## Прикладное программное обеспечение и его топология.

Программа*(program, routine) —*упорядоченная последовательность команд (инструкций) компьютера для решения задачи.

Программное обеспечение*(sowtware) —*совокупность программ обработки данных и необходимых для их эксплуатации документов.

Программные продукты можно классифицировать по различным признакам:

1. системное программное обеспечение, выполняющее различные функции, например:

управление ресурсами компьютера; создание копий используемой информации; проверка работоспособности устройств компьютера; выдача справочной информации о компьютере и др.;

2. прикладные программы, непосредственно обеспечивающие выполнение необходимых пользователям работ;

3. инструментарий технологии программирования, облегчающий процесс создания новых программ для компьютера.

К прикладным программам относятся программные комплексы, обеспечивающие выполнение различных прикладных задач, т. е. выполнение фактических задач пользователей. К типовому прикладному ПО относят следующие программы:

1)текстовые редакторы; 2)табличные процессоры; 3)системы иллюстративной и деловой графики (графические процессоры); 4)системы управления базами данных; 5)экспертные системы; программы математических расчетов, 6)моделирования и анализа экспериментальных данных.

***Те́кстовый реда́ктор***— компьютерная программа, предназначенная для создания и изменения текстовых файлов, а также их просмотра на экране, вывода на печать, поиска фрагментов текста и т. п.

Условно выделяют два типа редакторов: текстовый редактор и текстовый процессор.

Первый тип ориентирован на работу с последовательностью символов в текстовых файлах. Такие редакторы обеспечивают расширенную функциональность — подсветку синтаксиса, сортировку строк, шаблоны, конвертацию кодировок, показ кодов символов и т. п. Иногда их называют **редакторы кода**, так как основное их предназначение — написание исходных кодов компьютерных программ. Примером может являться также редакторWindows– Блокнот.

Второй тип текстовых редакторов имеет расширенные функции форматирования текста, внедрения в него графики и формул, таблиц и объектов. Такие редакторы часто **называют** ***текстовыми процессорами*** и предназначены они для создания различного рода документов, от личных писем до официальных бумаг.

Возможности текстовых редакторов различны — от программ, предназначенных для подготовки небольших документов простой структуры, до программ для набора, оформления и полной подготовки к типографскому изданию книг и журналов (издательские системы).

Наиболее известный текстовый процессор — Microsoft Word. Полнофункциональные издательские системы — Microsoft Publisher, Corel Ventura и Adobe PageMaker.

***Табличный процессор***— это комплекс взаимосвязанных программ, предназначенный для обработки электронных таблиц.***Электронная таблица***— это компьютерный эквивалент обычной таблицы, состоящей из строк и граф, на пересечении которых располагаются клетки, в которых содержится числовая информация, формулы или текст.

Табличные процессоры представляют собой удобное средство для проведения бухгалтерских и статистических расчетов. В каждом пакете имеются сотни встроенных математических функций и алгоритмов статистической обработки данных. Кроме того, имеются мощные средства для связи таблиц между собой, создания и редактирования электронных баз данных. Специальные средства позволяют автоматически получать и распечатывать настраиваемые отчеты с использованием десятков различных типов таблиц, графиков, диаграмм, снабжать их комментариями и графическими иллюстрациями.

Табличные процессоры имеют встроенную справочную систему, предоставляющую пользователю информацию по конкретным командам меню и другие справочные данные. Многомерные таблицы позволяют быстро делать выборки в базе данных по любому критерию.

Самые популярные табличные процессоры — Microsoft Excel (Эксель) и Lotus 1-2-3.

**Графический редактор**— программа (или пакет программ), позволяющая создавать и редактировать двумерные изображения с помощью компьютера.

*Растровые графические редакторы.*Наиболее популярны: Adobe Photoshop для операционных систем Microsoft Windows и Mac OS X.

*Векторные графические редакторы.*Наиболее популярны:AdobeIllustrator,CorelDraw,MacromediaFreeHand — для операционных систем Microsoft Windows и Mac OS X.

