойдет сбой.

***Ошиб­ки син­хро­низа­ции ин­терфейсов*** встре­ча­ют­ся в сис­те­мах ре­ально­го вре­мени, где ис­пользу­ют­ся ин­терфейсы раз­де­ля­емой па­мяти или пе­реда­чи со­об­ще­ний. Под­систе­ма-ис­точник и под­систе­ма-при­ем­ник дан­ных мо­гут ра­ботать с раз­ной ско­ростью. Ес­ли при про­ек­ти­рова­нии ин­терфейса не учи­тывать этот фак­тор, при­ем­ник мо­жет по­лучить дос­туп к ус­та­рев­шим дан­ным, по­тому что ис­точник к то­му мо­мен­ту еще не ус­пел об­но­вить сов­мес­тно ис­пользу­емые дан­ные.

**Объектно-ориентированные системы (ООС)**

Сис­те­мы, раз­ра­ботан­ные по фун­кци­ональной мо­дели, и объек­тно-ори­ен­ти­рован­ные сис­те­мы име­ют сле­ду­ющие су­щес­твен­ные от­ли­чия:

* объек­ты, как от­дельные прог­рам­мные ком­по­нен­ты, пред­став­ля­ют со­бой неч­то большее, чем от­дельные под­прог­раммы или фун­кции;
* объек­ты, ин­тегри­рован­ные в под­систе­мы, обыч­но сла­бо свя­заны меж­ду со­бой, и по­это­му слож­но оп­ре­делить са­мый вер­хний уро­вень сис­те­мы;
* при ана­лизе пов­торно ис­пользу­емых объек­тов их ис­ходный код мо­жет быть не­до­ступ­ным для ис­пы­тате­лей.

При­мени­тельно к О­ОС мож­но оп­ре­делить че­тыре уров­ня тес­ти­рова­ния:

1) тес­ти­рова­ние от­дельных ме­тодов, ас­со­ци­иро­ван­ных с объек­та­ми;

2) тес­ти­рова­ние от­дельных клас­сов объек­тов;

3) тес­ти­рова­ние клас­те­ров объек­тов — обыч­но ис­пользу­ет ме­тоды, ос­но­ван­ные на сце­нари­ях;

4) тес­ти­рова­ние сис­те­мы — ве­рифи­кация и ат­теста­ция О­ОС вы­пол­ня­ет­ся так же, как и для лю­бых дру­гих ти­пов сис­тем.



***Ор­га­низа­тор тес­тов*** уп­равля­ет вы­пол­не­ни­ем тес­тов, от­сле­живая тес­то­вые дан­ные, ожи­да­емые ре­зульта­ты и тес­ти­ру­емые фун­кции прог­раммы.

***Ге­нера­тор тес­то­вых дан­ных*** по­рож­да­ет тес­то­вые дан­ные для тес­ти­ру­емой прог­раммы, ли­бо вы­бира­ет их из ба­зы дан­ных, ли­бо ис­пользу­ет спе­ци­альные шаб­ло­ны для ге­нера­ции слу­чайных дан­ных не­об­хо­димо­го ви­да.

***Ора­кул***ге­нери­ру­ет ожи­да­емые ре­зульта­ты тес­тов, ко­торые в бло­ке «Ком­па­ратор файлов» срав­ни­ва­ют­ся с ре­зульта­тами тес­ти­рова­ния, по­лучен­ны­ми от но­вой вер­сии прог­раммы.

***Ком­па­ратор файлов*** по ре­зульта­там срав­не­ния сос­тавля­ет от­чет об об­на­ружен­ных раз­ли­чи­ях. Ком­па­рато­ры осо­бен­но важ­ны при срав­не­нии раз­личных вер­сий прог­раммы. Раз­ли­чия в ре­зульта­тах ука­зыва­ют на воз­можные проб­ле­мы, су­щес­тву­ющие в но­вой вер­сии сис­те­мы.

***Ге­нера­тор от­че­тов*** фор­ми­ру­ет от­че­ты по ре­зульта­там тес­ти­рова­ния.

***Ди­нами­чес­кий ана­лиза­тор*** под­счи­тыва­ет, сколько раз вы­пол­ня­ет­ся каж­дый опе­ратор тес­ти­ру­емой прог­раммы и вы­да­ет от­чет по ре­зульта­там сче­та.

***Ими­татор***мо­дели­ру­ют ма­шину, на ко­торой бу­дет вы­пол­няться прог­рамма или мо­дели­ру­ет вза­имо­действия с ин­терфейсом пользо­вате­ля или, на­конец, ге­нери­ру­ет пос­ле­дова­тельнос­ти пов­то­ря­ющих­ся тран­закций, т. е. зап­ро­сов к ба­зе дан­ных.

1. **Эксплуатационная документация.**

Для из­го­тов­ле­ния, соп­ро­вож­де­ния, тес­ти­рова­ния и экс­плу­ата­ции ПК соз­да­ет­ся про­грам­мная и экс­плу­ата­ци­он­ная до­кумен­та­ция.

Го­сударст­вен­ный стан­дарт ГОСТ 19.101—77 «Еди­ная сис­те­ма прог­рам­мной до­кумен­та­ции (ЕСПД).

Ви­ды прог­рамм и прог­рам­мных до­кумен­тов» пре­дус­матри­ва­ет соз­да­ние ря­да ос­новных до­кумен­тов, ка­са­ющих­ся ПО.

**Ос­новные до­кумен­ты ЕСПД**, соз­да­ние ко­торых ого­вари­ва­ет ГОСТ 19.301—79 «ЕСПД.

Прог­рамма и ме­тоди­ка ис­пы­таний. Тре­бова­ния к со­дер­жа­нию и офор­мле­нию», — это «Прог­рамма и ме­тоди­ка ис­пы­таний», «Опи­сание прог­раммы», «По­яс­ни­тельная за­пис­ка», «Текст прог­раммы» и «Опи­сание при­мене­ния».

**Прог­рам­мная до­кумен­та­ция:**

***текст прог­раммы*** — за­пись прог­раммы с не­об­хо­димы­ми ком­мента­ри­ями;

***опи­сание прог­раммы*** — све­дения о ло­гичес­кой струк­ту­ре и по­ряд­ке ее ра­боты;

***программа и ме­тоди­ка ис­пы­таний*** — тре­бова­ния, под­ле­жащие про­вер­ке при ис­пы­тании прог­раммы, а так­же по­рядок и ме­тоды их кон­тро­ля;

***тех­ни­чес­кое за­дание***— спе­цифи­кация тре­бова­ний к соз­да­ва­емо­му ПК с точ­ки зре­ния за­каз­чи­ка;

***по­яс­ни­тельная за­пис­ка*** — схе­ма ал­го­рит­ма, об­щее опи­сание ал­го­рит­ма и (или) фун­кцио­ни­рова­ния прог­раммы, а так­же обос­но­вание при­нятых тех­ни­чес­ких и эко­номи­чес­ких ре­шений.

В до­кумен­те **«Текст прог­раммы»** при­водит­ся сим­во­личес­кая за­пись прог­рамм ком­плек­са на ис­ходных язы­ках, соп­ро­вож­да­емая не­об­хо­димы­ми ком­мента­ри­ями.

В раз­де­лах до­кумен­та **«Опи­сание прог­раммы»** ука­зыва­ет­ся обоз­на­чение, на­име­нова­ние и фун­кци­ональное наз­на­чение прог­раммы, сос­тав ее ПО, язы­ки прог­рамми­рова­ния, на ко­торых на­писа­на прог­рамма.

В раз­де­ле **«Опи­сание ло­гичес­кой струк­ту­ры»** дол­жны быть ука­заны ал­го­ритм прог­раммы, ис­пользу­емые ме­тоды, струк­ту­ра прог­раммы с опи­сани­ем фун­кций сос­тавных час­тей и свя­зи меж­ду ни­ми, а так­же свя­зи прог­раммы с дру­гими прог­рамма­ми ПК.

В раз­де­ле **«Ис­пользу­емые тех­ни­чес­кие средс­тва»** ука­зыва­ют­ся ти­пы и па­рамет­ры ус­тройств, ко­торые ис­пользу­ют­ся при ра­боте прог­раммы.

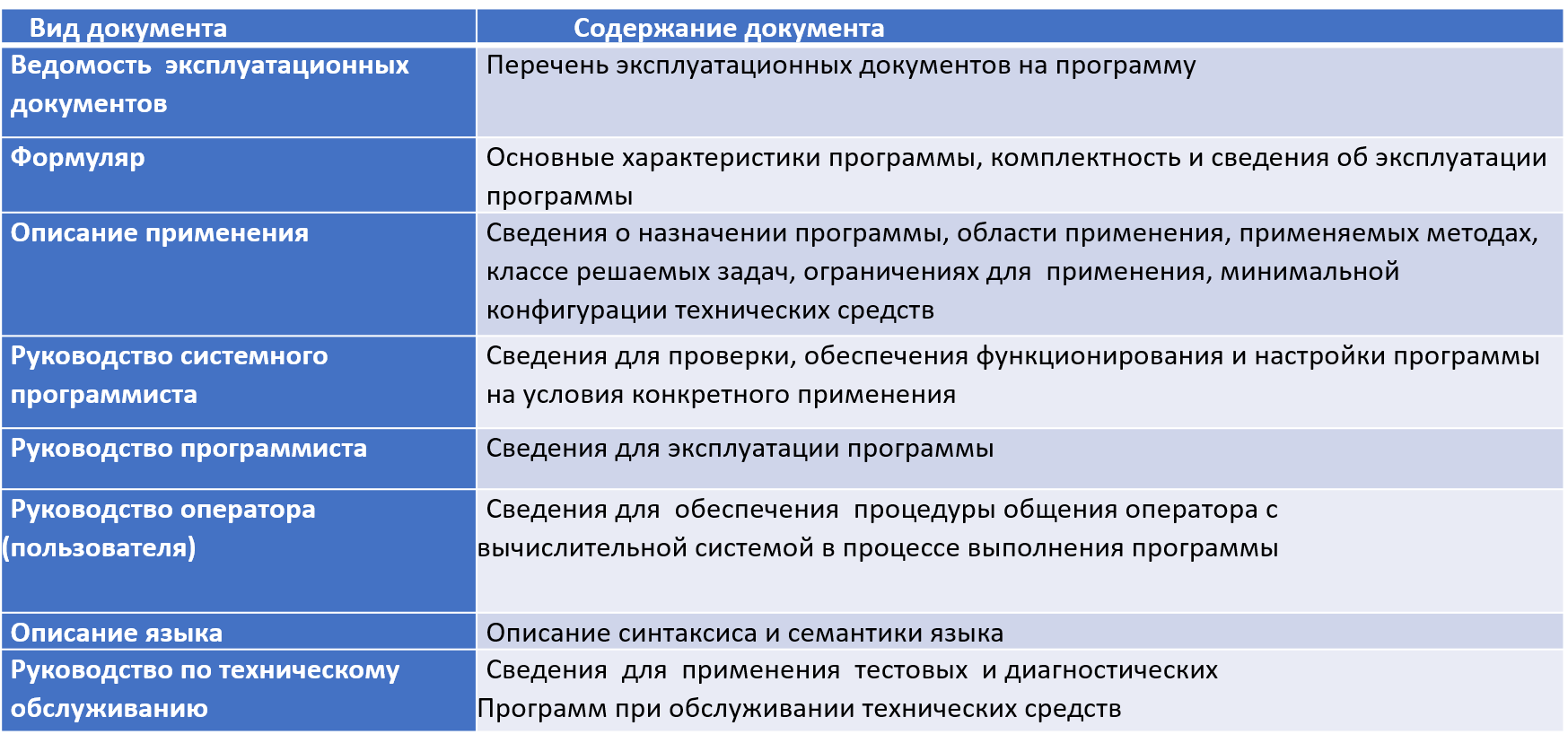
В раз­де­ле **«Вы­зов и заг­рузка»** ука­зыва­ют­ся спо­соб вы­зова прог­раммы с со­от­ветс­тву­юще­го но­сите­ля дан­ных и точ­ки вхо­да в прог­рамму.

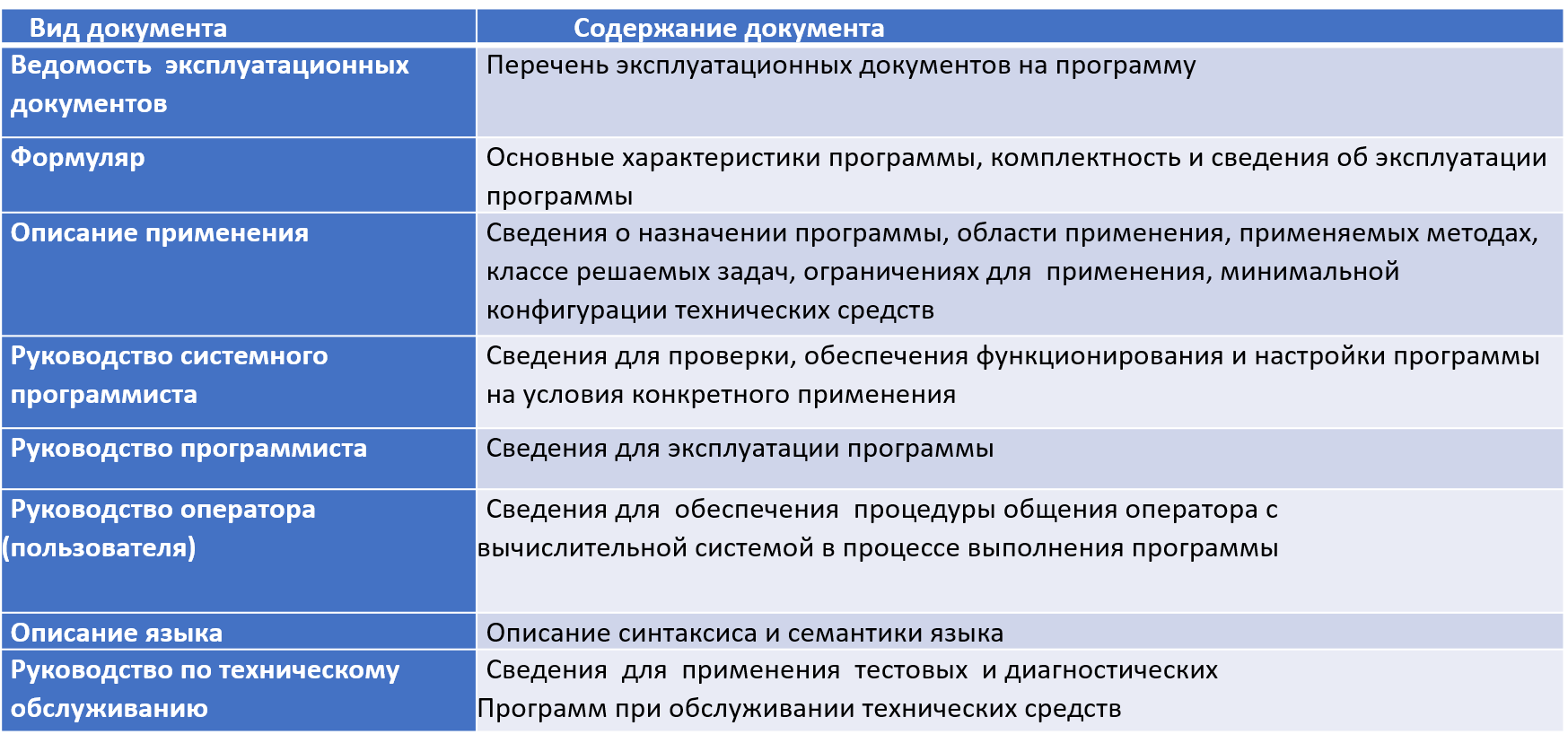
В раз­де­лах **«Вход­ные дан­ные» и «Вы­ход­ные дан­ные**» да­ют­ся све­дения о под­го­тов­ке дан­ных, опи­сыва­ет­ся их фор­мат и спо­соб ко­диро­вания.

В раз­де­лах до­кумен­та **«Программа и ме­тоди­ка ис­пы­таний»** ука­зыва­ют­ся на­име­нова­ние, об­ласть при­мене­ния и обоз­на­чение ис­пы­ту­емо­го ПК, цель про­веде­ния ис­пы­таний, при­водит­ся пе­речень до­кумен­та­ции, предъяв­ля­емой при про­веде­нии ис­пы­таний, тех­ни­чес­кие тре­бова­ния к прог­рам­мной до­кумен­та­ции и тех­ни­чес­ким ха­рак­те­рис­ти­кам.

В раз­де­лах до­кумен­та **«По­яс­ни­тельная за­пис­ка»** при­водит­ся пе­речень до­кумен­тов, на ос­но­вании ко­торых ве­дет­ся раз­ра­бот­ка, из­ла­га­ет­ся пос­та­нов­ка за­дачи на раз­ра­бот­ку прог­раммы.

**Са­мос­то­ятельную груп­пу сос­тавля­ет *экс­плу­ата­ци­он­ная до­кумен­та­ция***





В до­кумен­те **«Опи­сание при­мене­ния»** при­водят­ся раз­де­лы, опи­сыва­ющие наз­на­чение прог­раммы, ус­ло­вия ее при­мене­ния, не­об­хо­димые для ее вы­пол­не­ния, да­ет­ся опи­сание пос­та­нов­ки за­дачи и ме­тодов ее ре­шения, а так­же опи­сыва­ют­ся вход­ные и вы­ход­ные дан­ные.

Ори­гинальная часть до­кумен­та **«Ру­ководс­тво сис­темно­го прог­раммис­та»** со­дер­жит све­дения о нас­тройке и про­вер­ке прог­раммы, со­об­ще­ния сис­темно­му прог­раммис­ту и до­пол­ни­тельные воз­можнос­ти прог­раммы.

Ори­гинальная часть до­кумен­та **«Ру­ководс­тво прог­раммис­та»** со­дер­жит све­дения об об­ра­щении к прог­рамме (про­цеду­ры вы­зова и спо­собы пе­реда­чи уп­равле­ния и па­рамет­ров) и тек­сты ад­ре­сован­ных прог­раммис­ту со­об­ще­ний.

Документ **ру­ководс­тво пользо­вате­ля** со­дер­жит све­дения о наз­на­чении прог­раммы, об об­ласти ее при­мене­ния, о при­меня­емых ме­тодах и ог­ра­ниче­ни­ях, кон­фи­гура­ции тех­ни­чес­ких средств и све­дения, не­об­хо­димые для об­ще­ния пользо­вате­ля с пер­со­нальным компьюте­ром в про­цес­се вы­пол­не­ния про­грам­мы.

документ **"Описание языка"** включается в программную документацию, если разработанный программный продукт реализует некий язык программирования, управления заданиями, организации вычислительного процесса и т. п.

документ **"Руководство по техническому обслуживанию"** документацию, если разработанный программный продукт требует использования тестовых или диагностических программ.

1. **Понятие совместимости программного обеспечения.**

**Программная совместимость** (ПС) - это если программы, составленные для одной модели, могут выполняться на других моделях ряда.

Программная совместимость обеспечивается единым набором команд, единой формой представления данных и единой системой адресации для всех программных модулей.

**Кон­цепция прог­рам­мной сов­мести­мос­ти выд­ви­нула в чис­ло при­ори­тет­ных ряд но­вых тре­бова­ний к вновь раз­ра­баты­ва­емым ПО и ап­па­рат­но-прог­рам­мным ком­плек­сам**. Действи­тельно, прог­рам­мно сов­мести­мая вы­чис­ли­тельная сре­да дол­жна:

* поз­во­лять гиб­ко ме­нять ко­личес­тво и сос­тав ап­па­рат­ных средств и ПО в со­от­ветс­твии с ме­ня­ющи­мися тре­бова­ни­ями ре­ша­емых за­дач;
* обес­пе­чивать воз­можность за­пус­ка од­них и тех же ПК на раз­личных ап­па­рат­ных плат­формах, обес­пе­чивая мо­бильность ПО;
* га­ран­ти­ровать воз­можность при­мене­ния од­них и тех же че­лове­ко-ма­шин­ных ин­терфейсов на всех компьюте­рах, вхо­дящих в не­од­но­род­ную сеть.

**Аппаратная совместимость** означает способность одного компьютера работать с узлами или устройствами, входящими в состав другого компьютера.

**Информационная совместимость**  — это способность двух и более компьютеров или систем адекватно воспринимать одинаково представленные данные.

**Программная совместимость**  — это возможность выполнения одних и тех же программ на разных компьютерах с получением одинаковых результатов.

В области персональных компьютеров сегодня наиболее широко распространены две аппаратные платформы — IBM РС и Аррlе Macintosh.

**Обеспечение совместимости компьютерных платформ**.

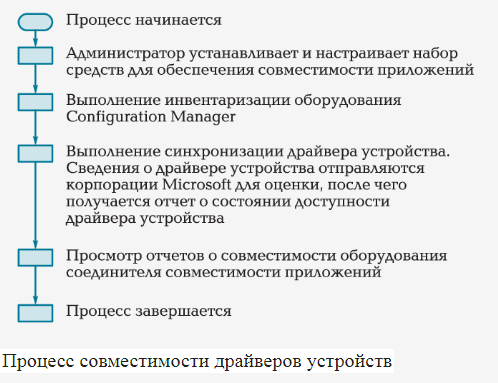
В  настоящее время существуют два основных варианта решения этой проблемы: *аппаратные и программные*.

**Аппаратные решения** заключаются в  использовании печатных модулей (плат), содержащих дополнительный микропроцессор и  «лишнюю» оперативную и  видеопамять от другой аппаратной платформы.

**Программные решения** заключаются в  написании программ эмуляторов, позволяющих запускать ПО, разработанное для ПЭВМ одного типа, на другой ПЭВМ.

**Программа-эмулятор** представляет собой программу, выполняющую каждую команду эмулируемой программы посредством одной или нескольких команд той ПЭВМ, на которой реализуется эмуляция.

**К программным решениям** совместимости компьютерных платформ относится также обеспечение совместимости драйверов  — частей кода ОС, отвечающих за взаимодействие ПО с физической и (или) виртуальной аппаратурой.



1. **Аппаратная и программная совместимость. Совместимость драйверов.**

**Аппаратная совместимость** означает способность одного компьютера работать с узлами или устройствами, входящими в состав другого компьютера.

**Информационная совместимость**  — это способность двух и более компьютеров или систем адекватно воспринимать одинаково представленные данные.

**Программная совместимость**  — это возможность выполнения одних и тех же программ на разных компьютерах с получением одинаковых результатов.

В отличие от программной совместимости термин «совместимость программ» означает пригодность программ к взаимодействию друг с другом и, в частности, к объединению в  персональном компьюторе для решения более сложных задач, например в автоматизированных системах.

**Полная совместимость**  — это аппаратная, программная и  информационная совместимость двух или более компьютеров без каких-либо ограничений для их пользователей.

В области персональных компьютеров сегодня наиболее широко распространены две аппаратные платформы — IBM РС и Аррlе Macintosh.

**Обеспечение совместимости компьютерных платформ**.

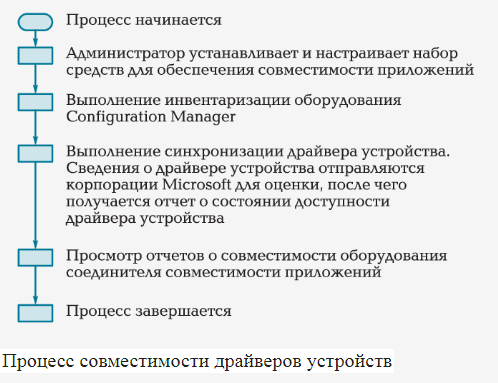
В  настоящее время существуют два основных варианта решения этой проблемы: *аппаратные и программные*.

Аппаратные решения заключаются в  использовании печатных модулей (плат), содержащих дополнительный микропроцессор и  «лишнюю» оперативную и  видеопамять от другой аппаратной платформы.

**Программные решени**я заключаются в  написании программ эмуляторов, позволяющих запускать ПО, разработанное для ПЭВМ одного типа, на другой ПЭВМ.

**Программа-эмулятор** представляет собой программу, выполняющую каждую команду эмулируемой программы посредством одной или нескольких команд той ПЭВМ, на которой реализуется эмуляция.

**К программным решениям** совместимости компьютерных платформ относится также обеспечение совместимости драйверов  — частей кода ОС, отвечающих за взаимодействие ПО с физической и (или) виртуальной аппаратурой.



Компьютерное программное обеспечение, с помощью которого другое программное обеспечение (операционная система) получает доступ к аппаратному обеспечению некоторого устройства.

**Раз­ли­ча­ют од­но- и мно­го­уров­не­вые драйве­ры.**

У многоуровневых драйверов об­ра­бот­ка зап­ро­сов вво­да-вы­вода рас­пре­деля­ет­ся меж­ду от­дельны­ми драйве­рами, каж­дый из ко­торых вы­пол­ня­ет свою часть ра­боты.

При об­ра­бот­ке зап­ро­са дан­ные идут от драйве­ров вер­хне­го уров­ня к драйве­рам ниж­не­го, а при ис­полне­нии зап­ро­са — в об­ратном нап­равле­нии.

1. шинные драйверы — обслуживают контроллер шины, мосты, причем для каждого типа шины в системе имеется собственный шинный драйвер;
2. фильтр-драйвер  — сортирует запросы ввода-вывода для шины и  других шинных устройств и  находится на различных уровнях (как выше, так и ниже шинного драйвера);
3. функциональный драйвер  — является обязательным и  основным драйвером устройства, для которого он реализует интерфейс.

По структуре драйвер представляет собой обычную динамически подключаемую DLL-библиотеку уровня ядра ОС, дополняя его набором процедур, периодически вызываемых внешними программами.

1. **Причины возникновения проблем совместимости. Методы выявления проблем совместимости ПО.**

Ос­новные при­чины воз­никно­вения прог­рам­мных оши­бок — это не­совер­шенс­тво ПО и ОС.

Нап­ри­мер, ес­ли при сво­ей ра­боте прог­рамма зах­ва­тыва­ет ре­сур­сы, кон­тро­лиро­вать ко­торые она не спо­соб­на, по­яв­ля­ет­ся ПО, име­ющее мно­жес­тво не­доче­тов и пос­то­ян­но вы­да­ющее ошиб­ки.

также, если программа пытается эксплуатировать установленные в ОС системные библиотеки, не учитывая, что они постоянно обновляются корпорацией, поставляющей данную ОС.

Часто для успешной работы квалифицированно составленной программы требуется большее количество ресурсов, чем ей может предоставить ОС.

Серьезно нарушить стабильность работы ОС могут непрофессиональные действия программиста с реестром ОС, которые прописывают свои файлы и ссылки в самых различных его разделах.

Ошибки в реестре негативно сказываются на всех процессах, происходящих в компьютере.

**Методы выявления проблем совместимости программного обеспечения.**

*1) проверить используемые на предприятии программы на совместимость с  новой ОС и  определить те, у  которых возникли проблемы совместимости;*

*2) обновить версии проблемных программ, а  если в  них предприятие больше не нуждается — удалить из ОС.*

Наиболее распространенным методом устранения проблем совместимости ПО является использование динамически загружаемых библиотек.

**Динамические исполняемые библиотеки (DLL)** представляют собой специальные программные компоненты, которые позволяют обращаться к одному файлу сразу нескольким приложениям. Основным предназначением таких библиотек является расширение возможностей утилит и экономия места, поскольку для использования аналогичных компонентов можно иметь всего одну версию файла, которая будет делиться между всеми программами

DLL – это файлы системных библиотек в Windows. Формально они являются исполняемыми, но запустить их по отдельности не получится.

**Причины появления проблем с файлами DLL:**

Случайное удаление файла.

Отказ оборудования

В системе не установлены последние обновления.

Проблемы с драйверами.

Несовместимость технических требований

**Типы ошибок DLL**:

**«Файл не найден».** Возможно, удалена нужная библиотека, переименована или перемещена в другое место на жестком диске;

**«Файл отсутствует».** Такое сообщение появляется в том случае, если система не может найти нужный объект на диске и часто причиной такой ошибки является работа вредоносного ПО, которое заменило программный код в библиотеке;

**«Нарушение прав доступа к файлу».** С этим типом неполадки сталкиваемся при попытке включения ПО. Вместо запуска работ ПО система выдает сообщение о невозможности получения доступа к библиотеке;

**«DLL не удалось зарегистрировать»** — это означает, что произошла фатальная ошибка при установке приложения. Устранить неполадку поможет повторная инсталляция ПО.

Следующим методом устранения проблем совместимости ПО является **применение группы методов специального тестирования ПО, к которым относятся: (16в подробнее)**

инсталляционное тестирование;

регрессионное тестирование;

тестирование функциональности;

конфигурационное тестирование;

тестирование совместимости;

тестирование интерфейса человек/машина

Проблему решает «штатный» режим совместимости Windows10, позволяющий запустить старую программу, нормально работающую в  одной из предыдущих версий Windows (XP, Vista или Windows 7), обеспечивая ее стабильную работу в новой версии — Windows 10.

**Проблемы совместимости**

1.По­мешать ус­та­нов­ке и за­пус­ку при­ложе­ния мо­гут две при­чины:

-приложения пытаются копировать файлы и ярлыки, созданные в предыдущей ОС Windows, но не существующие в новой ОС;

-приложения пытаются получить доступ к компонентам Windows, которые были переименованы.

 2. Проблемы совместимости могут возникнуть, если включен

контроль учетной записи пользователей (UAC). UAC увеличивает безопасность Windows, ограничивая доступ к  компьютеру без уровня администратора, что ограничивает запуск приложений большинству пользователей в  качестве обычных пользователе

4.Папки и файлы ОС защищены системной программой от перезаписи, поэтому установщики приложений, пытавшиеся их изменить, «натыкаются» на сообщение об ошибке, указывающем на невозможность обновления ресурса.

5. Защищенный режим Internet Explorer помогает защититься от атак с  несанкционированным получением прав, ограничивая возможность записи для любой зоны ресурсов локального компьютера, за исключением временных файлов Интернета.

6. Windows 7 полностью поддерживает 64-битную архитектуру. Приложения или компоненты, использующие 16-битные исполняемые файлы, 16-битные установщики или 32-битные драйвера ядра, могут вызвать сбой при запуске или будут неправильно функционировать.

7. Если пользователь в своей системе использует предыдущую версию интерфейса прикладного программирования (API), то у него могут возникнуть сбои при работе таких приложений, как сканеры сети, антивирусы и фаерволы.

1. **Выполнение чистой загрузки.**

Чистая загрузка обеспечивает бесконфликтную работу ОС и применяется в тех случаях, когда пользователь не может определить причину сбоев ОС при ее обычной загрузке.

Данная функция предусмотрена в ОС компании Microsoft, начиная с Windows 7.

Чистая загрузка Windows отличается от режима «безопасный», используемого, например, при возникновении проблем с поврежденными драйверам, из-за которых не удается запустить компьютер в стандартном режиме.

В безопасном режиме ОС Windows отключает все сторонние драйверы, которые могут помешать ее загрузке, и администратор получает возможность восстановить нормальную систему, удалив поврежденный драйвер

* Чистые старты ОС Windows 10 и Windows 8 идентичны.
* Чтобы произвести старт, необходимо выполнить следующие действия:
* нажать сочетание клавиш [Windows + R] — откроется строка Выполнить;
* в строке окна Выполнить ввести: msconfig — откроется окно изменения конфигураций системы
* на вкладке Общие активировать Выборочный запуск и отключить пункт Загружать элементы автозагрузки
* перейти на вкладку Службы и установить в левом нижнем углу флаг напротив пункта Не отображать службы Microsoft, чтобы случайно не выключить важные для загрузки системы службы;
* нажать кнопку Отключить все — все службы от сторонних разработчиков будут деактивированы
* для отключения программ в меню Автозагрузка перейти на вкладку Автозагрузка и нажать кнопку Открыть диспетчер задач, который загрузится на вкладке с автоматически активируемыми при включении компьютера программами;
* выделить здесь по очереди все приложения и нажать кнопку Отключить и после реального отключения закрыть диспетчер задач и нажать кнопку ОК в окне изменений конфигураций системы;

**Выбор методов решения проблем совместимости на основе анализа**

После определения несовместимых приложений необходимо принять попытку по полному устранению проблем или, хотя бы, по уменьшению последствий.

**Для обеспечения совместимости приложений необходимо выполнить:**

* + **Определить наиболее значимые приложения.**
  + **Определить проблемы совместимости.**

Уменьшение проблем с совместимостью приложения обычно зависит от различных факторов, таких как тип приложения и текущей поддержки приложения.

**Общие методы уменьшение проблем с совместимостью приложений включают:**

Изменение конфигурации существующего приложения: Вы можете использовать инструменты, Compatibility Administrator или Standard User Analyzer, для обнаружения проблемы и создания исправления данного приложения, что решит проблему совместимости

Применение обновлений или пакетов обновлений ОС: это может помочь решить многие из проблем с совместимостью и дать возможность приложению работать в новой среде операционной системы.

Апгрейд приложения до совместимого релиза: если более новая, совместимая версия приложения существует, лучшее решение — обновить до более новой версии.

Изменение конфигурации безопасности: как пример, Защищенный режим Internet Explorer может быть смягчен, добавив сайт в список надежных сайтов или выключив Защищенный режим (что не рекомендуется).

Запуск приложения в виртуализированной среде: если все другие методы недоступны, для решения проблем вы можете запустить приложение в более раннем релизе Windows, используя инструменты виртуализации, такие как PC Microsoft Virtual, Microsoft Virtual Server и т.п.

Использование функций совместимости приложения: проблемы приложения, такие как управление версиями операционной системы, могут быть смягчены, запуском приложения в режиме эмуляции (режим совместимости с предыдущей версией ОС).

Выбор другого приложения, которое выполняет ту же самую функцию, но не имеет проблем с совместимостью: если другое совместимое приложение доступно, вы можете использовать его.

1. **Выявление причин возникновения проблем совместимости ПО. Выбор методов выявления совместимости.**

При переходе на новую версию операционной системы неизбежно возникает проблема совместимости приложений, которая должна быть решена в ходе обслуживания компьютеров.

Проблема совместимости чаще всего возникает, когда приложение некорректно обращается к функциям операционной системы.

С учетом возможных проблем совместимости используемых приложений и новой версии операционной системы рекомендуется в обязательном порядке включить в подготовку к обновлению ОС на компьютерах предприятия следующие этапы:

**обязательная проверка используемого ПО** на совместимость с новой операционной системой и сбор информации о выявленных в процессе этого тестирования проблема

**Анализ приложений**, у которых выявлены проблемы совместимости.

В процессе этого анализа нужно выяснить, насколько необходимы предприятию данные приложения, выпущены ли для них новые версии, в которых решена проблема совместимости, и в случае отсутствия таких версий - поиск инструментов для решения проблем совместимости

**тестирование инструментария** для решения проблем совместимости, желательно не слишком усложняющего обслуживание компьютеров.

**Тестирование на совместимость** позволяет проверить приложение в различных браузерах, на разных ОС, с различным программным обеспечением и аппаратными средствами.

**Инсталляционное тестирование**

В процессе инсталляционного тестирования проверяется корректность установки и удаления программного продукта в среде, максимально приближенной к эксплутационной. Об этом аспекте корректной работы программного обеспечения очень часто просто забывают (и напрасно). Правильно выполненная установка программы — необходимое условие её корректной дальнейшей работы. Проверка правильности установки должна быть обязательным элементом проекта по тестированию любого продукта. Если программу невозможно корректно установить, и при этом что-то не будет работать или будет работать неправильно, работа по тестированию самого программного тестирования бессмысленна. Почему? Потому что заказчику не нужен продукт, который даже невозможно установить. Если пользователь уже на этапе установки сталкивается с проблемами в разработанном программном продукте, что он подумает о самом программном продукте?

**Регрессионное тестирование**

Повторное выполнение тестов для проверки того, что изменения, внесённые в программу в результате разработки новой или изменения существующей функциональности, устранения ошибок, не повлияли на функциональность, которая не изменялась (т.е. текущая версия ведёт себя идентично предыдущей, за исключением измененных областей).

**Тестирование новой функциональности**

В данном виде тестирования акцент делается на тестировании новой функциональности, появившейся в конкретном выпуске программного продукта.

**Конфигурационное тестирование**

С помощью конфигурационных тестов проверяется совместимость продукта с различным программным (software) и аппаратным (hardware) обеспечением. Как правило, программный продукт делается с тем расчётом, чтобы он сразу работал в максимально разнообразной внешней среде. Если же речь идёт о «коробочном продукте», то фактор совместимости приобретает ещё более важное значение. Для того, чтобы выяснить реакцию продукта на окружение и соседство с другим программным обеспечением, и проводяданные тесты.

**Тестирование совместимости**

Тестирование совместимости помогает убедиться в функциональных возможностях и надёжности работы продукта в поддерживаемых браузерах (если речь идет о Web-приложениях) и операционных системах. Также может проверяться работоспособность продукта при использовании различных аппаратных платформ.

**Тестирование удобства эксплуатации**

Тестирование интерфейса человек/машина производится в отношении таких моментов как внешний вид пользовательского интерфейса, удобство навигации (преимущественно для Web-сайтов). Практичность и удобство использования – очень важные характеристики программного продукта. Например, программа может вполне соответствовать всем предъявляемым к ней требованиям с точки зрения функциональности. Но функции реализованы неудобно: некоторые шаги приходится повторять много раз, тогда как по логике достаточно выполнить однажды; расположение элементов интерфейса нелогично, программа быстро вызывает утомление и т.д. Для выявления такого рода недочётов и применяют тесты на удобство использования. Часто эта группа тестов относится к категории некритичных, но когда речь идёт, например, о рыночном готовом продукте, пренебрегать удобством эксплуатации весьма опасно.

1. **Проблемы перехода на новые версии программ. Мастер совместимости программ.**

**1.Апгрейд.** На данном этапе происходит обмен одного из имеющихся у пользователя продукта на новый. Это предложение действует только для обладателей лицензионной системы. «Старая» программа остается у клиента и ею можно пользоваться во время «переходного» периода.

**2. Установка**.Предусматривает проведение самих работ, результат – рабочая новая версия ПО, которую можно сразу же использовать.

**3. Перенос данных**.Необходимый этап в рамках, которого информация со старой версии переносится на новую.

4. **Освоение**.Научиться работать с новой системой можно:

самостоятельно, для этого выпускается специализированная литература;

в специально созданных учебных центрах;

индивидуально, с квалифицированным специалистом

Несмотря на все преимущества новой версии, многие фирмы продолжают работать на базе старой.

**Для перехода к более поздней версии ПО**

**и трансформации предприятия ИТ-отделам и организации необходимо провести:**

* Сокращение риска благодаря устранению опасных пробелов в системе безопасности и несоответствий, возникающих в результате продолжительного использования старого ПО;
* Увеличение продуктивности  ИТ-отделов благодаря использованию виртуальных десктопов для перехода на последнее ПО;
* Экономия за счет доставки виртуальных приложений и десктопов на существующее  аппаратное обеспечение
* Модернизация среды  и предоставление сотрудникам возможности достигать большей эффективности при работе в любое время, в любом месте и на любом устройстве.

**Безопасность.**Окончание поддержки версии Windows также означает прекращение выпуска обновлений безопасности.

ПК, которые на момент окончания поддержки продолжат использовать

эту ОС, окажутся гораздо более уязвимыми к угрозам, вирусам, шпионскому ПО и другим вредоносным действиям.

**Соответствие стандартам**. Как внутренние, так и внешние аудиторские проверки будут требовать от предприятий использования версий ПО с действующей поддержкой и со всеми установленными обновлениями; Несоблюдение этих стандартов может легко привести к  ущербу для бизнеса, повышенным ставкам за страхование предпринимательской деятельности и даже пробелам в системе страхового обеспечения бизнеса.

**Увеличивающийся уровень сложности**. Многочисленные зависимости от платформ и технологии других компаний, обычно используемые в среде предприятия, потенциально усложняют предстоящий переход к более поздним версиям ПО (Windows), а уровень сложности будет возрастать по мере переноса этих элементов инфраструктуры в среды, несовместимые с Windows

**Каким образом осуществить переход к более поздней версии ПО в кратчайшие сроки и с меньшими затратами?**

**Исключите** **проблемы,** **связанные с  совместимостью приложений.** (Нужно определять место каждого приложения с точки зрения совместимости. Пользователи смогут без проблем получать доступ как к устаревшим приложениям, так и к обновленным приложениям в течение всего процесса.)

**Обеспечьте** **защиту пользовательских  данных  в  кратчайшие  сроки.** (Пока данные остаются на конечных устройствах на базе старой ОС, организации ежедневно подвергаются риску нарушения безопасности, воздействия вредоносных программ, несоответствия нормативным требованиям и наложения штрафов контролирующими органами. )

**Доставляйте  приложения  на  существующее  аппаратное  обеспечение  и  сокращайте  расходы (**Чтобы не  поддерживать практически бесконечное разнообразие пользовательских конфигураций (каждая из которых имеет свой собственный набор потенциальных конфликтов и сбоев), необходимо использовать единый стандартный образ для поддержки сотрудников вне зависимости от используемого ими устройства.)

**От миграции к  трансформации (**После решения вопроса перехода к более поздней версии ПО, важно обратить внимание на главные стратегические инициативы, касающиеся мобильности предприятия, и добиться прогресса в этом направлении.)

**Мастер совместимости программ**. Режим совместимости программ Windows 10 позволяет запускать ПО на компьютере, которое нормально работало только в предыдущих версиях Windows, а в последней ОС программа не запускается или работает с ошибками. В этой инструкции о том, как включить режим совместимости с Windows 8, 7, Vista или XP в Windows 10 для исправления ошибок запуска программ.

**Включение режима совместимости через свойства программы или ярлыка**. Первый способ включить режим совместимости в Windows 10 очень просто — кликните правой кнопкой мыши по ярлыку или исполняемому файлу программы, выберите пункт «Свойства» и откройте, если таковая обнаружится, вкладку «Совместимость».

1. **Инструментарий учета аппаратных компонентов.**

Включение режима совместимости через свойства программы или ярлыка

• Первый способ включить режим совместимости в Windows 10 очень просто — кликните правой кнопкой мыши по ярлыку или исполняемому файлу программы, выберите пункт «Свойства» и откройте, если таковая обнаружится, вкладку «Совместимость».

• Все, что останется сделать — задать параметры режима совместимости: указать версию Windows, в которой программа запускалась без ошибок. При необходимости, включите запуск программы от имени администратора или в режиме меньшего разрешения экрана и пониженной цветности (для совсем старых программ). Затем примените сделанные настройки. В следующий раз программа будет запущена с уже измененными параметрами.

Как включить режим совместимости программ с предыдущими версиями ОС в Windows 10 через устранение неполадок

• Чтобы запустить настройку режима совместимости программ, потребуется запустить специальное средство устранения неполадок Windows 10 «Выполнение программ, предназначенных для предыдущих версий Windows».

• Сделать это можно либо через пункт панели управления «Устранение неполадок» (панель управления можно открыть через правый клик по кнопке Пуск. Чтобы увидеть пункт «Устранение неполадок» вверху справа в поле «Просмотр» должно стоять «Значки», а не «Категории»), либо, что быстрее, через поиск в панели задач.

• Запустится средство устранения проблем с совместимостью старых программ в Windows 10 Имеет смысл использовать пункт «Запуск от имени администратора» при его использовании (это позволит применить параметры к программам, находящимся в папках с ограниченным доступом). Нажмите «Далее».

• После некоторого ожидания, в следующем окне будет предложено выбрать программу, с совместимостью которой есть проблемы. Если вам нужно добавить собственную программу (например, в списке не будут отображаться portable-приложения), выберите пункт «Нет в списке» и нажмите «Далее», после чего задайте путь к исполняемому exe файлу программы.

• После выбора программы или указания ее расположения, будет предложено выбрать режим диагностики. Для ручного указания режима совместимости с конкретной версией Windows, нажмите «Диагностика программы».