*Гибридные графические редакторы.*Наиболее популярны: RasterDesk для AutoCAD, Spotlight для операционных систем Microsoft Windows

**Системы управления базами данных.**

***База данных***— это один или несколько файлов данных, предназначенных для хранения, изменения и обработки больших объемов взаимосвязанной информации. В базе данных предприятия, например, может храниться: вся информация о штатном расписании, о рабочих и служащих предприятия; сведения о материальных ценностях; данные о поступлении сырья и комплектующих; сведения о запасах на складах; данные о выпуске готовой продукции; приказы и распоряжения дирекции и т.п. Даже небольшие изменения какой-либо информации могут приводить к значительным изменениям в разных других местах.

***Пример.***Издание приказа о повышении в должности одного работника приводит к изменениям не только в личном деле работника, но и к изменениям в списках подразделения, в котором он работает, в ведомостях на зарплату, в графике отпусков и т.п.

Базы данных используются под управлением ***систем управления базами данных (СУБД)***.

**Система управления базами данных (СУБД*)***— это система программного обеспечения, позволяющая обрабатывать обращения к базе данных, поступающие от прикладных программ конечных пользователей. Системы управления базами данных позволяют:

объединять большие объемы информации и обрабатывать их, сортировать, делать выборки по определённым критериям и т.п.

включать в них не только текстовую и графическую информацию, но и звуковые фрагменты и даже видеоклипы.

создавать новые базы данных, не прибегая к программированию, а пользуясь только встроенными функциями.

СУБД обеспечивают правильность, полноту и непротиворечивость данных, а также удобный доступ к ним. Популярные СУБД — FoxPro,AccessforWindows,Paradox.

**Экспе́ртные систе́мы –** это компьютерные программы , способные частично заменить специалиста-эксперта в разрешении проблемной ситуации. В информатике экспертные системы рассматриваются совместно с базами знаний как модели поведения экспертов в определенной области знаний с использованием процедур логического вывода и принятия решений, а базы знаний — как совокупность фактов и правил логического вывода в выбранной предметной области деятельности.

**Пакет прикладных программ**(applicationprogrampackage) — комплекс взаимосвязанных программ для решения задач определенного класса конкретной предметной области. Каждый ППП обладает обычно рядом возможностей по методам обработки данных и формам их представления, полноте диагностики, что дает возможность пользователю выбрать подходящий для конкретных условий вариант. ППП обеспечивают значительное снижение требований к уровню профессиональной подготовки пользователей в области программирования, вплоть до возможности эксплуатации пакета без программиста.

***Интегрированные пакеты***представляют собой набор нескольких программных продуктов, объединенных в единый удобный инструмент. Наиболее развитые из них включают в себя текстовый редактор, органайзер, электронную таблицу, СУБД, средства поддержки электронной почты, программу создания презентационной графики. Результаты, полученные отдельными подпрограммами, могут быть объединены в окончательный документ, содержащий табличный, графический и текстовый материал. Интегрированные пакеты, как правило, содержат некоторое ядро, обеспечивающее возможность тесного взаимодействия между составляющими. Наиболее известные интегрированные пакеты: Microsoft Office. В этот мощный профессиональный пакет вошли такие необходимые программы, как текстовый редактор WinWord , электронная таблица Excel, программа создания презентаций PowerPoint, СУБД Access, средство поддержки электронной почты Mail. Мало того, все части этого пакета составляют единое целое.

## Этапы и последовательность разработки программ, понятие приложения.

***Инструментарий технологии программирования***обеспечивает процесс разработки программ и включает специализированные программные продукты, которые являются инструментальными средствами разработчика. Программные продукты данного класса поддерживают все технологические этапы процесса проектирования, программирования (кодирования), отладки и тестирования создаваемых программ. Пользователями технологии программирования являются системные и прикладные программисты.

Но прежде чем обсуждать сами инструментальные средства разработки программ, рассмотрим принципы разработки программного обеспечения.

При разработке программ всегда следует помнить одно золотое правило: надежность программы достигается, в первую очередь, благодаря ее правильному проектированию, а не бесконечному тестированию. О том, как правильно организовать разработку программ (независимо от языка), написана не одна сотня книг. Большинство авторов предлагают следующий***процесс разработки (создания) программы***:

**1. Этап постановки и формулировки задачи.** **Постановка задачи***(problem definition) —*это точная формулировка решения задачи на компьютере с описанием входной и выходной информации:

изучение предметной области и сбор материала в проблемно-ориентированном контексте;

определение назначения программы, выработка требований к ней и представление требований, если возможно, в формализованном виде;

формулирование требований к представлению исходных данных и выходных результатов;

определение структур входных и выходных данных;

формирование ограничений и допущений на исходные и выходные данные.

**2. Этап проектирования:**

формирование модели задачи;

выбор метода реализации задачи;

разработка алгоритма реализации задачи;

разработка структуры программы

**3. Этап кодирования:**

уточнение структуры входных и выходных данных и определение формата их представления;

программирование задачи;

комментирование текста программы и составление предварительного описания программы.

**4. Этап отладки и тестирования:**

составление тестов для проверки правильности работы программы;

обнаружение, локализация и устранение ошибок в программе, выявленных в тестах;

корректировка кода программы и ее описания.

**5. Этап эксплуатации и сопровождения:**

настройка программы на конкретные условия использования;

обучение пользователей работе с программой;

организация сбора сведений о сбоях в работе программы, ошибках в выходных данных, пожеланиях по улучшению интерфейса и удобства работы с программой;

модификация программы с целью устранения выявленных ошибок и, при необходимости, изменения ее функциональных возможностей.

После написания программы на языке программирования нужно ввести программу в компьютер, перевести в машинное представление и выполнить. Как это сделать? Дальнейшее обсуждение будет посвящено именно этому вопросу.

Программа, подготовленная на языке высокого уровня, проходит несколько этапов:

**1. этап**. В текстовом редакторе пишется***исходный код***программы на алгоритмическом языке (source code) и сохраняется в файле с расширением \*.pas.

**2 этап*.*****Трансляция**, происходит преобразование***исходного кода***программы (source code) в***объектный код***(object code), т.е. происходит проверка синтаксиса написания операторов, и если ошибок в написании нет, осуществляется перевод на язык машинных кодов. Файл объектного кода имеет расширение \*.obj;

***Трансляторы***предназначены для проверки правильности написания операторов и преобразования программ, написанных на языках программирования, в программы на машинном языке. Программа, подготовленная на каком-либо языке программирования, называется***исходным модулем***. В качестве входной информации трансляторы применяют исходные модули и формируют в результате своей работы***объектные модули***, являющиеся входной информацией для редактора связей. Объектный модуль содержит текст программы на машинном языке и дополнительную информацию, обеспечивающую настройку модуля по месту его загрузки и объединение этого модуля с другими независимо оттранслированными модулями в единую программу.

**Трансляторы**делятся на два класса:***компиляторы***(compiler)и***интерпретаторы***(interpreter). Компиляторы транслируют **всю**программу, но без ее выполнения. Интерпретаторы, в отличие от компиляторов, выполняют**пооператорный** перевод на машинный язык и выполнение всей программы.

**3. этап**.***Компоновка****,*когда происходит обработка объектного кода***редактором связей***, специальной программой осуществляющей построение***загрузочного модуля***(load module), пригодного к выполнению (рис 16.).

***Компоновщик, или редактор связей***- системная обрабатывающая программа, редактирующая и объединяющая объектные (ранее оттраслированные) модули в единые загрузочные, готовые к выполнению программные модули. Загрузочный модуль может быть помещен ОС в основную память и выполнен.