• В следующем окне будет предложено указать проблемы, которые были замечены при запуске вашей программы в Windows 10 Выбираем «Программа работала в предыдущих версиях Windows, но не устанавливается или не запускается сейчас» (или другие варианты, по ситуации).

• В следующем окне потребуется указать, с какой версией ОС включить совместимость — Windows 7, 8, Vista и XP. Выберите свой вариант и нажмите «Далее».

• В следующем окне, чтобы завершить установку режима совместимости, требуется нажать «Проверить программу». После ее запуска, проверки (которую вы выполняете сами, необязательно) и закрытия, нажмите «Далее».

• И, в завершение, либо сохраните параметры совместимости для этой программы, либо используйте второй пункт, если ошибки остались — «Нет, попытаться использовать другие параметры». Готово, после сохранения параметров программа будет работать в Windows 10 в выбранном вами режиме совместимости.

Для упрощения процесса перехода на новые версии программ был разработан инструментарий, называемый мастером совместимости программ. Основная задача мастера совместимости программ - проверить, будет ли новая версия программы работать на вашем компьютере без проблем и совместима ли она с вашим аппаратным и программным обеспечением.

1. **Анализ приложений с проблемами совместимости.**

Основная причина неработоспособности программных продуктов на разных ОС – это разные рабочие библиотеки системы.

• В каждой операционной системе они имеют разное размещение директорий (да и содержимого в конечном итоге).

• Самый распространенный метод решения проблем совместимости – это использование DLL (динамически загружаемых библиотек), которые перехватывают «проблемные» вызовы системных функций.

• Эти библиотеки нередко называют «системными заплатками».

Анализ приложений с проблемами совместимости позволяет выявить их основные причины и найти оптимальные решения. Для этого можно использовать различные инструменты, такие как отладчики, профилировщики и лог-анализаторы. При проведении анализа необходимо обратить внимание на следующие аспекты:

- Используемые версии операционной системы и комплекта программного обеспечения;

- Список используемых библиотек и их версии;

- Необходимые аппаратные требования (если такие есть);

- Особенности API и его изменения.

1. **Использование динамически загружаемых библиотек**.

Основная причина неработоспособности программных продуктов на разных ОС – это разные рабочие библиотеки системы.

Самый распространенный метод решения проблем совместимости – это использование DLL (динамически загружаемых библиотек), которые перехватывают «проблемные» вызовы системных функций. Эти библиотеки нередко называют «системными заплатками».

Библиотеками называют «сборники» подпрограмм или объектов, ориентированных на решение набора близких по тематике задач.

**Статические(LIB**) библиотеки — это набор исходных кодов, подключаемых программистом в свою программу, либо набор заранее скомпилированных объектных файлов, связываемых вместе на этапе компиляции.

**Динамические (DLL)** библиотеки загружаются ОС по «требованию» запущенной программы уже в  ходе ее выполнения.

Использовать динамическую библиотеку в  программе можно либо неявным, либо явным связыванием.

Неявное связывание подразумевает использование библиотеки импорта для определения адресов функций, предоставляемых библиотекой.

При явном связывании исполняемый файл, использующий библиотеку DLL, должен делать вызовы функции для *явной загрузки и выгрузки библиотеки DLL и  осуществления доступа к  экспортированным функциям библиотеки DLL*Использование динамически загружаемых библиотек.

Динамические библиотеки представляют собой исполняемые модули, содержащие функции и  данные, которые могут быть использованы другими модулями (приложениями или динамическими библиотеками).

Применяют два метода вызова функции из DLL.

**1. Динамическое связывание во время загрузки**

В  этом случае модуль осуществляет явный вызов экспортируемой функции так, будто она является его локальной функцией.

**2. Динамическое связывание во время выполнения программы**).

В  этом случае вызывающее приложение должно использовать специальные API-функции для загрузки DLL, выгрузки DLL и  получения адреса экспортируемой функции.

1. **Механизм решения проблем совместимости на основе «системных заплаток». (19 +)**

**Системные заплатки (patches**) — это изменения в программном коде, которые вносятся разработчиком или поставщиком операционной системы для решения различных проблем, включая проблемы совместимости. В современных ИТ-системах, где используется множество разнообразных программных и аппаратных компонентов, проблемы совместимости являются обычным явлением. Механизм решения таких проблем на основе системных заплаток позволяет оперативно устранять несовместимости и поддерживать стабильность системы.

**Шаги решения проблем совместимости на основе системных заплаток**

-Выявление и анализ проблемы совместимости

-Разработка и тестирование системной заплатки

-Распространение и установка заплатки

-Проверка корректности и эффективности заплатки

**1. Выявление и анализ проблемы совместимости**

Выявление проблемы:

Отчеты пользователей: Пользователи могут сообщать о проблемах совместимости через систему поддержки, форумы или напрямую разработчикам.

Мониторинг и логирование: Системы мониторинга и логирования могут автоматически выявлять аномалии или ошибки, связанные с совместимостью.

Тестирование: Регулярное тестирование программного обеспечения на разных конфигурациях позволяет выявить потенциальные проблемы до их появления у конечных пользователей.

Анализ проблемы:

Диагностика: Изучение логов ошибок и проведение диагностических тестов для определения точной причины проблемы.

Анализ конфигурации: Исследование различных конфигураций систем, на которых возникает проблема, для выявления общих факторов.

Изучение исходного кода: Анализ исходного кода программного обеспечения для обнаружения участков, вызывающих несовместимость.

**2. Разработка и тестирование системной заплатки**

Разработка заплатки:

Исследование и проектирование: Определение наилучшего подхода к устранению проблемы, разработка архитектуры решения.

Программирование: Написание кода для исправления проблемы, внесение изменений в существующий код.

Документация: Подготовка технической документации, описывающей внесенные изменения и их влияние на систему.

Тестирование заплатки:

Юнит-тесты: Проведение модульных тестов для проверки корректности отдельных частей кода.

Интеграционные тесты: Проверка взаимодействия заплатки с другими компонентами системы.

Ручное тестирование: Тестирование заплатки в условиях, максимально приближенных к рабочим, с участием тестировщиков.

Бета-тестирование: Внешнее тестирование заплатки ограниченным кругом пользователей перед широким распространением.

**3. Распространение и установка заплатки**

Распространение заплатки:

Объявление: Информирование пользователей о доступности новой заплатки через официальные каналы связи, такие как веб-сайты, рассылки или уведомления в самой системе.

Размещение: Публикация заплатки на официальных серверах обновлений, репозиториях или других каналах распространения.

Установка заплатки:

Автоматическое обновление: Настройка систем на автоматическую установку заплаток через средства обновления, такие как Windows Update, apt-get, yum и другие.

Ручная установка: Предоставление инструкций и пакетов для ручной установки пользователями или администраторами.

Контроль установки: Мониторинг процесса установки для предотвращения сбоев и обеспечения успешного применения заплатки.

**4. Проверка корректности и эффективности заплатки**

Проверка корректности:

Мониторинг: Наблюдение за системой после установки заплатки для выявления новых ошибок или несовместимостей.

Обратная связь: Сбор отзывов от пользователей о работе системы после применения заплатки.

Анализ метрик: Сравнение ключевых показателей производительности и стабильности системы до и после установки заплатки.

Эффективность заплатки:

Оценка исправлений: Проверка, устранила ли заплатка заявленные проблемы и как это отразилось на работе системы.

Валидация: Подтверждение корректности работы всех функциональных возможностей системы после применения заплатки.

Документирование результатов: Подготовка отчетов о результатах установки заплатки и их влияние на систему.

Заключение

Решение проблем совместимости на основе системных заплаток является важным механизмом поддержания стабильности и безопасности современных ИТ-систем. Процесс включает несколько ключевых этапов, начиная с выявления проблемы и заканчивая проверкой эффективности примененной заплатки. Каждый из этапов требует внимательного подхода и координации между разработчиками, тестировщиками и пользователями, чтобы обеспечить успешное и своевременное устранение проблем совместимости.

1. **Разработка модулей обеспечения совместимости.**

**Разработка модулей обеспечения совместимости** - это процесс создания дополнительного программного кода, который позволяет решить проблемы совместимости и обеспечить корректное функционирование приложения на различных платформах и с разными комплектами программного обеспечения.

**Правила создания программ:**

1)к разработке программ необходимо привлекать команду программистов;

2)программу необходимо разбить на несколько независимых программных модулей, каждый из которых должен выполнять определенную функцию;

3)необходимо обеспечить совместимость программных модулей. Разбиение программы на модули — типичная задача обеспечения совместимости.

**Технология реализации модульного принципа программирования**.

*1. Задача разбивается на относительно простые и  самостоятельные фрагменты.* Критериями качества разбиения являются:

решение своей задачи до конца каждым выделяемым фрагментом;

возможность запрограммировать модуль в заданный срок;

минимум связей с остальными модулями;

проверяемость входных данных и результатов выполнения модуля. К правильности и полноте входных данных должны быть сформулированы четкие требования, выполнение которых должно быть проверено на входе модуля. Ситуации получения неверных данных должны быть учтены и  обработаны. Решая определенную задачу до конца, модуль должен выдавать результаты, правильность которых также можно логически проверить

*2. Формируются требования к модулям*.

Для каждой конкретной задачи выполняемой программы (варианта использования) должна существовать цепочка последовательно выполняемых модулей, полностью решающая данную задачу.

В простейшем случае цепочка может состоять из одного модуля.

*3. Разрабатывается межмодульный интерфейс  —* правилавызова модулей, передачи им параметров и данных.

Информация, необходимая для работы большинства модулей, организуется в  виде общих областей, глобальных переменных, массивов или баз данных.

После завершения третьего этапа формируются задания на программирование.

*4. Проводится комплексная отладка* (после написания отдельных модулей). В  ходе отладки проверяется возможность совместной работы модулей, правильность решения всех задач программного продукта.

**Создание программных модулей.**

Это процесс реорганизации программы в целях объединения ее взаимосвязанных частей в отдельном модуле. После этого легче удалить избыточность в  соответствующих компонентах, оптимизировать взаимосвязи и  упростить интерфейс всей программы

1. **Создание в системе виртуальной машины для исполнения приложений.**

*Технологии виртуализации*

Виртуализация ресурсов физического сервера позволяет гибко распределять их между приложениями, каждое из которых при этом "видит" только предназначенные ему ресурсы и "считает", что ему выделен отдельный сервер, т. е. в данном случае реализуется подход "один сервер — несколько приложений", но без снижения производительности, доступности и безопасности серверных приложений.

Кроме того, решения виртуализации дают возможность запускать в разделах разные ОС с помощью эмуляции их системных вызовов к аппаратным ресурсам сервера.

***Виртуализация подразумевает запуск на одном физическом компьютере нескольких виртуальных компьютеров***

История развития технологий виртуализации насчитывает более сорока лет. Компания IBM была первой, кто задумался о создании виртуальных сред для различных пользовательских задач.

В 60-х годах прошлого века виртуализация представляла чисто научный интерес и была оригинальным решением для изоляции компьютерных систем в рамках одного физического компьютера.

В настоящее время на рынке указанной категории ПО лидируют две компании: Microsoft и VMware, Inc. — обе они производят средства управления виртуаль-ными машинами и  для рабочих станций, и для серверов.

Корпорация **Microsoft** вышла на рынок средств виртуализации в 2003 г. с приобретением компании Connectiх, выпустив свой первый продукт Virtual PC для настольных ПК.

С тех пор она последовательно наращивала спектр предложений в этой области и на сегодня почти завершила формирование виртуализацион-ной платформы, в состав которой входят такие решения как

Windows 2008 Server R2 c компонентом Hyper-V,

Microsoft Application Virtualization (App-v),

Microsoft Virtual Desktop Infrastructure (VDI),

Remote Desktop Services,

System Center Virtual Machine Manager.

*Преимущества виртуализации*

***1. Эффективное использование вычислительных ресурсов****.*

Вместо 3х, а то 10 серверов, загруженных на 5-20% можно использовать один, используемый на 50-70%.

***2.Сокращение расходов на инфраструктуру****:*

Виртуализация позволяет сократить коли-чество серверов и связанного с ними ИТ-оборудования в информационном центре. В результате этого потребности в обслуживании, электропитании и охлаждении материальных ресурсов сокращаются, и на ИТ затрачивается гораздо меньше средств.

***3. Снижение затрат на программное обеспечение****.*

Некоторые производители программного обеспечения ввели отдельные схемы лицензирования специально для виртуальных сред. Так, например, покупая одну лицензию на Microsoft Windows Server 2008 Enterprise, вы получаете право одновременно её использовать на 1 физическом сервере и 4 виртуальных (в пределах одного сервера), а Windows Server 2008 Datacenter лицензируется только на количество процессоров и может использоваться одновременно на неограниченном количестве виртуальных серверов.

***4. Повышение гибкости и скорости реагирования системы***:

Виртуализация предлагает новый метод управления ИТ-инфраструктурой и помогает ИТ-администраторам затрачивать меньше времени на выполнение повторяющихся заданий — например, на инициацию, настройку, отслеживание и техническое обслуживание.

***5. Несовместимые приложения могут работать на одном компьютере****.*

При использовании виртуализации на одном сервере возможна установка linux и windows серверов, шлюзов, баз данных и прочих абсолютно несовместимых в рамках одной не виртуализированной системы приложений.

***6. Повышение доступности приложений и обеспечение непрерывности работы предприятия****:*

Благодаря надежной системе резервного копирования и миграции виртуальных сред целиком без перерывов в обслуживании вы сможете сократить периоды планового простоя и обеспечить быстрое восстановление системы в критических ситуациях.

***7.Возможности легкой архивации****.*

Поскольку жесткий диск виртуальной машины обычно представляется в виде файла опреде-ленного формата, расположенный на каком-либо физическом носителе, виртуализация дает возможность простого копирования этого файла на резервный носитель как средство архиви-рования и резервного копирования всей виртуальной машины целиком.

***8. Повышение управляемости инфраструктуры****:*

использование централизованного управления виртуальной инфраструктурой позволяет сократить время на администрирование серверов, обеспечивает балансировку нагрузки и "живую" миграцию виртуальных машин.

***Виртуальная машина*** *—* это полностью изолированный программный контейнер, который работает с собственной ОС и приложениями, подобно физическому компьютеру.

* **Совместимость**. Виртуальные машины, как правило, совместимы со всеми стандартными компьютерами.
* **Изолированность**. Виртуальные машины полностью изолированы друг от друга, как если бы они были физическими компьютерами Виртуальные машины могут использовать общие физические ресурсы одного компьютера и при этом оставаться полностью изолированными друг от друга, как если бы они были отдельными физическими машинами.
* **Инкапсуляция**. Виртуальные машины полностью инкапсулируют вычислительную среду. Виртуальная машина представляет собой программный контейнер, связывающий, или "инкапсулирующий" полный комплект виртуальных аппаратных ресурсов, а также ОС и все её приложения в программном пакете.
* **Независимость от оборудования**. Виртуальные машины полностью независимы от базового физического оборудования, на котором они работают.

**Основные разновидности виртуализации**

* виртуализация серверов (полная виртуализация и паравиртуализация)
* виртуализация на уровне операционных систем,
* виртуализация приложений,
* виртуализация представлений.

**Полная виртуализация** – Виртуализация при которой используются не модифицированные экземпляры гостевых операционных систем, а для поддержки работы этих ОС служит общий слой эмуляции их исполнения поверх хостовой ОС, в роли которой выступает обычная операционная система.

**Паравиртуализация** – Виртуализация при которой производится модификация ядра гостевой ОС выполняется таким образом, что в нее включается новый набор API, через который она может напрямую работать с аппаратурой, не конфликтуя с другими виртуальными машинами.

**Виртуализация на уровне ОС** – Вид виртуализации, который подразумевает использование одного ядра хостовой ОС для создания независимых параллельно работающих операционных сред.

**Виртуализация серверов** - это запуск на одном физическом сервере нескольких виртуальных серверов. Виртуальные машины или сервера представляют собой приложения, запущенные на хостовой операционной системе, которые эмулируют физические устройства сервера. На каждой виртуальной машине может быть установлена операционная система, на которую могут быть установлены приложения и службы.

**Виртуализация приложений** – вид виртуализации, которая подразумевает применение модели сильной изоляции прикладных программ с управляемым взаимодействием с ОС, при которой виртуализируется каждый экземпляр приложений, все его основные компоненты: файлы (включая системные), реестр, шрифты, INI-файлы, COM-объекты, службы. Приложение исполняется без процедуры инсталляции в традиционном ее понимании и может запускаться прямо с внешних носителей.

**Виртуализация представлений (рабочих мест)** Виртуализация представлений имеет место, когда сервер предоставляет свои ресурсы клиентам, причем клиентское приложение выполняется на этом сервере, а клиент получает только представление.

1. **Изменение настроек по умолчанию в образе. Подключение к сетевому ресурсу.**

Для использования сетевого ресурса необходимо получить дос­туп к нему.

**Метод доступа** – набор правил, которые определяют, как компьютер должен отправлять и принимать данные по сетевому кабелю.

Компьютеры подключаются к сети по очереди на короткое время. Несколько компьютеров могут совместно использовать кабель. Если два компьютера попытаются передать данные одновременно, их пакеты столкнутся и будут повреждены.

**Коллизия** – наложение двух и более пакетов от компьютеров, пытающихся передать пакет в один и тот же момент времени.

**Существуют четыре метода доступа:**

1. Множественный доступ с контролем несущей и обнаружением кол­лизий – все компьютеры в сети прослушивают ка­бель, стремясь обнаружить передаваемые данные. Ком­пьютер может начать передачу только тогда, когда убедится, что ка­бель свободен. Если возникает коллизия, то компьютеры приостанавливают передачу на случайный интервал времени, а затем вновь стараются наладить связь.

2. Множественный доступ с контролем несущей и предотвращением коллизий– каждый компьютер перед передачей данных в сеть сигнализирует о своем намерении, поэтому остальные компьютеры «узнают» о гото­вящейся передаче и могут избежать коллизий. Однако широковеща­тельное оповещение увеличивает общий трафик и уменьшает пропус­кную способность сети. Поэтому CSMA/CA работает медленнее, чем CSMA/CD.

3. Доступ с передачей маркера – пакет особого типа, маркер, циркулирует от компьюте­ра к компьютеру. Чтобы послать данные в сеть, любой компьютер должен сначала «дождаться» прихода свободного маркера и «захва­тить» его. Захватив маркер, компьютер может передавать данные. Когда какой-либо компьютер наполнит маркер своей информацией и пошлет его по сетевому кабелю, другие компьютеры уже не смогут передавать данные, так как в каждый момент времени только один компьютер использует маркер. В сети не возникает ни состязания, ни коллизий, ни временных задержек.

4. Доступ по приоритету запроса – концентраторы управляют доступом к кабелю, последовательно опрашивая каждый узел в сети и выявляя запросы на передачу. Концентратор должен знать все адреса связи и узлы и про­верять их работоспособность.

**25. Настройка обновлений программ. Обновление драйверов.**

**Настрайка обновлений программ Windows**

**Ручное обновление**

Многие программы предусматривают автообновление или, по крайней мере, проверку наличия новой версии. Данная опция обычно находится в разделе меню «Справка», откуда можно сразу скачать новую версию. Также обновиться можно с сайта разработчиков, адрес которого обычно указан в разделе меню «О приложении».

**Необходимость обновления стороннего ПО**

В новых версиях программ производители добавляют дополнительные функции, исправляют найденные ошибки, исправляют конфликты с «железом».

**Ручное обновление**

Многие программы предусматривают автообновление или, по крайней мере, проверку наличия новой версии. Данная опция обычно находится в разделе меню «Справка», откуда можно сразу скачать новую версию.

**Обновление с помощью специальных утилит**

Процесс проверки наличия актуальных версий установленных программ и загрузку их обновлений станет гораздо проще с помощью специальных приложений. Рассмотрим некоторые из них.

**Kaspersky Software Updater**

Эта бесплатная утилита от Лаборатории Касперского имеет минимальный интерфейс и простое управление. После ее установки и запуска просто нажимаем кнопку «Искать обновления» и ждем завершения поиска.

Найденные обновления будут представлены в виде списка, где можно будет непосредственно запустить апдейт.

Утилита добавляется в автозагрузку и в дальнейшем следит за обновлениями самостоятельно.

**Обновление драйвера и его отмена**

Одной из наиболее распространенных причин непо­ладок оборудования являются устаревшие или по­врежденные драйверы. Для каждой версии ОС требуются особые драйверы, поэтому некорректное обновление драйвера может повлечь за собой сбой устройства либо даже вывести его из строя. Напри­мер, после обновления Windows 98 до Windows XP может перестать работать драйвер принтера, кото­рый не был специально разработан для Windows XP. Умение находить и обновлять драйверы, а также от­менять их установку поможет вам в решении многих проблем с настройкой оборудования.

**Поиск драйверов.**

Ниже перечислены четыре основных источника в том порядке, в котором рекомендуется производить поиск требуемого драйвера.

- Компакт-диск из комплекта устройства. Большинство периферийных устройств по­ставляется с диском, на котором есть все не­обходимые драйверы для различных версий ОС. Это наилучший источник драйверов для нового оборудования.

- Сайт Windows Update. Компания Microsoft публикует последние версии драйверов раз­личных устройств, а также обновления самой ОС и ее приложений. Проанализировав ваш компьютер, программа Windows Update пред­ложит список подходящих для него драйве­ров и обновлений.

- Веб-сайт изготовителя устройства. Боль­шинство производителей оборудования пуб­ликуют обновляемые ими драйверы для своей продукции на принадлежащих им веб-сайтах.

- Установочный диск Windows XP Professional содержит библиотеку драйверов для многих устройств входящих в список поддерживаемого оборудования «HCL»

Установка новой версии драйвера для уже имеюще­гося оборудования называется обновлением драйве­ра. Windows XP Professional значительно облегчает эту процедуру.

**Чтобы обновить драйвер, необходимо выполнить следующие действия:**

1. Найдите требуемый драйвер на одном из ранее перечисленных источников.