Получив исполняемый модуль, не спешите радоваться. К сожалению, устранение синтаксических ошибок еще не гарантирует того, что программа будет хотя бы запускаться, не говоря уже о правильности работы. Поэтому обязательным этапом процесса разработки является ***отладка***. На этапе***отладки***, используя описание алгоритма, выполняется контроль правильности функционирования, как отдельных участков кода, так и всей программы в целом.

**Отладчик**позволяет управлять процессом исполнения программы, является инструментом для поиска и исправления ошибок в программе. Базовый набор функций отладчика включает:

пошаговое выполнение программы (режим трассировки) с отображением результатов,

остановка в заранее определенных точках,

возможность остановки в некотором месте программы при выполнении некоторого условия;

изображение и изменение значений переменных.

Но даже успешное окончание отладки еще не является гарантией того, что программа будет работать правильно со всеми возможными исходными данными. Поэтому нужно обязательно провести тестирование программы, то есть проверить ее работу на «пограничных» и заведомо некорректных исходных данных. Для этого составляются тесты. Вполне возможно, что результаты тестирования вас не удовлетворят. В этом случае придется вносить поправки в код программы, то есть возвращаться к первому шагу процесса разработки (см. рисунок 16).

**Инструментарий технологии программирования**— совокупность программ и программных комплексов, обеспечивающих технологию разработки, отладки и внедрения создаваемых программных продуктов.

Сформировались следующие группы программных продуктов (рис.17):

**Средства для создания приложений***,*включающие:

* *локальные средства*, обеспечивающие выполнение отдельных работ по созданию программ;
* *интегрированные среды*разработчиков программ, обеспечивающие выполнение комплекса взаимосвязанных работ по созданию программ;

**САSE-технология (Computer-Aided** **System** **Engineering),**представляющая методы анализа, проектирования и создания программных систем и предназначенная для автоматизации процессов разработки и реализации информационных систем. Подробнее вопрос 46!!!

**Современные технологии разработки программного обеспечения**

Microsoft Solutions Framework (MSF) — это методология ведения проектов и разработки решений, базирующаяся на принципах работы над продуктами самой фирмы Microsoft и предназначенная для использования в организациях, нуждающихся в концептуальной схеме для построения современных решений.

Microsoft Solutions Framework является схемой для принятия  решений по планированию и  реализации новых технологий в организациях. MSF включает обучение, информацию, рекомендации и инструменты для идентификации и структуризации информационных потоков бизнес-процессов и всей информационной инфраструктуры новых технологий.

Microsoft Solutions Framework представляет собой хорошо сбалансированный набор методик организации процесса разработки, который может быть адаптирован под потребности практически любого коллектива разработчиков. MSF содержит не только рекомендации общего характера, но и предлагает адаптируемую модель коллектива разработчиков, определяющую взаимоотношения внутри коллектива, гибкую модель проектного планирования, основанного на управлении проектными группами, а также набор методик для оценки рисков.

MSF состоит из  двух моделей:

* модель проектной группы;
* модель процессов,

и трех дисциплин:

* управление проектами;
* управление рисками;
* управление подготовкой.

В MSF нет роли «менеджер проекта» и иерархии руководства, управление разработкой распределено между руководителями отдельных  проектных групп внутри коллектива, выполняющих следующие задачи:

* Управление программой
* Разработка
* Тестирование
* Управление выпуском
* Удовлетворение потребителя
* Управление продуктом

При управлении проектом четко ставится цель, которую  необходимо достичь в результате, и учитываются ограничения, накладываемые на проект. Все виды ограничений могут быть отнесены к одному из трех видов: ограничения ресурсов, ограничения времени и ограничения возможностей. Эти три вида ограничений и приоритетность задач по их преодолению образуют треугольник приоритетов в MSF.Треугольник приоритетов является основой для матрицы компромиссов — заранее утвержденных представлений о том, какие аспекты процесса разработки будут четко заданы, а какие будут согласовываться или приниматься как есть.