2. Раскройте меню Пуск (Start), щелкните правой кнопкой мыши на пункте Мой компьютер (My Computer) и выберите в контекстном меню команду Управление (Manage).

3. В окне Управление компьютером выберите в списке слева пункт Диспетчер устройств.

4. Щелкните мышью на знаке «+» рядом с типом устройства, драйвер для которого нужно обно­вить.

5.Выполните двойной щелчок мышью на названии нужного устройства откроется окно его свойств. Перейдите в нем на вкладку Драйвер (Driver).

6. Щелкните мышью на кнопке Обновить (Update Driver) запустится Мастер обновление оборудования (Hardware Update Wizard).

7. Если новый драйвер имеется на компакт-диске, щелкните мышью на кнопке Далее; иначе выберите опцию Установка из указанного места и вновь щелкните мышью на кнопке Далее .

8. Пометьте флажок Включить следующее место поиска, щелкните мышью на кнопке Обзор (Browse), найдите каталог с требуемым драйвером и щелк­ните мышью на кнопке Далее (Next).

После этого драйвер будет установлен, а устрой­ство должно начать нормально работать (возможно, для этого придется включить его вручную в окне свойств в Диспетчере устройств).

**Отмена установки драйвера**

Иногда после обновления драйвера устройство начи­нает давать сбои, а то и вовсе перестает работать. В этом случае необходимо отменить («откатить») уста­новку драйвера. После этого система возвращается к использованию прежнего драйвера. В Windows XP Professional отменить установку драйвера достаточно просто, поскольку система сохраняет прежние драй­веры.

Чтобы отменить установку драйвера, необходимо выполнить следующие действия:

1. В окне свойств устройства (открытом при помо­щи Диспетчера устройств) перейдите на вкладку Драйвер.

2. Щелкните мышью на кнопке «Откатить» (Roll Back Driver).

1. **Решение проблем конфигурации с помощью групповых политик.**

Групповые политики нужны для управления операционной системы Windows. Они применяются во время персонализации интерфейса, ограничения доступа к определенным ресурсам системы и многого другого. Используют данные функции преимущественно системные администраторы. Они создают однотипную рабочую среду на нескольких компьютерах, ограничивают доступ пользователям.

Запуск редактора групповой политики

Переход к среде работы с параметрами и настройками осуществляется за несколько простых действий. Вам только необходимо:

1.Зажать клавиши Win + R, чтобы открыть «Выполнить».

2.Напечатать в строке gpedit.msc и подтвердить действие, нажав «ОК».

**Работа в редакторе**

Разделяется главное окно управления на две части. Слева располагается структурированные категории политик. Они в свою очередь делятся еще на две различные группы – настройка компьютера и настройка пользователя.

Выберите, «Административные шаблоны» в «Конфигурации пользователя» и перейдите в папку «Меню «Пуск» и диспетчер задач». Теперь справа отобразятся параметры и их состояния. Нажмите на любую строку, чтобы открыть ее описание.

**Настройки политики**

Каждая политика доступна для настройки.

Стандартное простое окно имеет три различных состояния, которые настраиваются пользователем. Если точка стоит напротив «Не задано», то политика не действует. «Включить» – она будет работать и активируются настройки. «Отключить» – находится в рабочем состоянии, однако параметры не применяются.

**Фильтры политик**

Минусом редактора является отсутствие функции поиска. Существует множество различных настроек и параметров, их больше трех тысяч, все они разбросаны по отдельным папкам, а поиск приходится осуществлять вручную.

Например, в разделе «Административные шаблоны», в любой конфигурации, находятся политики, которые никак не связаны с безопасностью. В этой папке находится еще несколько папок с определенными настройками, однако можно включить полное отображение всех параметров, для этого нужно нажать на ветвь и выбрать пункт в правой части редактора «Все параметры», что приведет к открытию всех политик данной ветви.

**Экспорт списка политик**

Если все-таки появляется необходимость найти определенный параметр, то сделать это можно только путем экспорта списка в текстовый формат, а потом уже через, например Word, осуществлять поиск. В главном окне редактора есть специальная функция «Экспорт списка», он переносит все политики в формат TXT и сохраняет в выбранном месте на компьютере.

1. **Тестирование на совместимость в безопасном режиме. Восстановление системы.**

Безопасный режим — это режим запуска системы Windows, предназначенный для устранения неполадок, в котором используется ограниченный набор служб и компонентов. Загружаются только базовые файлы и драйверы, необходимые для запуска Windows. В углах экрана отображаются слова Безопасный режим, соответствующие используемому режиму Windows.

Если проблем нет в безопасном режиме, то причина может быть в параметрах по умолчанию и базовых драйверах. Если причина неизвестна, используйте метод исключения. Запустите все программы, в том числе из автозагрузки, чтобы найти проблемную.

Если компьютер автоматически, без запроса, запускается в безопасном режиме, возможно, проблема препятствует обычной загрузке Windows.

**Запуск компьютера в безопасном режиме**

В безопасном режиме запуск Windows выполняется с использованием ограниченного набора файлов и драйверов. В безопасном режиме не производится запуск программ, загружаемых при запуске, и устанавливается только базовый набор драйверов, необходимый для запуска Windows.

Безопасный режим полезен для устранения неполадок, вызванных программами и драйверами, которые могут неправильно загружаться или препятствовать правильному запуску Windows. Если недавно установленная программа, устройство или драйвер препятствуют правильному запуску Windows, можно запустить компьютер в безопасном режиме и удалить программу, вызывающую проблему. Дополнительные сведения об устранении неполадок в безопасном режиме см. в разделе Средства диагностики, которые следует использовать в безопасном режиме.

Извлеките из компьютера все гибкие диски, компакт-диски и DVD-диски, а затем перезагрузите компьютер.

Нажмите кнопку Пуск, щелкните стрелку рядом с кнопкой Завершение работы, затем нажмите кнопку Перезагрузка.

**Выполните одно из следующих действий**.

Если установлена одна операционная система, во время перезагрузки компьютера нажмите и удерживайте клавишу F8. Клавишу F8 необходимо нажать до того, как появится логотип Windows. Если появилась эмблема Windows, дождитесь появления приглашения Windows для входа в систему, затем завершите работу и перезагрузите компьютер.

Если на компьютере установлено несколько операционных систем, при помощи клавиш со стрелками выберите операционную систему, которую необходимо запустить в безопасном режиме, и затем нажмите клавишу F8.

С помощью клавиш со стрелками выберите на экране Дополнительные варианты загрузки необходимый вариант безопасного режима и нажмите клавишу ВВОД.

Войдите в систему под учетной записью пользователя с правами администратора.

Когда компьютер находится в безопасном режиме, по углам экрана отображается надпись Безопасный режим. Чтобы выйти из безопасного режима, перезагрузите компьютер и позвольте Windows запуститься в обычном режиме.

Средства диагностики, которые следует использовать в безопасном режиме

При использовании безопасного режима для устранения неполадок компьютера значительную помощь могут оказать описанные в данном разделе средства и функции.

Откройте компонент «Восстановление». Для этого нажмите кнопку Пуск и выберите пункт Панель управления. В поле поиска введите восстановление и затем щелкните пункт Восстановление.

С помощью компонента «Восстановление» можно восстановить состояние системы до состояния на определенный момент времени, а также восстановить содержимое жесткого диска из архива. Также можно переустановить Windows из образа для восстановления, расположенного на жестком диске компьютера.

Откройте панель управления. Для этого нажмите кнопку Пуск и выберите пункт Панель управления.

Панель управления служит для доступа к различным средствам изменения параметров Windows.

Откройте диспетчер устройств. Для этого нажмите кнопку Пуск, выберите последовательно компоненты Панель управления, Система и безопасность, а затем в разделе Система выберите Диспетчер устройств.‌ Если отображается запрос на ввод пароля администратора или его подтверждения, укажите пароль или предоставьте подтверждение.

Диспетчер устройств позволяет обновлять драйверы устройства и для настраивать установленное оборудование. Для использования этой программы необходимо войти в систему в качестве администратора. В противном случае изменять можно только параметры текущей учетной записи пользователя.

Откройте раздел «Просмотр событий». Для этого нажмите кнопку Пуск, выберите Панель управления, Система и безопасность, Администрирование, затем дважды щелкните Просмотр событий. Если отображается запрос на ввод пароля администратора или его подтверждения, укажите пароль или предоставьте подтверждение.

Просмотр событий служит для просмотра подробных записей о событиях системы и программ на вашем компьютере. Для использования этой программы необходимо войти в систему в качестве администратора. В противном случае изменять можно только параметры текущей учетной записи пользователя.

Откройте раздел «Сведения о системе». Для этого нажмите кнопку Пуск. В поле поиска введите Сведения о системе, а затем в списке результатов выберите пункт Сведения о системе.

Компонент «Сведения о системе» служит для просмотра подробных сведений о конфигурации оборудования, компонентах и драйверах компьютера.

Откройте окно командной строки. Для этого нажмите кнопку Пуск. В поле поиска введите Командная строка, а затем в списке результатов выберите пункт Командная строка.

Опытные пользователи могут использовать окно командной строки для вызова программ командной строки. Для использования этой программы необходимо войти в систему в качестве администратора. В противном случае изменять можно только параметры текущей учетной записи пользователя.

Откройте редактор реестра. Для этого нажмите кнопку Пуск, введите regedit в поле поиска и затем нажмите клавишу ВВОД.‌ >Если отображается запрос на ввод пароля администратора или его подтверждения, укажите пароль или предоставьте подтверждение.

Опытные пользователи могут использовать редактор реестра, чтобы вносить изменения в файлы реестра Windows. Для использования этой программы необходимо войти в систему в качестве администратора. В противном случае изменение параметров будет доступно только для вашей учетной записи пользователя.

В безопасном режиме Windows запускается только с базовым набором драйверов и служб, перечисленных в данном разделе.

**Восстановление системы.**

**Восстановление системы**— компонент ОС Windows , предназначенный для восстановления работоспособности ОС путем отката *(восстановления предыдущего состояния в  персональном компьютере системных файлов, ключей реестра, установленных программ и  т. д*.). Восстановление системы присутствует в  системах Windows XP, Windows Vista, Windows 7, Windows 8 и Windows 10.

Пользователь может создавать новую точку восстановления вручную, производить откат к  уже существующей точке или изменять конфигурацию восстановления системы.

Иногда из-за установки программы или драйвера Windows может начать работать медленно или непредсказуемо. Функция восстановления системы позволяет восстановить системные файлы и программы компьютера.

Восстановление системы использует точки восстановления для возврата системных файлов и параметров к состоянию на определенный момент времени, не влияя на личные файлы. Точки восстановления автоматически создаются еженедельно, а также перед значительными системными событиями, такими как установка программ или драйверов устройств. Точку восстановления также можно создать вручную.

В Windows 7 можно не только создавать дополнительные точки восстановления системы, но и просматривать, какие именно файлы будут удалены или добавлены при восстановлении системы.

Прежде чем запустить восстановление системы, сохраните все открытые файлы и закройте все программы. В процессе восстановления системы будет перезагружен компьютер.

■ Откройте восстановление системы, нажав кнопку Пуск. В поле поиска введите восстановление системы, а затем в списке результатов выберите Восстановление системы. Введите пароль администратора или подтверждение пароля, если отобразится соответствующий запрос.

■ Выполните указания мастера, чтобы выбрать точку восстановления и восстановить компьютер.

1. **Производительность ПК. Проблемы производительности. Анализ журналов событий.**

Производительность компьютера — это скорость обрабатываемой информации за определенное время. И чем больше скорость, тем лучше показатель производительности.

Детали, отвечающие за производительность. Самая главная деталь — это процессор компьютера, затем по значимости идет оперативная память и нужно знать, что не объем, а скорость, с кото рой данные считываются с жесткого диска и передаются процессору на обработку, является более важной характеристикой.

Главным фактором падения производительности является засорение компьютера различными данными. Это в основном игры и программы, а музыка, видео, картинки не столь важны.

Также на производительности значительно сказываются антивирусы и  подобные сканирующие программы.

Для того чтобы можно было объективно оценить степень улучшения системы после ее модернизации или сравнить быстродействие разных компьютеров, необходима количественная оценка производительности

Существуют специализированные испытательные программы (benchmark), которые измеряют производительность компьютера при выполнении на ней определенного набора задач.

В зависимости от ситуации они могут быть использованы в разных целях.

**1. Сравнение систем.** Испытательные программы часто используются *для сравнения компьютеров конкурирующих фирм* или выявления преимуществ новых моделей перед старыми.

**2. Оценка степени улучшения системы после ее модернизации**. Испытательные программы часто используют для оценки степени улучшения характеристик системы после ее модернизации.

**3. Диагностика неисправностей**. Иногда испытательные программы оказываются полезными для диагностики системы. Проверяя и перенастраивая ключевые компоненты компьютера и оценивая результаты произведенных манипуляций с помощью испытательных программ, *специалист сможет быстрее определить и устранить причину замедления работы системы, чем просто наблюдая за ее работой*.

**4. Проверка достоверности результатов работы испытательных программ**. Одной из самых серьезных проблем, связанных с испытательными программами, является *достоверность получаемых с их помощью результатов.*

Термин «узкое место» используется при рассмотрении вопросов производительности ПО и оборудования, относится к ограничивающему производительность состоянию, вызванному каким-либо компонентом или набором компонентов.

Причиной аппаратных узких мест могут становиться такие компоненты оборудования, как ЦП, память и  подсистема ввода-вывода;

причиной программных узких мест могут становиться приложения SQL Server и операторы SQL.

**Центральный процессор** Одной из наиболее распространенных проблем производительности является недостаток мощности.

Процессорная мощность системы определяется количеством, типом и  скоростью ЦП в  этой системе. Если системе недостает мощности ЦП, то она не может достаточно быстро обрабатывать транзакции, как это требуется пользователям.

**Память** Количество памяти, доступной SQL Server, является одним из наиболее критичных факторов для производитель-ности SQL Server. Важным фактором является также соотношение между памятью и  мощностью подсистемы ввода-вывода.

**Подсистема ввода-вывода**. Узкие места, возникающие в подсистеме ввода-вывода, — это наиболее распространенные проблемы оборудования для систем баз данных.

**Неисправные компоненты** Время от времени в системе могут возникать проблемы из-за неисправных компонентов. Если компонент не отказал полностью, но постепенно ухудшает свою работу, то такую проблему бывает трудно обнаружить. Эти проблемы принимают различные формы и  сложны для разрешения.

**Приложения.**

Еще одним компонентом системы, который обычно вызывает проблемы производительности, являются приложения SQL Server.

**Журнал событий** —в  Microsoft Windows стандартный способ для приложений и  ОС записи и  централизованного хранения информации о  важных программных и аппаратных событиях.

Служба журналов событий сохраняет события от различных источников в  едином журнале событий, программа просмотра событий позволяет пользователю наблюдать за журналом событий, программный интерфейс (API) дает возможность приложениям записывать в  журнал информацию и просматривать существующие записи.

Администраторы могут просматривать и очищать журнал, разделение прав невозможно. Администратор может использовать утилиту Winzapper для удаления записей о событиях. Если учётная запись администратора взломана, история событий в журнале становится недостоверной. Решение проблемы — создание удалённого сервера журнала с доступом через консоль.

1. **Настройка управления питанием.**
2. **Оптимизация использования процессора**.

Схема управления питанием представляет собой набор аппаратных и системных параметров, которые управляют тем, как компьютер использует энергию.

Некоторые планы питания предназначены для обеспечения высокой производительности с ущербом для автономности, в то время как другие настроены на обеспечение максимально возможного времени автономной работы, при этом производительность устройства ставится под угрозу.

**Оптимизация параметров персонального компьютера**

Можно решить проблему радикально: попросту поменять процессор и  (или) материнскую плату на более мощные современные модели, добавить модули оперативной памяти, установить новую видеокарту и  более «шустрые» жесткие диски.

1. Устранение недостатков конструкции компьютера, негативно сказывающихся на его работе.

2. Оптимизация параметров BIOS  — базовой системы ввода-вывода. Именно она управляет работой большинства ключевых компонентов системы, определяя их производительность.

3. Использование программ и утилит, позволяющих оптимизировать компоненты штатными средствами.

4. Регулярная проверка жесткого диска: дефрагментация, выявление и устранение ошибок.

5. Оптимизация настроек ОС. Основные области оптимизации — настройка виртуальной памяти, работающих служб и автоматически запускаемых программ.

6. Тщательная очистка системы от вирусов, троянов и  вредоносного ПО.

7. Оптимизация количества и  режима работы установленных в системе программ.

8. Выработка оптимальных приемов работы на персональном компьютере ограничен-ными системными ресурсами.

**Все манипуляции по улучшению качества работы ЦП можно поделить на две группы:**

**Оптимизация**. Основной акцент делается на грамотное распределение уже доступных ресурсов ядер и системы, дабы добиться максимальной производительности. В ходе оптимизации трудно нанести серьёзный вред ЦП, но и прирост производительности, как правило, не очень высокий.

**Разгон.** Манипуляции непосредственно с самим процессором через специальное ПО или BIOS для повышения его тактовой частоты. Прирост производительности в этом случае получается весьма ощутимым, но и возрастает риск повредить процессор и другие компоненты компьютера в ходе неудачного разгона.

**Оптимизация работы ОС.** Это самый безопасный метод увеличения производительности ЦП путём очистки автозагрузки от ненужных приложений и дефрагментации дисков.

**Автозагрузка** – это автоматическое включение той или иной программы/процесса при загрузке операционной системы. Когда в этом разделе скапливается слишком много процессов и программ, то при включении ОС и дальнейшей работе в ней, на центральный процессор может быть оказана слишком высокая нагрузка, что нарушит производительность.

**Очистка Автозагрузки**

В автозагрузку приложения можно добавлять как самостоятельно, так и приложения/процессы могут добавляться сами. Чтобы второго случая не было, рекомендуется внимательно читать все пункты, которые отмечены галочкой во время установки того или иного софта.

**Проведение дефрагментации**

Дефрагментация диска увеличивает не только скорость работы программ на этом диске, но также немного оптимизирует работу процессора.

В современных компьютерных системах эти два аспекта играют важную роль в эффективности работы и экономии энергии. В этой лекции мы рассмотрим основные концепции и методы настройки управления питанием, а также подходы к оптимизации использования процессора для повышения производительности.

Часть 1: Настройка управления питанием

1. Значение управления питанием:

а. Экономия энергии и продолжительность работы устройства без подзарядки.

б. Снижение выделения тепла и уменьшение шума от вентиляторов.

в. Улучшение производительности путем управления энергопотреблением.

2. Основные составляющие управления питанием:

а. Управление питанием процессора:

- Технология DynamIQ Voltage and Frequency Scaling (DVFS) для динамического изменения частоты и напряжения ядра процессора.

- Определение энергопрофилей для различных режимов работы процессора.

б. Управление питанием графического ускорителя (GPU):

- Управление частотой графического процессора и памяти в зависимости от нагрузки.

- Включение и отключение эффектов для снижения энергопотребления.

в. Управление питанием других компонентов:

- Оптимальное использование энергии для жесткого диска, оперативной памяти, периферийных устройств и других компонентов системы.

3. Стратегии настройки управления питанием:

а. Динамическое управление:

- Анализ нагрузки и настройка питания в режиме реального времени.

- Использование алгоритмов прогнозирования и адаптации для оптимального распределения ресурсов.

б. Статическое управление:

- Задание параметров управления на этапе конфигурирования системы.

- Определение энергопрофилей для различных сценариев использования.

Часть 2: Оптимизация использования процессора

1. Понимание нагрузки на процессор:

а. Измерение загрузки процессора и определение пропускной способности системы.

б. Анализ производительности приложений и определение причин узкого места.

в. Определение приоритетов и оптимизация использования процессорного времени.

2. Техники оптимизации использования процессора:

а. Многопоточность и параллелизм:

- Использование многопоточных приложений для распределения вычислительной нагрузки между несколькими ядрами процессора.

- Оптимизация параллельных алгоритмов для ускорения работы.

б. Планирование задач:

- Использование планировщика операционной системы для оптимального распределения процессорного времени между различными приложениями.

в. Использование оптимизированных библиотек и инструментов:

- Использование специальных библиотек и инструментов для оптимизации работы процессора в конкретных типах приложений, например, графических или математических вычислений.

Заключение:

Настройка управления питанием и оптимизация использования процессора являются ключевыми аспектами в современных компьютерных системах. Правильная настройка позволяет достичь оптимального баланса между производительностью и энергопотреблением, что в свою очередь сокращает затраты на энергию и продлевает время автономной работы. Оптимизация использования процессора создает условия для более эффективной работы приложений и повышения производительности системы в целом.

1. **Оптимизация использования памяти. Оптимизация использования жесткого диска. Оптимизация использования сети.**

**Способ 1: Очистка кэша ОЗУ**

Как известно, в оперативную память загружаются данные приложений, что позволяет ускорить их запуск и выполнение каких-либо операций. Информация, которая считается устаревшей, выгружается или перезаписывается автоматически, однако это происходит не всегда, что напрямую влияет на быстродействие и загруженность RAM.

Время от времени необходимо очищать кэш самостоятельно и проверять, как это скажется на работе Windows 10.

**Способ 2: Обновление драйверов**

Следующая стандартная рекомендация заключается в ручной проверке обновлений драйверов для всех комплектующих, установленных в ПК. Это требуется для того, чтобы исключить вероятность появления конфликтов из-за недостающих файлов или несовместимости. Вы можете сами с помощью стандартных или сторонних средств запустить эту проверку и установить все найденные драйверы

**Способ 3: Установка системных обновлений**

Исправления и нововведения от Майкрософт тоже оказывают прямое влияние на быстродействие и загрузку оперативной памяти разными службами и процессами. Лучше всегда поддерживать ПК в актуальном состоянии, чтобы избегать различных сбоев и конфликтов.

**Способ 4: Проверка системы на наличие вирусов**

**Способ 5: Отключение программ автозагрузки**

**Способ 6: Отключение открытия приложений после перезапуска**

По умолчанию в Windows 10 активирована функция, автоматически запускающая незакрытые программы при перезагрузке или обновлении системы. Далеко не всем эта опция требуется, поэтому ее можно отключить, чтобы разгрузить оперативную память, ведь теперь кэш сохраняться не будет.

**Способ 7: Отключение фоновых приложений -** компьютерный процесс, который выполняется за кулисами и без вмешательства пользователя

**1.Оптимизация использования жесткого диска**

**Поддержание порядка на диске**

Для того, чтобы на диске всегда был порядок, достаточно свободного места и не снижалась его производительность, нужно выполнить несколько простых настроек и придерживаться определенных правил хранения файлов.

**Отключение лишних обновлений**

**Систематизация файлов**

Чтобы на диске всегда был порядок, вы знали где и какие файлы у вас находятся и сколько они занимают места, приучите себя к их правильному размещению.

Не храните файлы и папки на рабочем столе, он предназначен в основном для ярлыков

**Ускорение работы диска**

**Обновление драйвера контроллера дисков**

От драйвера контроллера дисков зависит не только скорость его работы, но и стабильность всей системы.

**Включение функции TRIM на SSD**

Все современные диски SSD поддерживают функцию TRIM, которая предназначена для оптимизации их скорости.

**Отключение лишних служб**

Во всех версиях Windows есть множество не совсем нужных системных служб. Некоторые из них замедляют работу диска и всей системы, а также приводят к повышенному износу HDD и SSD дисков.

**Отключение автозагрузки программ**

**Дефрагментация диска**

В процессе работы диска происходит его фрагментация, т.е. файлы разбиваются на множество мелких фрагментов. При обращении к тому или иному файлу, диску приходится собирать все его фрагменты в единое целое, что значительно снижает производительность самого диска и всей системы.

**2.Оптимизация использования сети.**

Скорость интернета на устройствах с Windows 10 может быть низкой из-за:

* механических повреждений кабеля при проводном подключении;
* устаревших драйверов Wi-Fi адаптера;
* включённых ограничений скорости;
* неправильных настроек файлообменника или браузера и т. д.

**Оптимизация скорости файлообмена на устройстве**

**Отключение ограничения скорости QoS**

QoS — планировщик пакетов, обеспечивающий нормальную работу интернет-соединения, когда сеть перегружена.

**Изменение лимита пропускной способности сети**

Причиной низкой скорости сетевого соединения может быть активное использование исходящего канала каким-либо приложением, работающим в фоновом режиме

**Программы и утилиты для увеличения скорости интернета**

**Анализ скорости выполнения** после разбиения чистого кода даст вам объективную оценку того, в какую сторону направить свои усилия по оптимизации.

1. **Инструменты повышения производительности программного обеспечения.**

На скорость выполнения вашей программы влияет 2 непосредственных фактора:

• Скорость исполнения кода

• Объём выделяемой памяти

Как вы понимаете, минимизировать одновременно влияние обоих этих факторов невозможно: заигрывания с памятью неизбежно приводят к раздуванию кода, а «малобуквенный» текст к повышенному потреблению ресурсов исполняемой машиной.

Поэтому главным правилом оптимизации является исключение излишеств в вашей программе.

Это означает никакого лишнего кода и строго ограниченное использование памяти.

Если задача оптимизации встала перед вами уже после написания кода, то наиболее разумным решением будет предварительная очистка и разбиение на части с последующим изучением времени выполнения отдельных блоков.

Очистка предполагает удаление неиспользуемых участков, переменных, избыточных заходов циклов.

Анализ скорости выполнения после разбиения чистого кода даст вам объективную оценку того, в какую сторону направить свои усилия по оптимизации.

Логично, что если одна операция выполняется 40% времени, а другая – 2%, то и разработчик должен куда больше усилий приложить к уменьшению производственных потерь в первом случае.

Так как мы предполагаем, что код уже чистый, то необходимо в первую очередь рассмотреть выполнение следующих правил:

• Минимизировано количество используемых переменных.

Например, ваша программа изобилует различными циклами. По правилам красивого кода, вы можете создать для каждого цикла свой уникальный счётчик, но с точки зрения оптимизации – это трата драгоценных ресурсов. Если в одной части программы переменная “отработала”, то её вполне можно применить в другой.

• Правильно выбраны типы данных.

Как известно, каждый тип данных имеет свой используемый диапазон, то есть собственно тот размер памяти, который резервируется под его использование.

Например, создавая код в Java и имея переменную, способную принимать только два значения (например, «on» и «off»), лучше использовать boolean с созданием последующего соответствия, но никак не char.

• Минимизировано количество присваиваний.

Опять-таки, руководствуясь принципами красоты кода, вы можете разбить длинное арифметическое выражение на несколько более мелких. Это чревато появлением избыточных переменных и лишними операциями присваивания, что позитивно на быстродействии точно не скажется.

• Однотипные циклы объединены.

Допустим, у вас есть несколько массивов одинаковой размерности, которые надо заполнить в цикле. Вы можете создать несколько циклов и повысить читабельность или запихнуть все операции в общий цикл и повысить быстродействие. Решать вам.

• Использованная память немедленно очищается. Безусловно, не стоит удалять каждую переменную сразу после окончания её использования, но когда речь идет о работе с существенным объемом памяти (например, с большими массивами), контролировать потребление ресурсов просто необходимо.

Когда оптимизация не имеет смысла

• Безусловно, подобные приемы не являются универсальными, а в случае с рядом языков и компиляторов (как в случае Dart в JavaScript с помощью dart2js), вы и вовсе получите настолько оптимизированный код, что собственными руками вряд ли

создадите что-то лучше.

• Однако, в любом случае, опыт применения описанных выше приемов поможет новичку перейти из разряда хороших программистов в разряд эффективных, ведь так вы сэкономите не только ресурсы машины, но и свое личное время.

1. **Средства диагностики оборудования. Разрешение проблем аппаратного сбоя.**

**Конфликты отличаются тремя характерными признаками:**

* в систему было установлено новое устройство или программа;
* проблема возникла после установки нового устройства или программы;
* до установки нового устройства или программы система работала нормально.

**Выявление и устранение конфликтов оборудования**

компьютер зависает в процессе выполнения POST или инициализации ОС;

система зависает при выполнении прикладной программы;

система зависает при использовании конкретного устройства(например, сканер);

компьютер зависает произвольным образом, без предупреждения и независимо от исполняемой программы;

компьютер может и не зависать, но новое устройство не работает. При этом ранее установленные в систему устройства могут продолжать работать нормально;

компьютер может не зависать, но устройства или программы, ранее работавшие нормально, теперь не функционируют. Вновь установленное устройство (и соответствующее ПО) может работать, а может и не работать.

*Основное правило разрешения конфликта* можно кратко сформулировать следующим образом: **то, что было подключено последним, отключается первым**.

**Программные конфликты**. В компьютере конфликтовать могут программы двух типов: резидентные программы и драйверы устройств.

**Резидентные программы**

Загружаются в память обычно в период инициализации компьютера и ожидают некоторого системного события (например, нажатия на клавиатуре комбинации «горячих клавиш»).

**Драйверы устройств**

Их загрузка производится при обработке командных строк в  файле CONFIG.SYS в  период инициализации компьютера (или же они загружаются вместе с  ОС Windows).

**Аппаратные конфликты**.

нужно перенастроить параметры одного из конфликтующих устройств или программы.

Средства диагностики подразделяют на программные и сервисную диагностическую аппаратуру.

Средства диагностики подразделяют на программные и сервисную диагностическую аппаратуру.

**Диагностические программы подразделяются по нескольким основаниям.**

* Диагностическое ПО включает в  себя стандартное ПО, встроенное в состав ОС, и сторонние утилиты.
* По широте использования выделяют комплексные и специализированные утилиты.

**Комплексные утилиты** предназначены для получения подробной информации об установленных компонентах, позволяют провести тестирование их на быстродействие, работоспособность, производительность, наличие ошибок и конфликтов и т. д.

**Специализированные утилиты**— более узкого назначения. Позволяют получить более детальную информацию о  конкретном устройстве, произвести его настройку, включить/отключить различные режимы работы, которые не доступны с  помощью стандартных средств.

**По возможностям выделяют программы:**

* тестирующие быстродействие и производительность системы;
* проверяющие на наличие ошибок и конфликтов;
* информационные, выдающие информацию об устройстве;
* позволяющие управлять работой устройства, изменять параметры

**К аппаратным средствам диагностики неисправностей СВТ относятся**:

универсальные измерительные приборы (тестеры, осциллографы, мультиметры и др.);

специальная сервисная аппаратура для СВТ (диагностическая плата POST Card, диагностические заглушки для проверки портов Loop Back).

1. **Аппаратно-программные платформы серверов и рабочих станций. Установка серверной части.**

В качестве серверов в средних и крупных информационных системах используют специализированные многопользовательские мощные компьютеры  — серверы.

В настоящее время на рынке представлено несколько основных платформ компьютеров, каждая из которых отличается как по назначению, так и по типу использованного «железа» и программ.

**Основными достоинствами персональных компьютеров являются:**

* небольшие физические габаритные размеры;
* мощные вычислительные возможности;
* простота эксплуатации пользователем-непрофессионалом;
* невысокая стоимость;
* отсутствие серьезных требований и  ограничений по условиям эксплуатации.

Платформа IBM включает разнообразные компьютеры, от домашних до серверов. Платформа . с середины 1970-х гг. Apple представлена множеством моделей персональных компьютеров, включая Apple I и iMac. На компьютерах Apple впервые появились такие новшества, как графический интерфейс, мышь, звуковая подсистема и компьютерное видео. Интерфейс Windows частично скопирован с ОС Apple.

**Кластерная структура сервера**. Кластер представляет собой многомашинный компьютерный комплекс, который с точки зрения пользователя:

является единой системой;

обеспечивает высокую надежность (отказоустойчивость);

имеет общую файловую структуру;

обладает свойством эффективной масштабируе-мости — производительности при добавлении ресурсов;

управляется (администрируется) как единая система.

Иногда *кластером называют комплекс* из двух компьютеров, один из которых делает полезную работу, а другой включен и находится в  «горячем» резерве.

**Серверное программное обеспечение** — в информационных технологиях — программный компонент вычислительной системы, выполняющий сервисные (обслуживающие) функции по запросу клиента, предоставляя ему доступ к определенным ресурсам или услугам.

**Универсальные серверы** — особый вид серверной программы,не предоставляющий никаких услуг самостоятельно. Вместо этого универсальные серверы предоставляют серверам услуг упрощенный интерфейс к  ресурсам межпроцессного взаимодействия и (или) унифицированный доступ клиентов к различным услугам.

**Сервер маршрутизации** не является сервером в классическом смысле, а является базовой функцией поддержки сети ОС.

Маршрутизацию своих пакетов к месту назначения выполняет любая система в сети, маршрутизацию же чужих пакетов (форвардинг) осуществляют только маршрутизаторы (также известные как роутеры или шлюзы)

**Файл-серверы** представляют собой серверы для обеспечения доступа к  файлам на диске сервера. Прежде всего это серверы передачи файлов по заказу, по протоколам FTP, TFTP, SFTP и HTTP.

**Серверы доступа к  данным** обслуживают базу данных и  отдают данные по запросам. Один из самых простых серверов подобного типа — LDAP /

**Службы обмена сообщениями** позволяют пользователю передавать и  получать сообщения (обычно  — текстовые). В  первую очередь это серверы электронной почты, работающие по протоколу SMTP. SMTP-сервер принимает сообщение и  доставляет его в  локальный почтовый ящик пользователя или на другой SMTP-сервер

**Серверы удаленного доступа** через соответствующую клиентскую программу обеспечивают пользователя консольным доступом к  удаленной системе. Для обеспечения доступа к  командной строке служат серверы telnet, RSH, SSH. Графический интерфейс для UNIX-систем  — X Window System, имеет встроенный сервер удаленного доступа, так как с такой возможностью разрабатывался изначально.

**Игровой сервер** служит для одновременной игры нескольких пользователей в  единой игровой ситуации. Некоторые игры имеют сервер в основной поставке и позволяют запускать его в невыделенном режиме (т. е. играть на машине, на которой запущен сервер).

**Серверные решения** — операционные системы и (или) пакеты программ, оптимизированные под выполнение компьютером функций сервера и  (или) содержащие в  своем составе комплект программ для реализации типичного набора сервисов.

1. **Виды серверного программного обеспечения. Особенности эксплуатации различных видов серверного программного обеспечения.**

**Серверное программное обеспечение** — в информационных технологиях — программный компонент вычислительной системы, выполняющий сервисные (обслуживающие) функции по запросу клиента, предоставляя ему доступ к определенным ресурсам или услугам.

**Серверное ПО** – это ПО, предоставляющее услуги или функции на компьютере, выступающим в качестве среды.

**Виды серверного ПО**:

**Файловый сервер** – предназначен для обеспечения доступа к файлам, хранящимся на серверных дисках организации.

**Сервер баз данных** – обеспечивает хранение, обработку и доступ к базам данных компании, осуществляемы с клиентских компьютеров.

**Сервер приложений** – осуществляет программную обработку данных, которые посылает ему пользователь, и выдает этому пользователю конечный результат.

**Веб-сервер** – отвечает за выдачу интернет страницы. Может одновременно обрабатывать большое количество запросов.

**Почтовый сервер** – предназначен для отправки, получения, хранения и распределения электронных писем.

**Брандмауэр** (файервол) – обеспечивает защиту внутренней сети и ее ресурсов от интернет-атак.

**Прокси-сервер** – служба, позволяющая выполнять клиентам косвенные запросы к другим сетевым службам.

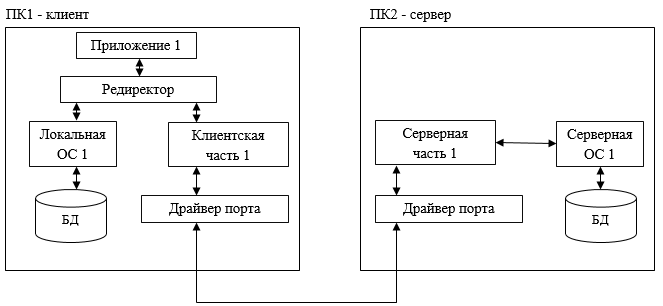
Прокси-сервер скрывает от внешних пользователей структуру сети, обеспечивает доступ к сети по одному IP-адресу. Позволяет вмести с DNS-сервером производить автоматическую раздачу и назначение IP-адреса.

**DNS-сервер** – служит для автоматического учета и выдачи уникальных IP-адресов всем узлам, которые к нему обращаются.

**Сервер удаленного доступа** – позволяет получать через Интернет доступ к локальной сети.

**Принт-сервер** – позволяет получить доступ к сетевому принтеру.

## Взаимодействие серверного и клиентского программного обеспечения



Если запрос передается к ресурсу данного ПК, то он переадресовывается к локальной ОС.

Если же это запрос к удаленному ресурсу, то он перенаправляется в клиентскую часть, где преобразуется из локальной формы в сетевой формат и передается портом.

Серверная часть ОС ПК2 принимает запросы, преобразуя их в локальную форму и передает для выполнения своей локальной ОС.

После того как результат получен сервер образуется к транспортной подсистеме и направляет ответ клиенту, выдавшему запрос.

Клиентская часть преобразует результат в соответствующий формат и адресует его тому приложению, которое выдало запрос.

**Порядок установки ПО:**

1. Убедиться, что конфигурация компьютера отвечает минимальным требованиям ПО.

2. Проверить наличие свободного места на жестком диске, по необходимости освободить.

3. Отключить антивирусные программы.

4. Остановить выполнение посторонних программ.

1. **Виды клиентского программного обеспечения. Установка, адаптация и сопровождение клиентского программного обеспечения.**

**Программное обеспечение** – это совокупность программных и документальных средств для создания и эксплуатации систем обработки данных средствами вычислительной техники.

**Клиентское ПО** – это ПО, осуществляющее доступ к ресурсам, которые предоставляет сервер.

Для работы с сетью на клиентских рабочих станциях должно быть установлено клиентское ПО.

**Компоненты клиенского ПО:**

* + Редиректоры
  + Распределители
  + Имена UNC

**Редиректор** – сетевое программное обеспечение, которое принимает запросы ввода-вывода для удаленных файлов, именованных каналов или почтовых слотов и затем переназначает их сетевым сервисам другого компьютера.

**Распределитель** представляет собой часть ПО, управляющую присвоением букв накопителя как локальным, так и удаленным сетевым ресурсам или разделяемым дисководам, что помогает во взаимодействии с сетевыми ресурсами.

**UNC** представляют собой стандартный способ именования сетевых ресурсов. Эти имена имеют форму \\Имя\_сервера\имя\_ресурса

**Виды клиентского программного обеспечения**

**Выделенный серв**ер не принято использовать в качестве компьютера для выполнения текущих задач, не связанных с  его основным назначением, так как это может уменьшить производительность его работы как сервера.

**В одноранговых сетях** все компьютеры равны в правах доступа к ресурсам друг друга.

В таких сетях на всех компьютерах устанавливается одна и та же ОС, которая предоставляет всем компьютерам в сети потенциально равные возможности.

**Установка, адаптация и сопровождение клиентского программного обеспечения**

**Подготовка компьютера перед установкой**

* Убедится, что конфигурация компьютера отвечает минимальным требованиям ПО.
* Проверить наличия свободного места на жёстком диске, если его не хватает, освободить.
* Отключить антивирусные программы.
* Остановить выполнение посторонних программ.

**Поддержка программного обеспечения**

**Пакеты исправлений и заплатки**

Время от времени производители программного обеспечения выпускают исправления к своим программным продуктам. Протестировав исправление и приняв решение о его развертывании, администратор копирует необходимые файлы в точку распространения ПО, производя замену старых файлов.

Производители программного обеспечения распространяют обновления либо в виде нового пакета установщика, либо в качестве исправления установщика.

Между пакетами обновления и исправлениями нет больших отличий. Как правило, в состав пакета обновления входят несколько уже протестированных исправлений. Пакеты обновления распространяются реже, чем исправления, но чаще, чем полные обновления программного продукта

**Обновления**

Обновление подразумевает замену значительного числа файлов. Поскольку производятся большие изменения в программном продукте, обновление получает другой номер версии.

Производитель предоставляет новую версию в пакете установщика ПО, с заложенной в него информацией о том, какие более ранние версии он может обновлять.

В пакете также должен содержаться сценарий процесса обновления, в котором описано, какие файлы должны быть заменены, удалены и добавлены.

1. **Многоуровневая модель качества программного обеспечения.**

Теоретические модели – основанные на гипотезе отношений между переменными качества.

Модели управления данными – основаны на статическом анализе.

Комбинированная модель – в которой интуиция исследователя используется для определения нужного вида модели, а анализ данных применяется для определения констант модели качества.

**Модель МакКола**

Первая модель качества была предложена в 1977 г. Дж.МакКолом.

Эта модель предназначена для определения полной характеристики качества ПО через его различные характеристики. Имеет три главных направления :

**Использование** ( корректность, надежность, эффективность, целостность, практичность)

**Модификация** (тестируемость, гибкость, сопровождаемость).

**Переносимость** (мобильность, возможность многократного использования, функциональная совместимость) — факторы качества, важные для переносимости программного продукта на другие аппаратные и программные платформы

**Модель Боэма. 1978**

В этой модели:

практичность описывает то, как легко, надежно и эффективно может быть использован пакет программ;

сопровождаемость характеризует, насколько легко изменить и повторно протестировать пакет программ;

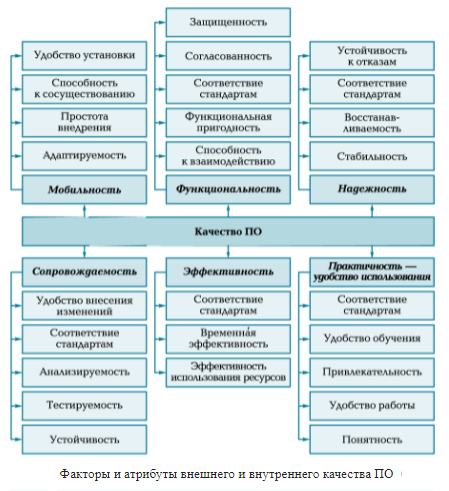
мобильность описывает, как пакет программ может использоваться даже при изменении программных и  аппаратных средств.

**Модель качества ISO 9126.**

Модель качества ISO 9126-1 различает понятия

1. внутреннего качества, связанного с характеристиками ПО самого по себе, без учета его поведения;
2. внешнего качества, характеризующего ПО с  точки зрения его поведения;
3. качества ПО при использовании в различных контекстах — того качества, которое ощущается пользователями при конкретных сценариях работы ПО.

***Метрика качества программ****— система измерений качества программ. Эти измерения могут проводиться на уровне критериев качества программ или на уровне отдельных характеристик качества.*



1. **Объекты уязвимости.**

Объект уязвимости - это элемент, ресурс, система, или процесс, который имеет потенциальные уязвимости, делая его подверженным различным видам угроз и атак.

*Дефект безопасности* - недостаток создания ПО, потенциально влияющий на степень безопасности информации.

**Классификации угроз и атак**

**Классификации выделяют класс угроз**, связанный с возможностью реализации нарушителем программных уязвимостей. Данные классификации являются основой для построения моделей угроз безопасности информации.

**Классификации вредоносных программ**

Разработчики средств антивирусной защиты придерживаются классификаций «вредоносного» ПО по модели распространения, по способу активации, по действию и другим параметрам, что позволяет разрабатывать эффективные тесты и базы сигнатур антивирусов. Данные классификации полезны при описании подкласса уязвимостей эксплуатационного типа.

**Классификации дефектов**

Большая часть дефектов возникает в процессе создания ПО. Это могут быть ошибки проектирования, ошибки кодирования программистов, ошибки, допущенные при сборке дистрибутива и интеграции различных версий компонентов ПО.

**Систематика дефектов и уязвимостей**

Требования к перспективным таксономиям дефектов и уязвимостей в области безопасности ПО:

в классификации или свойствах отдельного таксона должна содержаться информация об этапе жизненного цикла, на котором возникает дефект ПО и его области (общая архитектура, код, внутренняя конфигурация, внешнее окружение);

**Систематика дефектов и уязвимостей**

Классификация уязвимостей включает два типа и восемь классов.

Типы представляют собой:

* уязвимости, вызванные дефектами проектирования и программирования;
* уязвимости, вызванные дефектами конфигурирования и управления

**Восемь классов** соответствуют наиболее применимым с точки зрения практики анализа кода международ-ным таксономиям, а именно включают уязвимости, связанные со следующим:

● обработкой и представлением данных;

● внутренней структурой и зависимостями компонентов;

● обработкой событий и состояний;

● внутренними механизмами и ресурсами;

● преднамеренным внедрением;

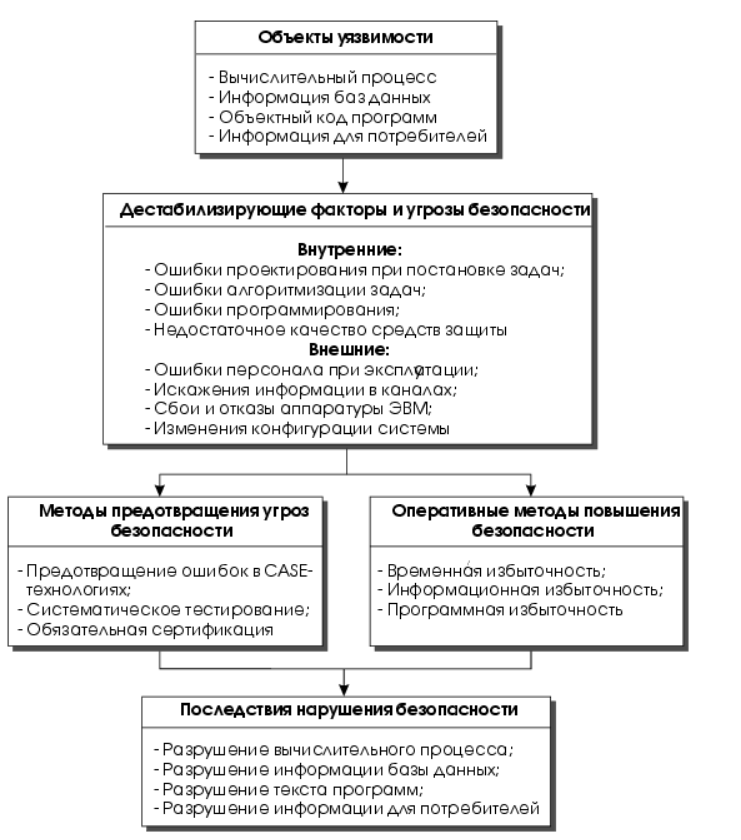
● качеством проектирования и документированием;

● конфигурациями;

● окружением.

1. **Дестабилизирующие факторы и угрозы надежности.**

**Непредумышленные дестабилизирующие факторы, влияющие на безопасность функционирования программных средств и баз данных (ПС и БД)**



**В качестве объектов уязвимости рассматриваются**:

динамический вычислительный процесс обработки данных, автоматизированной подготовки решений и выработки управляющих воздействий;

информация, накопленная в базах данных;

объектный код программ, исполняемых вычислительными средствами в процессе функционирования ИС;

информация, выдаваемая потребителям и на исполнительные механизмы.

**Внутренние источники** угроз безопасности функционирования сложных ИС

**системные ошибки** при постановке целей и задач проектирования ИС, формулировке требований к функциям и характеристикам решения задач, определении условий и параметров внешней среды, в которой предстоит применять ИС;

**алгоритмическая ошибки** проектирования при непосредственной алгоритмизации функций программных средств и баз данных, при определении структуры и взаимодействия компонент комплексов программ, а также при использовании информации баз данных;

ошибки программирования в текстах программ и описаниях данных;

**недостаточная эффективность** используемых методов и средств оперативной защиты программ и данных и обеспечения безопасности функционирования ИС в условиях случайных негативных воздействий.

**Внешние дестабилизирующие факторы ошибки оперативного и обслуживающего персонала в процессе эксплуатации ИС:**

искажения в каналах телекоммуникации информации, поступающей от внешних источников и передаваемой потребителям, а также недопустимые изменения характеристик потоков информации;

сбои и отказы аппаратуры;

изменения состава и конфигурации ИС за пределы, проверенные при испытаниях или сертификации.

**Полное устранение перечисленных угроз принципиально невозможно.**

* Задача состоит в выявлении факторов, от которых они зависят;
* В создании методов и средств уменьшения их влияния на безопасность ИС;
* В эффективном распределении ресурсов для обеспечения защиты, равнопрочной по отношению ко всем негативным воздействиям.

**Внутренние дестабилизирующие факторы**

**Различия между ожидаемыми и полученными результатами** функционирования программ и баз данных могут быть следствием ошибок не только в созданных программах и данных, но и системных ошибок в первичных требованиях спецификаций, явившихся основой при создании ИС.

**В результате важной особенностью** процесса выявления ошибок в программах и данных сложных, критических ИС является отсутствие полностью определенного эталона, которому должны соответствовать текст и результаты функционирования разработанной программы.

**При отладке и тестировании** обычно сначала обнаруживаются вторичные ошибки, то есть последствия и результаты проявления некоторых дефектов, которые следует квалифицировать как первичные ошибки.

**Проявления ошибок** в разной степени влияют на работоспособность программ и их нельзя целиком квалифицировать как отказы. В худшем случае вторичная ошибка проявляется как полный отказ — потеря работоспособности ПС и БД на длительное время, угрожающая безопасности.

1. **Методы предотвращения угроз надежности.**

**Методы снижения угроз безопасности ИС, вызванных дефектами программных средств и баз данных:**

Уровень и влияние внутренних дестабилизирующих факторов, а также некоторых внешних угроз на безопасность применения ИС определяется в наибольшей степени качеством технологий проектирования, разработки, сопровождения и документирования ИС и их основных компонент — программных средств и баз данных.

В современных автоматизированных технологиях создания и развития ПС и БД с позиции обеспечения их потенциальной технологической безопасности можно выделить методы и средства, позволяющие:

* предотвращать дефекты проектирования за счет эффективных технологий обеспечения всего жизненного цикла комплексов программ и данных;
* обнаруживать и устранять ошибки проектирования путем систематического тестирования на всех этапах жизненного цикла ПС и БД;
* удостоверять достигнутые качество и безопасность применения ПС и БД в процессе их сертификации перед передачей в эксплуатацию.

При разработке сложных информационных систем необходимо уделить особое внимание правильному системному и информационному проекту, чтобы обеспечить высокое качество и безопасность системы.

Благодаря качественной проработке и детальному описанию проекта, становится проще проводить отладку, тестирование и сопровождение прикладной информационной системы.

Совместное применение современных CASE-технологий и языков четвертого поколения способно снизить трудоемкость разработки сложных программных средств до 10 раз и сократить длительность их проектирования с 2-3 лет до нескольких месяцев.

**Предотвращение ошибок в CASE-технологиях**

Современные методы и технологии создания программного обеспечения основаны на использовании ранее разработанных решений на разных аппаратных и операционных системах. Сейчас лишь 10–15 % программ создаются заново, остальные берутся из существующих источников и собираются из готовых компонентов. Это повышает производительность труда разработчиков, сокращает сроки создания и улучшает качество проектов. Стандарты открытых систем помогают предотвратить ошибки в сложных распределённых системах, но массовый перенос программ может привести к распространению дефектов в переносимых компонентах.

Однако переносимые компоненты, как правило, тщательнее тестируются и испытываются и тем *самым имеют более высокое качество*, чем те, которые созданы без ориентации на переносимость.

Стандартизация и глубокий контроль интерфейсов и протоколов взаимодействия компонент ИС позволяют создавать сложные, распределенные ИС высокой надежности и безопасности.

Строгое соблюдение и контроль соответствия стандартам открытых систем (зачастую автоматически осуществляемые CASE-средствами) является высокоэффективным методом предотвращения ряда классов ошибок и повышения технологической безопасности ИС.

**Тестирование является основным методом измерения качества и определения реальной безопасности** применения программ и информации баз данных на любых этапах разработки.

Непосредственной целью тестирования является обнаружение, локализация и устранение ошибок в программах и данных.

Чтобы убедиться в качестве и надёжности работы сложных и важных информационных систем, нужно обязательно сертифицировать программное обеспечение и базы данных, которые в них используются. Но сертификация подтверждает безопасность только в рамках определённых стандартов и нормативных документов.

Во время реальной работы могут возникнуть ситуации, когда характеристики внешней среды выходят за рамки сертификата, и проблемы, которые не были обнаружены при проверке. Такие случаи могут угрожать безопасности системы.

1. **Оперативные методы повышения надежности: временная, информационная, программная избыточность.**

**Временная избыточность** — это использование части мощности компьютера для проверки правильности выполнения программ и восстановления работы в случае сбоев. При создании информационной системы закладывается определённый запас мощности, который потом используется для контроля и повышения надёжности и безопасности её работы. Уровень временной избыточности зависит от требований к безопасности критически важных систем управления и обработки информации и может составлять от 5–10 % производительности до трёх-четырёхкратного дублирования в мажоритарных вычислительных комплексах.

**Информационная избыточность** — это когда данные дублируются, чтобы обеспечить их сохранность. Она нужна для защиты самой важной информации, без которой система не сможет нормально работать и которая долго восстанавливается. Такая информация касается внешних процессов, и если она пропадёт, то система не сможет управлять этими процессами и обрабатывать информацию о них, что повлияет на безопасность системы.

**Программная избыточность** используется для контроля и обеспечения достоверности наиболее важных решений по управлению и обработке информации. Она заключается в сопоставлении результатов обработки одинаковых исходных данных разными программами и исключении искажения результатов, обусловленных различными аномалиями. Программная избыточность необходима также для реализации средств автоматического контроля и восстановления данных с использованием информационной избыточности и для функционирования всех средств защиты, использующих временную избыточность.

К способам обеспечения и повышения надежности ПО относятся:

* усовершенствование технологии программирования (например, формальное описание этапов программирования с  помощью языка UML);
* выбор алгоритмов, не чувствительных к различного рода нарушениям вычислительного процесса (использование алгоритмической избыточности);
* резервирование программ  — N-версионное программирование;
* верификация и  валидация программ с  последующей коррекцией

**Валидация** – гарантированная уверенность производителя в том, что он создал продукт по всем необходимым стандартам.

**Верификация** – помогает увериться в том, что изделие соответствует всем изначально заданным требованиям к нему.

**Оперативные методы повышения надежности**

Средства программного контроля обычно не могут обнаруживать возникновение искажения вычислительного процесса или данных (первичную ошибку) и фиксируют, как правило, только последствия первичного искажения (вторичную ошибку).

Результаты первичного искажения в ряде случаев могут развиваться во времени и принимать катастрофический характер отказа при увеличении запаздывания в обнаружении последствий первичной ошибки.

повторение функциональной группы программ при тех же исходных данных или восстановление данных в процессе последующей обработки;

кратковременное прекращение решения задач данной группы прикладных программ до обновления исходных данных;

перестройка режима работы или структуры ИС для снижения влияния перегрузки или в связи с потерей информации о ходе процесса обработки данных и управления;

переход на резервную ЭВМ с накопленной информацией о ходе процесса управления или восстановление информации за счет ее дублирования;

восстановление процесса управления или обработки информации с режима начального пуска всей ИС с оперативным вмешательством обслуживающего персонала.

1. **Первичные ошибки, вторичные ошибки и их проявления.**

Ошибка - непра­вильность, погрешность или неумышленное, невольное иска­жение объекта или процесса.

**Источники ошибок:**

*Плохая спецификация.* (Плохо представили назначение программы → невозможно предусмотреть обработку всех ошибок)

*Неполные программы.* В ходе разработки неизбежно возникают варианты, которые мы не предусмотрели.

*Непредусмотренные аргументы.* Если функция принимает аргумент, который не был предусмотрен, то возникнет проблема.

*Непредусмотренные входные данные.*

*Неожиданное состояние.* Большинство программ хранит большое количество данных. Что произойдет, если эти данные окажутся неполными или неправильными?

*Логические ошибки.* Эти ошибки приводят к тому, что программа просто делает не то, что от нее ожидается.

**По уровням ошибки подразделяют на первичные, вторичные, небольшие, умеренные и  критические.**

Критические ошибки с высоким влиянием, останавливают выпуск версии программного продукта.

При отладке и тестировании ПО обычно сначала обнаруживаются вторичные ошибки, т. е. последствия внутренних его дефектов, которые следует квалифицировать как первичные причины обнаруженных аномалий результатов.

Локализация и корректировка первичных ошибок должны приводить к устранению ошибок, первоначально обнаруживаемых в результатах функционирования программ.



**Вторичные ошибки по**-разному влияют на общую эффективность СПК(сложных программных комплексов) и являются определяющими при определении эффективности его функционирования.

Не каждая первичная ошибка вносит заметный вклад в выходные результаты, вследствие чего ряд из них может оставаться не обнаруженным.

Наибольшее число наиболее сложных для обнаружения и  устранения первичных ошибок вносится на этапах системного анализа и разработки модификаций программ.

На последующих этапах разработки изменений ПО ошибки вносятся и устраняются в программах в процессе их корректировки по результатам тестирования.

***Системные ошибки*** в большом программном обес­печении определяются, прежде всего неполной информацией о реальных процессах, происходящих в источниках и потребителях информации.

***Ошибки в выборе алгоритма.*** В настоящее время накоплен значительный фонд алгоритмов для решения типовых задач.

***Технологические ошибки***— это ошибки документации и фик­сирования программ в памяти ЭВМ.

Они составляют 5—10 % от общего числа ошибок, обнаруживаемых при отладке.

Боль­шинство технологических ошибок выявляются автоматически формализованными методами (например, транслятором).

***Программные ошибки.*** Языки программирования - это ис­кусственные языки, созданные человеком для описания алго­ритмов.

***Синтаксис****-*это набор правил построения из символов алфавита специальных конструкций, с помощью которых можно составлять различные алгоритмы (программы).

***Семантика языка****—*это система правил истолкования пост­роений конструкций.

**Небольшими ошибками** называют такие, на которые средний пользователь не обратит внимания при применении программного продукта вследствие отсутствия их проявления и  последствия которых обычно так и  не обнаруживаются.

**Умеренные ошибки** влияют на конечного пользователя, но имеются слабые последствия или обходные пути, позволяющие сохранить достаточную функциональность программного продукта.

**Чтобы избежать ошибок, необходимо проводить тестирование. Существует несколько видов проведения тестов. Используются дополнительные инструменты.**

*Отладчик*. Он позволяет в автоматическом режиме контролировать работоспособность программы и сразу предупреждает о проблемах.

*unit-тесты*. Специалист описывает ситуации для каждого компонента, и указывает, какой результат ожидается. Потом запускается проверка, и если результат не совпадает с ожидаемым, появляется предупреждение. Далее программисты находят и устраняют проблему.

*альфа-тестирования* команда разработчиков имитирует деятельность пользователей и пытается найти проблемные участки.

*бета-тестирования* программа предоставляется пользователям для реального использования. Все эти тесты помогают запустить продукт, минимизируя риски ошибок.

1. **Математические модели описания статистических характеристик ошибок в программах.**

*Математические модели позволяют оценивать характеристики ошибок в программах и прогнозировать их надёжность при проектировании и эксплуатации*.

Чтобы создать математическую модель надёжности, нужно учитывать, что модели носят вероятностный характер и хорошо работают при высоком уровне ошибок.

* Аналитические модели дают возможность рассчитать количественные показатели надежности, основываясь на данных о поведении программы в процессе тестирования.
* Эмпирические модели базируются на анализе структурных особенностей программ

**Аналитические модели,**в свою очередь, делятся на две группы: **динамические**и **статические.**

В **динамических моделях** поведение ПО, т. е. появление отказов, анализируется во времени. От способа фиксации момента отказа модели могут быть модели с непрерывным временем или модели с дискретным временем.

В **статических моделях**учитывают зависимость количества ошибок от числа тестовых прогонов.

***Эти математические модели предназначены для оценки:***

- показателей надёжности комплексов программ в процессе отладки;

- количества ошибок, оставшихся не выявленными;

- времени, необходимого для обнаружения следующей ошибки в функционирующей программе;

- времени, необходимого для выявления всех ошибок с заданной вероятностью.

**В настоящее время предложен ряд математических моделей, основными из которых являются:**

- экспоненциальная модель изменения ошибок в зависимости от времени отладки;

- модель, учитывающая дискретно - понижающуюся частоту появления ошибок как линейную функцию времени тестирования и испытаний;

- модель, базирующаяся на распределении Вейбула;

*При обосновании математических моделей выдвигаются некоторые гипотезы о характере проявления ошибок в комплексе программ*.

Наиболее обоснованными представляются предположения, на которых базируется первая Экспоненциальная модель изменения ошибок в процессе отладки и которые заключаются в следующем:

1. Любые ошибки в программе являются независимыми и проявляются в случайные моменты времени.

2. Время работы между ошибками определяется средним временем выполнения команды на данной ЭВМ и средним числом команд, исполняемым между ошибками. Это означает, что интенсивность проявления ошибок при реальном функционировании программы зависит от среднего быстродействия ЭВМ.

3. Выбор отладочных тестов должен быть представительным и случайным, с тем чтобы исключить концентрацию необнаруженных ошибок для некоторых реальных условий функционирования программы.

4. Ошибка, являющаяся причиной искажения результатов, фиксируется и исправляется после завершения тестирования либо вообще не обнаруживается.

Вторая модель построена на основе гипотезы о том, что частота проявления ошибок (интенсивность отказов) линейно зависит от времени испытания  между моментами обнаружения последовательных ***i - й***

Особенностью Третьей модели является учёт ступенчатого характера изменения надёжности при устранении очередной ошибки. В качестве основной функции рассматривается распределение времени наработки на отказ ***P(t)***. Если ошибки не устраняются, то интенсивность отказов является постоянной, что приводит к экспоненциальной модели для распределения:

Аналитические - статические модели

Тестируя программу в течение некоторого времени, собирают статистику об ошибках. В момент оценки надежности по протоколу искусственных ошибок все ошибки делятся на собственные и искусственные.

**Модель Липова**является модификацией модели Миллса. Она основана на анализе вероятности обнаружения ошибок при использовании различного числа тестов.

Использование **простой интуитивной модели** предполагает проведение тестирования двумя группами программистов, использующими независимые тестовые наборы, независимо одна от другой.

**Модель Нельсона**при расчете надежности ПО учитывает вероятность выбора определенного тестового набора для очередного выполнения программы.

Что касается **эмпирических моделей,**то они, как правило, основываются на сложности ПО и характеризуются размером программного обеспечения:

* количеством программных модулей,
* количеством и сложностью межмодульных интерфейсов.

Под программным модулем в данном случае понимают программную единицу, выполняющую определенную функцию и взаимосвязанную с другими модулями ПО

Существует несколько разновидностей модели сложности, основанных на метриках сложности ПО. В каждой из них определяется некоторая оценка сложности программы, которая считается пропорциональной ее надежности.

1. **Анализ рисков и характеристик качества программного обеспечения при внедрении.**

**Риск** - это возможность понести потери.

**Риск проекта ПО**- это возможность:

1) снижения качества конечного продукта,

2) повышения стоимости его разработки,

3) задержки окончания разработки или срыва проекта (то есть, отказа от проекта).

**Величина риска** представляет собой **R = V \*P** произведение величины потерь **V** от нежелательного события в проекте и вероятности **P** наступления этого события

**Эффективное управление риском состоит в принятии (по каждому риску) компромиссных решений по**:

учету рисков и анализу старых проектов;

оценке трудоемкости устранения определенного риска,

величине потенциального отрицательного воздействия этого риска на проект,

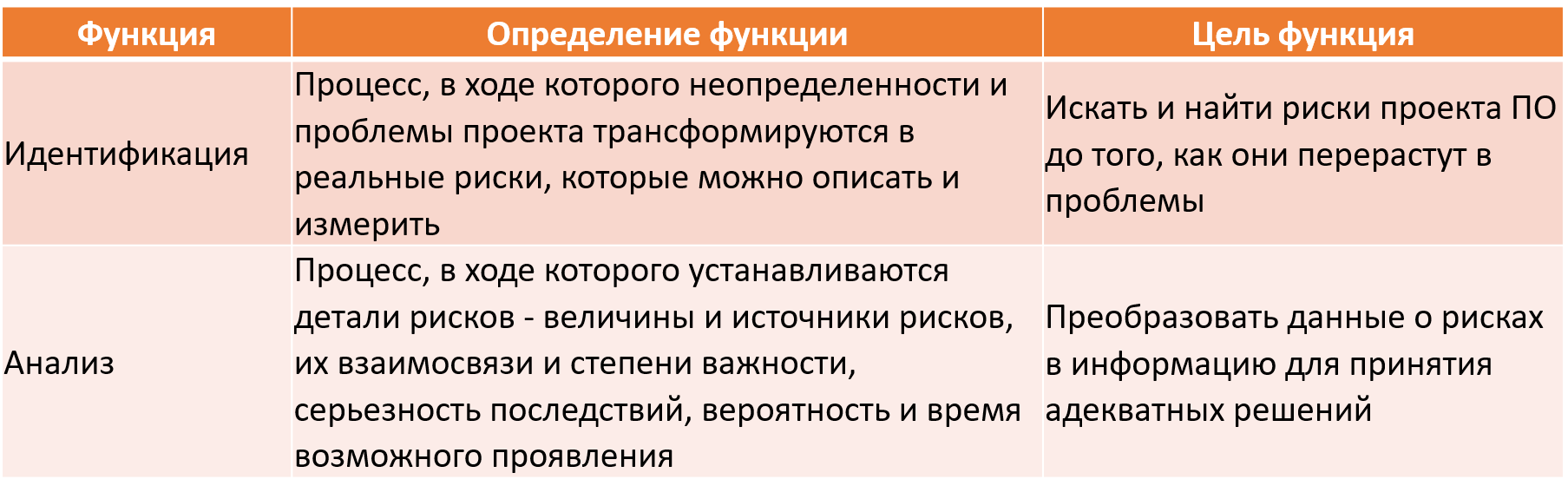
в правильной оценке взаимозависимости устраняемых рисков и возможного влияния принятых в определенный (текущий) момент времени решений на состояние проекта в будущем;

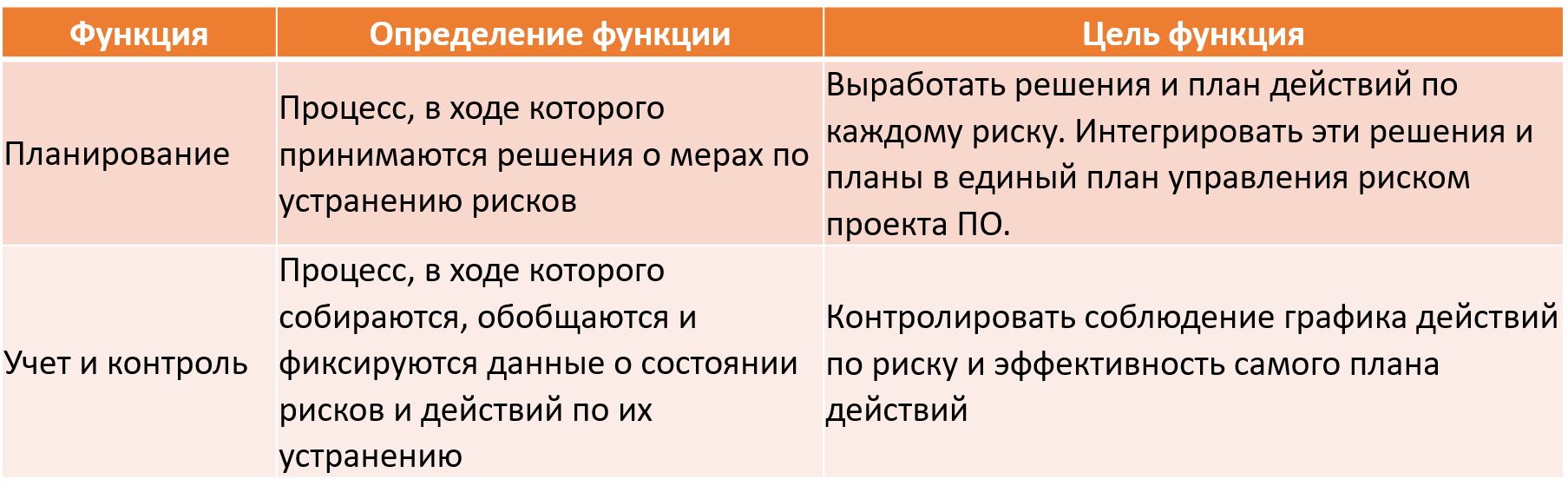
резервировании в проекте времени на борьбу с рисками.

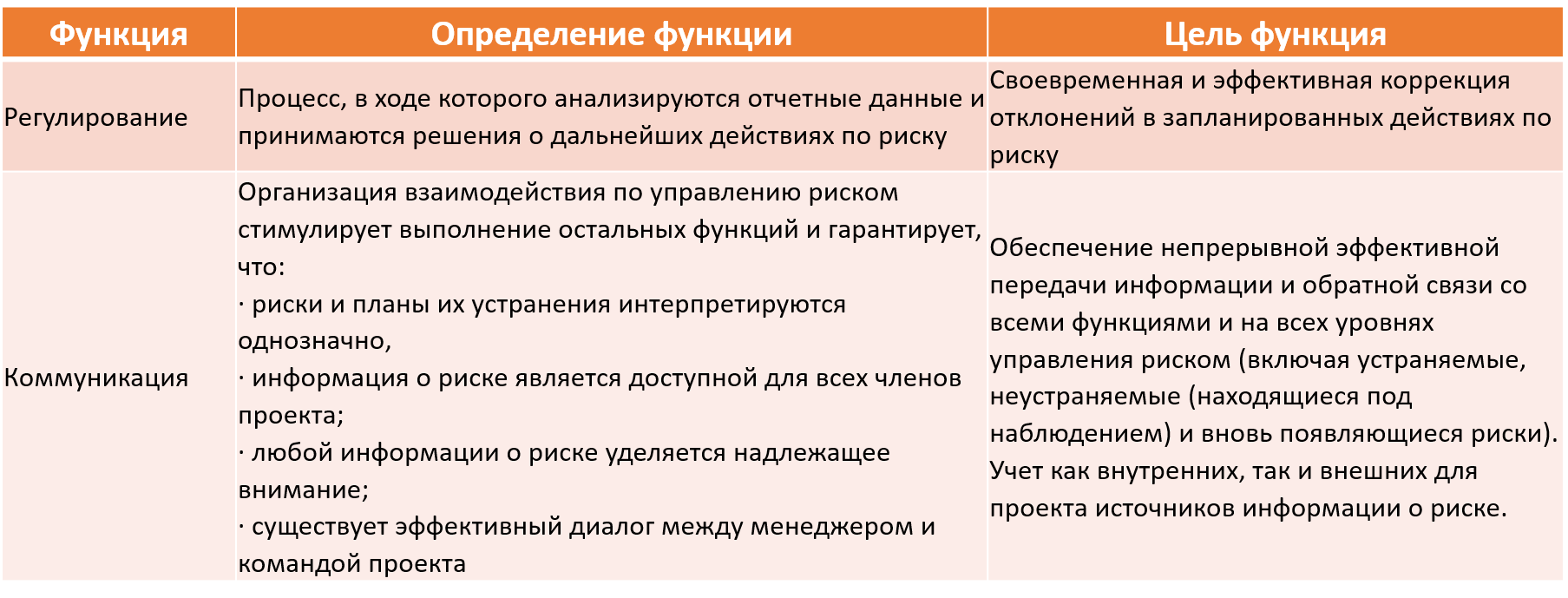
**Базовыми конструкциями концепции управления риском являются:**

* функции управления риском,
* таксономия (классификация) риска;
* методология оценки и управления риском.

ФУНКЦИИ УПРАВЛЕНИЯ РИСКОМ







**Таксономия** - это некая классификация или категоризация рисков:

-источнику риска

-Часто специфична для одной области применения

-Может включать множественные уровни

-Она составлена с учетом типовых процессов жизненного цикла ПО и охватывает наиболее общие -области риска проекта, касающиеся характеристик ПО, среды и процессов разработки и ограничений проекта.

**Исследования риска** - это длительный процесс, в ходе которого предпринимаются совместные усилия ведущей организацией по вопросам управления риском и организацией-клиентом:

-по изучению существующей практики разработки конкретного проекта,

-оценке альтернативных приемов управления риском,

-выработке концепции управления риском клиента,

-обучению методам управления риском

-созданию необходимой инфраструктуры и плана управления риском ПО в организации-клиенте.

**Процесс управления риском проекта ПО включает следующие четыре стадии:**

**согласование целей** - определение нужд и целей проекта, достижение соглашений по управлению риском,

**подготовка работ** - планирование и координация предстоящих работ по оцениванию риска проекта ПО,

**оценка риска** - выполнение функций управления риском и получение рекомендаций по управлению риском проекта ПО,

**подготовка к устранению риска** - разработка рекомендаций по устранению рисков по всем областям устранения риска, разработка плана управления риском и приведение его в действие.

**Разработано 3 методологии** :

1. Оценивание риска ПО - SRE (от Software Risk Evaluation),
2. Непрерывное управление риском - CRM (от Continuous Risk Мanagement),
3. Коллективное управление риском - TRM (от Team Risk Management).

**Оценивание риска ПО**

Методология SRE — это способ определения, анализа, контроля и устранения рисков ПО. Она используется на начальном этапе разработки проекта и на протяжении всего жизненного цикла проекта. Функции управления рисками в SRE делятся на основные и дополнительные.

**К основным функциям управления риском относятся:**

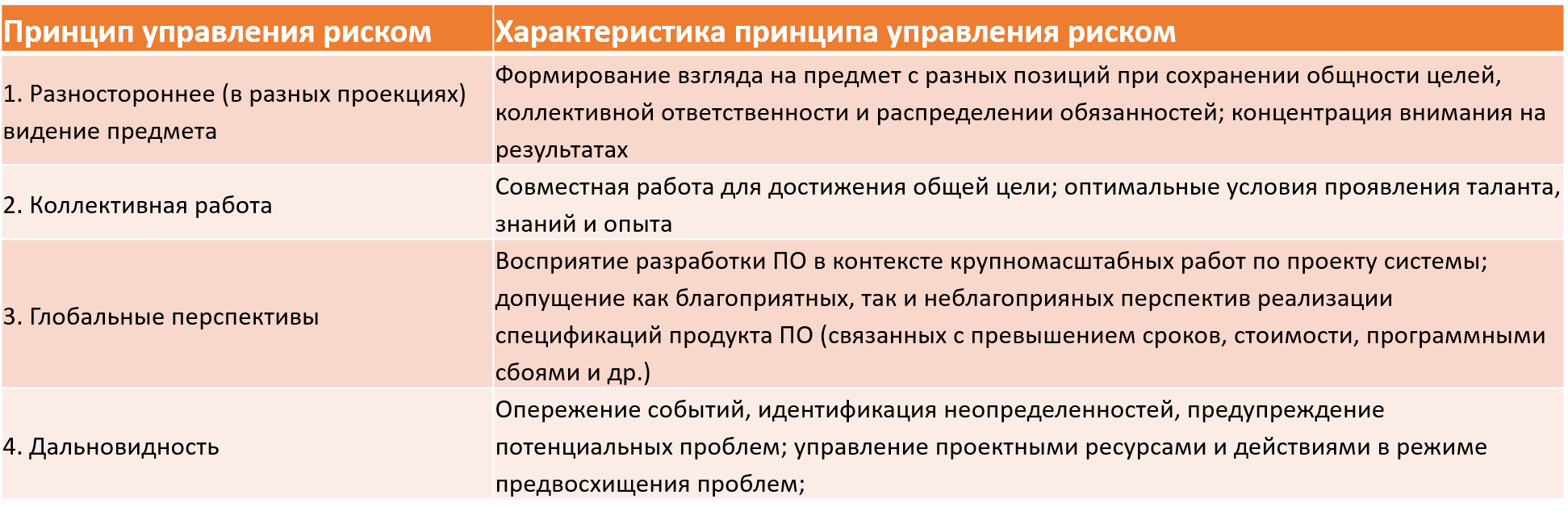
* обнаружение,
* спецификация,
* оценивание,
* структурирование (консолидация)
* устранение рисков.

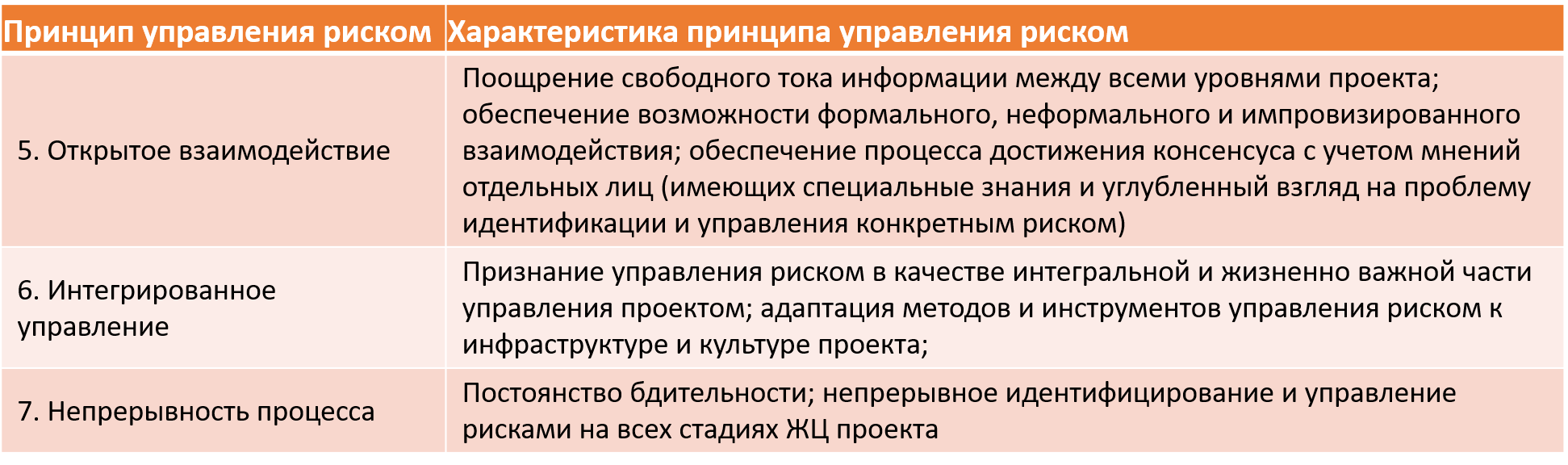
Выполнение функции **обнаружения рисков** обеспечивает систематический и полный охват всех потенциальных областей риска с применением адекватных инструментов и технологий, в частности, опросника TBQ.

Выполнение функции **спецификации риска** касается фиксации всех аспектов идентифицированного риска ПО. Спецификация риска представляет собой структуру, которая упрощает решение задач приоритезации рисков, локализации источников и причин возникновения рисков, определения методов и усилий, предпринимаемых для устранения рисков в источниках их возникновения.

Функция **оценивания риска** заключается в определении величины каждого риска ПО, что служит основанием для присваивания приоритета риску и выявления наиболее важных на текущий момент рисков проекта.

Методология основана на определенных принципах управления риском в ходе всего ЖЦ проекта и не зависит от конкретных применяемых методов и инструментов оценки и устранения риска.





**Коллективное управление риском**

осуществление совместного управ. риском со стороны заказчика проекта ПО и его исполнителя.

Методология основана на семи принципах управление риском и философии бригадной работы, к которым добавляет две функции - **инициирование коллективной работы и собственно коллективное управление риском.**

Эти функции выполняются над каждым риском последовательно, но действия по управлению риском проекта в целом могут быть как последовательными, так и параллельными.

Инициировать коллективную работу может либо заказчик, либо исполнитель.

**Организационная структура управления риском**

* Процесс управления риском проекта ПО начинается с создания координационной группы по управлению риском проекта.
* Численность группы может колебаться от одного человека с частичной занятостью до нескольких человек с полной занятостью.
* Кандидатами в координационную группу могут быть члены бригады проекта, представители пользователей, исполнителей и соисполнителей.
* Лидером этой группы является представитель нанимаемой независимой группы экспертов, который несет ответственность за выполнение обеспечивающих функций планирования и координации.

**План управления риском проекта ПО** определяет, каким образом процесс управления риском будет выполняться в рамках конкретного проекта разработки ПО, и указывает процессы, действия, этапы работ, методы и инструменты, используемые членами бригады проекта, ответственными за управление риском проекта ПО.

1. **Целесообразность разработки модулей адаптации.**

Адаптация программы (пп.9 п.2 ст.1270 ГК РФ) – это внесение изменений для функционирования ПО на конкретных технических средствах или под управлением конкретных программ пользователя. Программное обеспечение, способное к адаптации, имеет конкурентные преимущества:

Решает более широкий круг задач для пользователей.

Способствует формированию сообщества пользователей, что увеличивает распространение приложения.

Технология COM (Component Object Model):

COM – это стандарт Microsoft для создания ПО из взаимодействующих компонентов. Она предоставляет гибкую архитектуру для создания и управления объектами, используется в технологиях OLE и ActiveX.

Основные строительные блоки COM:

COM-интерфейс: набор абстрактных функций, которые задают общие принципы для всех производных классов.

COM-объект: объект CoClass, который реализует один или несколько COM-интерфейсов.

COM/ActiveX сервер: модуль (EXE, DLL или OCX), содержащий машинный код COM или ActiveX объектов.

Class factory: объект, создающий COM-объекты из CoClass.

Type library: файл с информацией о типах данных, используемых COM/ActiveX сервером.

Интерфейсы COM:

Интерфейсы представлены как указатели на vtable (virtual table), содержащие ссылки на методы класса.

Интерфейсы имеют уникальный идентификатор (GUID), обеспечивающий отсутствие конфликтов имен при обновлениях ПО.

Основные методы интерфейса:

QueryInterface: обеспечивает доступ к указателю на интерфейс.

AddRef и Release: управляют временем жизни объекта.

COM-сервер:

Local server: EXE-модуль, работающий в отдельном адресном пространстве.

Remote server: EXE-модуль, работающий на удаленной машине.

Class factories:

Используются для создания и регистрации COM-объектов.

Позволяют клиентским приложениям создавать объекты класса, зная только их CLSID.

Type library:

Содержит информацию об используемых типах данных и интерфейсах.

Важна для объектов, предназначенных для распространения и использования многими пользователями.

Dispatch interface (IDispatch):

Обеспечивает позднее связывание и маршаллинг, реализует таблицу диспетчерских идентификаторов (dispID).

Ограничен на типы данных, допускается использовать 13 стандартных типов.

Vtable-интерфейсы:

Позволяют вызывать методы интерфейса через ссылки из vtable, если известен порядок записи ссылок на методы и типы аргументов.

Преимущества двойных интерфейсов:

Позволяют получать указатели на ресурсы сервера по именам при компиляции.

Обеспечивают прямой доступ к ресурсам сервера через vtable-интерфейсы, что увеличивает скорость взаимодействия.

Обладают преимуществами проверки соответствия типов на этапе компиляции (раннее связывание).

1. **Вредоносные программы: классификация, методы обнаружения.**

**Компьютерный вирус** — это программа, способная создавать свои копии и внедрять их в файлы и системные области компьютера

Основная черта – способность распространяться при запуске!

**Вредоносные программы** — это программы, предназначенные для незаконного доступа к информации, для скрытого использования компьютера или для нарушения работы компьютера и компьютерных сетей.

**Последствия внедрения вирусов**

* Снижение производительности ПК;
* Беспорядочная смена пользовательских настроек;
* Появление новых сомнительных панелей инструментов(аддонов);
* Вирусы могут глубоко внедряться в  сложные механизмы работы ОС так, чтобы в значительной степени осложнить их обнаружение и уничтожение;
* Вирус способен уничтожить всю ОС.

**Виды вредоносных программ**

**Троян** — это вирус, проникающий на компьютер под видом безвредной программы.

Он не имеет собственного механизма распространения, и этим отличается от вирусов, которые распростра-няются, прикрепляя себя к обычной программе, и от «червей», которые копируют себя по сети.

Если же троян несет вирусное тело, то он становится очагом «заразы»

**Шпион**— вирус, скрытно устанавливающийся на ПЭВМ в целях полного или частичного контроля за работой компьютера и пользователя без согласия последнего. Существуют и другие определения шпионов.

собирать информацию о наиболее часто посещаемых сайтах;

запоминать нажатия клавиш на клавиатуре,

записывать скриншоты экрана и отправлять информацию хакерам;

несанкционированно и удаленно управлять компьютером;

инсталлировать на компьютер пользователя дополнительные программы;

сканировать порты, пароли и др.;

изменять параметры ОС (руткиты, перехватчики управления);

перенаправлять активность браузеров, что влечет за собой посещение веб-сайтов вслепую с риском заражения вирусами.

**Сетевой червь** — разновидность самовоспроизводящихся вирусов — программ, распространяющихся в локальных и глобальных компьютерных сетях.

Черви являются самостоятельными программами, которые могут использовать различные механизмы распространения.

**Руткит**— вирусная программа или набор программ, использующих технологии сокрытия системных объектов (файлов, процессов, драйверов и др.) посредством обхода механизмов системы.

Руткит позволяет хакеру закрепиться во взломанной системе и скрыть следы своей деятельности.

В системе Windows под термином «руткит» принято понимать программу, которая внедряется в систему и перехватывает системные функции или производит замену системных библиотек.

**Можно отметить следующие основные признаки заражения ПЭВМ вирусами:**

вывод на экран непредусмотренных сообщений или изображений;

подача непредусмотренных звуковых сигналов;

неожиданное открытие и закрытие лотка CD-ROM-устройства;

самопроизвольный запуск на компьютере каких-либо программ;

при наличии на ПЭВМ межсетевого экрана, появление предупреждений о попытке программы выйти в Интернет, хотя вы это никак не инициировали;

друзьям или знакомым идут от вас сообщения, которые вы не отправляли;

наличие в почте массы сообщений без обратного адреса и заголовка.

**К косвенным признакам заражения ПЭВМ относятся:**

частые зависания и сбои в работе компьютера;

медленная работа компьютера при запуске программ;

невозможность загрузки ОС;

исчезновение файлов и каталогов или искажение их содержимого;

частое несанкционированное обращение к жесткому диску;

зависание интернет-браузера.

Чтобы снизить риск потерь от воздействия вредоносных программ, рекомендуется:

использовать современные ОС;

включить режим автоматического обновления ОС;

постоянно работать на ПЭВМ исключительно под правами пользователя;

использовать антивирусы известных производителей с автоматическим обновлением сигнатурных баз;

использовать персональный Firewall, контролирующий выход в Интернет с персонального компьютера на основании политик, которые устанавливает сам пользователь

ограничить физический доступ к компьютеру посторонних лиц;

использовать внешние носители информации от проверенных источников;

не открывать компьютерные файлы, полученные от ненадежных источников;

отключить автозапуск со сменных носителей.

1. **Антивирусные программы: классификация, сравнительный анализ.**

Самым популярными и эффективными антивирусными программами являются **антивирусные сканеры**, принцип работы которых основан на проверке файлов, секторов и системной памяти в целях поиска в них известных и новых вирусов.

Для поиска известных вирусов используются так называемые маски.

**Маской вируса** является некоторая последовательность кода, специфичная для этого конкретного вируса.

**Антивирусные сканеры**

Сканеры подразделяются на **резидентные**, производящие сканирование «на лету», и **нерезидентные**, обеспечивающие проверку системы только по запросу.

**Резидентные сканеры** обеспечивают более надежную защиту системы, поскольку они немедленно реагируют на появление вируса, в то время как нерезидентный сканер способен опознать вирус только во время своего очередного запуска.

**К достоинствам** сканеров относиться их универсальность, к **недостаткам** – размеры антивирусных баз, которые сканерам приходится хранить и пополнять, и относительно небольшая скорость поиска вирусов

**Программы-детекторы** осуществляют поиск характерной для конкретного вируса сигнатуры в оперативной памяти и в файлах, и  при обнаружении выдают соответствующее сообщение.

**Недостатком** таких антивирусных программ является то, что они могут находить только те вирусы, которые известны разработчикам таких программ.

**Антивирусные блокировщики** — это резидентные программы, перехватывающие вирусоопасные ситуации и  сообщающие об этом пользователю.

К вирусоопасным относятся вызовы на открытие для записи в  выполняемые файлы, запись в  загрузочный сектор диска, которые характерны для вирусов в моменты их размножения.

К **достоинствам блокировщиков** относится их способность обнаруживать и блокировать вирус на самой ранней стадии его размножения, что, кстати, бывает очень полезно в  случаях, когда давно известный вирус постоянно активизируется.

**Программы-доктора,** или **фаги**, а также **программы-вакцины** не только находят зараженные вирусами файлы, но и «лечат» их, т. е. удаляют из файла тело программы-вируса, возвращая файлы в  исходное состояние.

В  начале своей работы фаги ищут вирусы в  оперативной памяти, уничтожая их, и  только затем переходят к  «лечению» файлов.

Среди фагов выделяют полифаги, т. е. программы-доктора, предназначенные для поиска и  уничтожения большого числа вирусов.

Наиболее известные из них Norton AntiVirus и  Doctor Web.

Учитывая, что постоянно появляются новые вирусы, программы-детекторы и  программы-доктора *быстро устаревают, и требуется регулярное обновление версий*.

**Программы-ревизоры** относятся к самым надежным средствам защиты от вирусов.

Ревизоры **запоминают исходное состояние программ**, каталогов и системных областей диска тогда, когда компьютер не заражен вирусом, а  затем периодически или по желанию пользователя **сравнивают текущее состояние с  исходным**.

Обнаруженные изменения выводятся на экран монитора. Как правило, сравнение состояний производят сразу после загрузки ОС. К числу программ - ревизоров относится широко распространенная в России программа Adinf

Программы-фильтры, или сторожа, представляют собой небольшие резидентные программы, предназначенные для обнаружения cktle.ob —**подозрительных действий** при работе компьютера, характерных для вирусов:

* попытки коррекции файлов с расширениями COM, EXE;
* изменение атрибутов файла;
* прямая запись на диск по абсолютному адресу;
* запись в загрузочные секторы диска;
* загрузка резидентной программы

Своевременное обнаружение вирусов и их уничтожение в ОС Windows обеспечивают антивирусные программы:

**ESET NOD32**  — отличается высокой скоростью сканирования ресурсов компьютера в специальном режиме, не мешая остальным запущенным процессам;

**Kaspersky Total Security** — является кроссплатформенным антивирусом и  применяется для защиты сразу нескольких синхронизированных устройств. Антивирус получил несколько полезных защитных функций.

**Norton Security** — отличается наличием системы интеллектуального обучения и имеет многоуровневый принцип защиты, который не допускает к запуску вредоносное ПО;

1. **Файрвол: задачи, сравнительный анализ, настройка.**

Межсетевой экран или сетевой экран — комплекс аппаратных или программных средств, осуществляющий контроль и фильтрацию проходящих через него сетевых пакетов в соответствии с заданными правилами.

Фаерволом может быть как программное средство, так и комплекс ПО и оборудования. И поначалу они были чисто железными, как и давшие им название противопожарные сооружения.

Основной задачей сетевого экрана является защита компьютерных сетей или отдельных узлов от несанкционированного доступа. Также сетевые экраны часто называют фильтрами, так как их основная задача — не пропускать пакеты, не подходящие под критерии, определённые в конфигурации.

Firewall контролирует порты и проходящие через них пакеты. Важной функцией Firewall является наблюдение за всеми установленными и запущенными приложениями.

**Виды Firewall в зависимости от охвата контролируемых потоков**

**традиционный сетевой** (или межсетевой) экран - программа на шлюзе или аппаратное решение, контролирующие входящие и исходящие потоки данных между подключенными сетями.

**персональный сетевой экран** — программа, установленная на пользовательском компьютере и предназначенная для защиты от несанкционированного доступа только этого компьютера.

**вырожденный случай** — использование традиционного сетевого экрана сервером, для ограничения доступа к собственным ресурсам.

**Виды Firewall в зависимости от уровня, на котором происходит контроль доступа**

**сетевой уровень** - когда фильтрация происходит на основе адресов отправителя и получателя пакетов, номеров портов транспортного уровня модели OSI и статических правил, заданных администратором;

**сеансовый уровень** отслеживающие сеансы между приложениями, не пропускающие пакеты нарушающих спецификации TCP/IP, часто используемых в злонамеренных операциях — сканировании ресурсов, взломах через неправильные реализации TCP/IP, обрыв/замедление соединений, инъекция данных.

**уровень приложений** -фильтрация на основании анализа данных приложения, передаваемых внутри пакета. Такие типы экранов позволяют блокировать передачу нежелательной и потенциально опасной информации, на основании политик и настроек.

**Виды Firewall в зависимости от отслеживания активных соединений**

**stateless (простая фильтрация)**фильтруют поток данных исключительно на основе статических правил;

**stateful, stateful packet inspection(фильтрация с учётом контекста)**с отслеживанием текущих соединений и пропуском только таких пакетов, которые удовлетворяют логике и алгоритмам работы соответствующих протоколов и приложений. Такие типы сетевых экранов позволяют эффективнее бороться с различными видами DoS-атак и уязвимостями некоторых сетевых протоколов.

Среди этих функций — URL-фильтрация, антивирус, защита от спама и ботов, предотвращение утечек данных, контроль доступа с мобильных устройств, а также многие другие, делающие файрвол мультисервисным шлюзом безопасности. С помощью модульного подхода, управляемого программным способом, можно добавлять и развертывать эти функции, усиливая защиту сети и решая новые проблемы по мере их возникновения.

**Проблемы, не решаемые с помощью Firewall**

не защищает узлы сети от проникновения через «люки» или уязвимости ПО;

не обеспечивает защиту от многих внутренних угроз, в первую очередь — утечки данных;

не защищает от загрузки пользователями вредоносных программ, в том числе вирусов;

Ashampoo FireWall FREE – бесплатный брандмауэр, перед установкой которого необходимо убедится, что в системе не установлена антивирусная программ или какой-то другой межсетевой экран.

1. **Групповые политики. Аутентификация. Учетные записи.**

**Групповая политика** – это один из главных инструментов для настройки Windows, с помощью которого можно как задействовать множество функций системы, так и отключить их.

С помощью групповых политик можно настроить систему для пользователя, под которым произведен вход в систему, для других пользователей данного компьютера или других пользователей на других компьютерах в доменной сети.

Групповые политики доступны только в профессиональных, максимальных и корпоративных версиях Windows, у пользователей домашней версии Windows данный набор инструментов отключен.

Также возможно задать групповые политики для всех персональных компьютеров в  сети:

* установить открытие определенных сайтов в  качестве домашней страницы
* запретить доступ к  различным настройкам панели управления
* закрыть доступ к определенным папкам
* запретить смену заставки и т. д.

Системный администратор может менять настройки компьютера пользователя с помощью групповых политик на сервере.

**Локальные групповые политики**

Групповые политики используются не только системными администраторами, их можно использовать для настройки своего же компьютера.

Например, с помощью групповых политик можно изменить экран входа в Windows, также можно ограничить доступ к некоторым папкам другим пользователям данного компьютера, можно управлять паролями, отображением папок, отключить автоматическое обновление, разрешать запуск только определенных приложений, отключить доступ к  панели управления и др.

Можно добавить через консоль групповые политики и настроить систему для других пользователей данного компьютера.

Чтобы настроить групповую политику в профессиональной, корпоративной и (или) максимальной версиях Windows.

*Например, можно отключить автозапуск съемных носителей, чтобы при подключении съемного носителя не появлялось окно с вопросом, что делать с этим устройством*.

**Аутентификация** — проверка подлинности пользователя путем сравнения введенного им пароля с  паролем, сохраненным в  базе данных пользователей.

Аутентификация пользователя включает сервер безопасности HTTP FireWall-1, который предоставляет механизм реализации указанной проверки.

Сервер FireWall обеспечивает **три вида аутентификации**:

1) аутентификация пользователя, которая позволяет администратору давать каждому пользователю свои привилегии доступа;

2) аутентификация клиентов, которая дает механизм для аутентификации пользователя любого приложения — стандартного — или собственной разработки;

3) аутентификация сессий, дающая прозрачную аутентификацию каждой сессии, что может быть интегрировано с любым приложением Сервер FireWall-1 поддерживает следующие схемы аутентификации для каждого пользователя:

S/Key  — пользователю требуется ввести значения S/Key для данной итерации;

SecurID  — пользователю требуется ввести номер, показанный на SecurID карте Security Dynamics;

по паролю  — от пользователя требуют ввести его (или ее) пароль OS;

внутренняя  — пользователь должен ввести его (или ее) внутренний пароль FireWall-1 на мосте;

RADIUS — пользователь должен ввести ответ на запрос сервера RADIUS;

AssureNet Pathways — пользователь должен ввести ответ на запрос сервера AssureNet Pathway.

Учетная запись  — это данные о  пользователе, хранящиеся в  компьютерной системе. Она нужна для верификации человека, предоставления ему доступа к личным настройкам и данным.

**Обычные учетные записи** предназначены для ежедневной работы пользователей.

**Администраторские учетные** записи дают полный контроль над компьютером. Это означает, что такая учетная запись позволяет владельцу изменять любые данные, принадлежащие системе, менять параметры безопасности и устанавливать программы на других компьютерах. Администраторские учетные записи обычно применяются в офисах.

**Гостевые учетные записи**, как правило, защищены стандартным логином и паролем и служат для обеспечения временного доступа

Учетная запись может вмещать фотографии, опросные данные, тайные вопросы, ответы на которые известны лишь пользователю и служат еще одним средством проверки.

1. **Тестирование защиты программного обеспечения.**

***Тестирование***– процесс проверки соответствия заявленных к продукту требований и реально реализованной функциональности, осуществляемый путем наблюдения за его работой в искусственно созданных ситуациях и на ограниченном наборе тестов, выбранных определенным образом.

**Для каждого уровня тестирования может быть определено:**

1. Цель
2. Объекты тестирования
3. Прослеживание связи с базисом тестирования (при наличии)
4. Критерии входа и выхода
5. Артефакты процесса тестирования, которые будет поставлять отдел тестирования - тестовые сценарии, протоколы тестирования, отчетность о результатах и другие
6. Тестовые методики
7. Измерения и метрики
8. Инструментарий

***Тестирование по требованиям безопасности*** - процесс выявления наличия или отсутствия уязвимостей в продукте в искусственно созданных ситуациях и на ограниченном наборе тестов, выбранных определенным образом.

**Тестирование защищенности:** Тестирование с целью оценить защищенность программного продукта

Объекты тестирования:

* пароли
* шифрование
* аппаратные устройства доступа
* уровни доступа к информации
* авторизация
* скрытые каналы
* безопасность на физическом уровне

Если при тестировании удалось обнаружить потенциально опасные участки кода,  это еще не значит, что найдены уязвимости (поэтому такие сигнатуры и называются ПОТЕНЦИАЛЬНО опасными).

Необходимо провести «экспериментальное тестирование», т.е. попробовать использовать опасные участки кода с целью компрометации приложения.

**При проведении тестирования по требованиям безопасности проводятся следующие испытания:**

аудит безопасности кода (направленный на выявление уязвимостей);

функциональное тестирование (характеристик безопасности тестируемого программного обеспечения);

экспериментальное тестирование (проверяется возможность, или невозможность эксплуатации обнаруженных сигнатур).

**ПРИЕМЫ ВЫЯВЛЕНИЯ УЯЗВИМОСТЕЙ**

* + Ручной (экспертный анализ)
  + Статически анализ безопасности (по шаблону)
  + Динамический анализ безопасности

**Уровни тестирования:**

* Модульное тестирование (Unit testing)

Этот уровень тестирования позволяет проверить функционирование отдельно взятого элемента системы. Что считать элементом – модулем системы определяется контекстом.

* Интеграционное тестирование (Integration testing)

Данный уровень тестирования является процессом проверки взаимодействия между программными компонентами/модулями.

* Системное тестирование (System testing)

Системное тестирование охватывает целиком всю систему.

Большинство функциональных сбоев должно быть идентифицировано еще на уровне модульных и интеграционных тестов.

В свою очередь, системное тестирование, обычно фокусируется на нефункциональных требованиях – безопасности, производительности, точности, надежности т.п.

На этом уровне также тестируются интерфейсы к внешним приложениям, аппаратному обеспечению, операционной среде и т.д.

**Виды тестирования**

**Приёмочное тестирование**

Проверяет поведение системы на предмет удовлетворения требований заказчика.

**Установочное тестирование**

Из названия следует, что данные тесты проводятся с целью проверки процедуры инсталляции системы в целевом окружении.

**Альфа- и бета-тестирование**

Перед тем, как выпускается программное обеспечение, как минимум, оно должно проходить стадии альфа (внутреннее пробное использование) и бета (пробное использование с привлечением отобранных внешних пользователей) версий.

Отчеты об ошибках, поступающие от пользователей этих версий продукта, обрабатываются в соответствии с определенными процедурами, включающими подтверждающие тесты (любого уровня), проводимые специалистами группы разработки.

**Функциональные тесты/тесты соответствия**

Эти тесты могут называться по разному, однако, их суть проста – проверка соответствия системы, предъявляемым к ней требованиям, описанным на уровне спецификации поведенческих характеристик.

**Достижение и оценка надежности (Reliability achievement and evaluation)**

Помогая идентифицировать причины сбоев, тестирование подразумевает и повышение надежности программных систем.

Случайно генерируемые сценарии тестирования могут применяться для статистической оценки надежности.

**Регрессионное тестирование (Regression testing)**

Определение успешности регрессионных тестов (IEEE 610–90 “Standard Glossary of Software Engineering Terminology”) гласит: “повторное выборочное тестирование системы или компонент для проверки сделанных модификаций не должно приводить к непредусмотренным эффектам”.

На практике это означает, что если система успешно проходила тесты до внесения модификаций, она должна их проходит и после внесения таковых.

**Тестирование производительности (Performance testing)**

Специализированные тесты проверки удовлетворения специфических требований, предъявляемых к параметрам производительности.

Существует особый подвид таких тестов, когда делается попытка достижения количественных пределов, обусловленных характеристиками самой системы и ее операционного окружения.

**Нагрузочное тестирование (Stress testing)**

Необходимо понимать отличия между рассмотренным выше тестированием производительности с целью достижения ее реальных (достижимых) возможностей производительности и выполнением программной системы c повышением нагрузки, вплоть до достижения запланированных характеристик и далее, с отслеживанием поведения на всем протяжении повышения загрузки системы.

**Сравнительное тестирование (Back-to-back testing)**

Единичный набор тестов, позволяющих сравнить две версии системы.

**Восстановительные тесты (Recovery testing)**

Цель – проверка возможностей рестарта системы в случае непредусмотренной катастрофы (disaster), влияющей на функционирование операционной среды, в которой выполняется система.

**Конфигурационное тестирование (Configuration testing)**

В случаях, если программное обеспечение создается для использования различными пользователями (в терминах “ролей”), данный вид тестирования направлен на проверку поведения и работоспособности системы в различных конфигурациях.

**Тестирование удобства и простоты использования (Usability testing)**

Цель – проверить, насколько легко конечный пользователь системы может ее освоить, включая не только функциональную составляющую – саму систему, но и ее документацию; насколько эффективно пользователь может выполнять задачи, автоматизация которых осуществляется с использованием данной системы; наконец, насколько хорошо система застрахована (с точки зрения потенциальных сбоев) от ошибок пользователя.

**Тесты, базирующиеся на блок-схеме (Control-flow-based criteria)**

Набор тестов строится исходя из покрытия всех условий и решений блок-схемы. В какой-то степени напоминает тесты на основе конечного автомата.

Отличие – в источнике набора тестов.

Максимальная отдача от тестов на основе блок-схемы получается когда тесты покрывают различные пути блок-схемы – по-сути, сценарии потоков работ (поведения) тестируемой системы.

Адекватность таких тестов оценивается как процент покрытия всех возможных путей блок-схемы.

1. **Средства и протоколы шифрования сообщений.**

***Шифрование*** – обратимое преобразование информации в целях скрытия от неавторизованных лиц, с предоставлением в это же время авторизованным пользователям доступа к ней.

Главным образом, шифрование служит задачей соблюдения конфиденциальности передаваемой информации.

Важной особенностью алгоритма шифрования является использование ключа, который утверждает выбор конкретного преобразования из совокупности возможных для данного алгоритма.

***Криптография*** – наука о защите информации от прочтения её посторонними.

Защита достигается шифрованием, т.е. преобразованием, которое делает защищенные данные трудно раскрываемыми по входным данным без знания специальной ключевой информации – ключа.

Под ключом понимается легко изменяемая часть криптосистемы, хранящаяся в тайне и определяющая какое шифрующее преобразование из возможных выполняется в данном случае.





Криптографические методы защиты информации  — это специальные методы шифрования, кодирования или иного преобразования информации, в  результате которого ее содержание становится недоступным без предъявления ключа криптограммы и  обратного преобразования.

Криптографический метод защиты, безусловно, самый надежный, так как охраняется непосредственно сама информация, а не доступ к ней (например, зашифро-ванный файл нельзя прочесть даже в случае кражи носителя).

Данный метод защиты реализуется в виде программ или пакетов программ.

**Все методы можно свести к следующим классам:**

***подстановка*** — символы шифруемого текста заменяются символами того же или другого алфавита в соответствии с заранее определенным правилом;

***перестановка*** — символы шифруемого текста переставляются по некоторому правилу в пределах заданного блока передаваемого текста;

***аналитическое преобразование***  — шифруемый текст преобразуется по некоторому аналитическому правилу, например, гаммирование заключается в  наложении на исходный текст некоторой псевдослучайной последовательности, генерируемой на основе ключа;

***комбинированное преобразование*** — представляет собой последовательность (с возможным повторением и чередованием) основных методов преобразования, применяемую к блоку (части) шифруемого текста

***Шифрование с закрытым ключом*** (симметричное шифрование) осуществляет преобразование данных для предотвращения их просмотра третьей стороной.

В данном способе для шифрования и расшифровки данных используется общий закрытый ключ.

При шифровании с закрытым ключом для дешифровки данных используется один закрытый ключ. *Необходимо обезопасить этот ключ от несанкционированного доступа*, потому что любое обладающее им лицо может использовать его для расшифровки данных или шифрования собственных данных с подменой источника.

**Асимметричные алгоритмы** шифрования имеют математические ограничения на объем шифруемых данных.

Недостатком шифрования с  закрытым ключом является необходимость того, чтобы две стороны согласовали ключ и  вектор инициализации, для чего может потребоваться их передача через систему связи.

Шифрование с  закрытым ключом часто используется в сочетании с шифрованием с открытым ключом для безопасной передачи ключа и вектора инициализации.

***Криптографический протокол***  — это такая процедура взаимодействия двух или более абонентов с  использованием криптографических средств, в  результате которой абоненты достигают своей цели, а их противники — не достигают.

Типы протоколов можно условно подразделить на две группы: ***прикладные протоколы и примитивные***.

* ***Прикладной протокол*** решает конкретную задачу, которая возникает (или может возникнуть) на практике.
* ***Примитивные протоколы*** используются как своеобразные «строительные блоки» при разработке прикладных протоколов.

1**. Протоколы конфиденциальной передачи сообщений**. Задача конфиденциальной передачи сообщений состоит в  следующем. Имеются два участника протокола, которые являются абонентами сети связи. Участники соединены некоторой линией связи, по которой можно пересылать сообщения в обе стороны.

Линию связи может контролировать противник. У одного из абонентов имеется конфиденциальное сообщение m, и  задача состоит в  том, чтобы это сообщение конфиденциальным же образом передать второму абоненту.

*Протоколы этого типа, наверное, появились раньше других криптографических протоколов, так как задача конфиденциальной передачи сообщений — исторически первая задача, которая решалась криптографией.*

***2. Протоколы аутентификации и идентификации***.

Они предназначены для предотвращения доступа к  некоторой информации лиц, не являющихся ее пользователями, а также предотвращения доступа пользователей к тем ресурсам, на которые у них нет полномочий.

Сфера применения — организация доступа пользователей к ресурсам некоторой большой информационной системы.

3. ***Протоколы распределения ключей***. Необходимы для обеспечения секретными ключами участников обмена зашифрованными сообщениями.

4. ***Протоколы электронной цифровой подписи***. Позволяют ставить под электронными документами подпись, аналогичную обыкновенной подписи на бумажных документах. В результате выполнения протокола электронной цифровой подписи к передаваемой информации добавляется уникальное числовое дополнение, позволяющее проверить ее авторство.

5***. Протоколы обеспечения неотслеживаемости*** (электронные деньги). Под электронными деньгами в  криптографии понимают электронные платежные средства, обеспечивающие неотслеживаемость, т. е. невозможность проследить источник пересылки информации

Этот протокол достаточно прост, однако он может действительно использоваться на практике.

Примеры средств и протоколов шифрования сообщений:

AES (Advanced Encryption Standard) — самый популярный алгоритм шифрования, используется в различных проектах и организациях, таких как правительство США, Google Cloud, Snapchat и Facebook Messenger.

RSA (Rivest–Shamir–Adleman) — асимметричный алгоритм шифрования, используется для шифрования данных на уровне файла, в электронных платежах, онлайн-банкинге, инфраструктуре открытых ключей (PKI) и системах аутентификации.

SHA-256 (Secure Hash Algorithm 256-bit) — алгоритм хеширования, разработанный Агентством национальной безопасности США, используется на сайтах для хранения паролей, в SSL/TLS-сертификатах, блокчейнах и антивирусных программах.