Microsoft выпустила  среду разработки, в полной мере  поддерживающей основные идеи MSF — Microsoft Visual Studio 2005 Team Edition. Это первый  программный комплекс, представляющий  собой не среду разработки  для индивидуальных членов коллектива, а комплексное средство поддержки коллективной работы.

В состав Visual Studio Team Edition входит специальная редакция для тестировщиков — Team Edition for Software Testers. Материалы семинарских занятий  по данному курсу ориентированы  на эту среду разработки, в то время как лекционные материалы ориентированы на изложение общих принципов и методик тестирования.

Rational Unified Process — это методология создания программного обеспечения, оформленная в виде размещаемой на Web базы знаний, которая снабжена поисковой системой.

Продукт Rational Unified Process (RUP) разработан и поддерживается Rational Software. Он регулярно обновляется с целью учета передового опыта и улучшается за счет проверенных на практике результатов.

RUP обеспечивает  строгий подход к распределению  задач и ответственности внутри организации-разработчика. Его предназначение заключается в том, чтобы гарантировать создание точно в срок и в рамках установленного бюджета качественного ПО, отвечающего нуждам конечных пользователей.

RUP способствует  повышению производительности коллективной разработки и предоставляет лучшее из накопленного опыта по созданию ПО, посредством руководств, шаблонов и наставлений по пользованию инструментальными средствами для всех критически важных работ, в течение жизненного цикла создания и сопровождения ПО. Обеспечивая каждому члену группы доступ к той же самой базе знаний, вне зависимости от того, разрабатывает ли он требования, проектирует, выполняет тестирование или управляет проектом — RUP гарантирует, что все члены группы используют общий язык моделирования и процесс, имеют согласованное видение того, как создавать ПО. В качестве языка моделирования в общей базе знаний используется Unified Modeling Language (UML), являющийся международным стандартом.

Особенностью RUP является то, что в результате работы над проектом создаются и совершенствуются модели. Вместо создания громадного количества бумажных документов, RUP опирается на разработку и развитие семантически обогащенных моделей, всесторонне представляющих разрабатываемую систему. RUP — это руководство по тому, как эффективно использовать UML. Стандартный язык моделирования, используемый всеми членами группы, делает понятными для всех описания требований, проектирование и архитектуру системы.

RUP поддерживается  инструментальными средствами, которые  автоматизируют многие элементы  процесса разработки. Они используются  для создания и совершенствования  различных промежуточных продуктов  на различных этапах процесса  создания ПО, например, при визуальном моделировании, программировании, тестировании и т.д.

RUP — это конфигурируемый  процесс, поскольку вполне понятно,  что невозможно создать единого  руководства на все случаи  разработки ПО. RUP пригоден как  для маленьких групп разработчиков,  так и для больших организаций, занимающихся созданием ПО. В основе RUP лежит простая и понятная архитектура процесса, которая обеспечивает общность для целого семейства процессов. Более того, RUP может конфигурироваться для учета различных ситуаций. В его состав входит Development Kit, который обеспечивает поддержку процесса конфигурирования под нужды конкретных организаций.

RUP описывает,  как эффективно применять коммерчески  обоснованные и практически опробованные  подходы к разработке ПО для  коллективов разработчиков, где каждый из членов получает преимущества от использования передового опыта в:

* итерационной разработке ПО;
* управлении требованиями;
* использовании компонентной архитектуры;
* визуальном моделировании;
* тестировании качества ПО;
* контроле за изменениями в ПО.

RUP организует работу над проектом в терминах последовательности действий  продуктов деятельности, исполнителей и других статических аспектов процесса, с одной стороны, и в терминах циклов, фаз, итераций и временных отметок завершения определенных этапов в создании ПО, т.е. в терминах динамических аспектов процесса — с другой.

Экстремальное программирование — сравнительно молодая методология разработки программных систем, основанная на постепенном улучшении системы и разработки ее очень короткими итерациями. По своей сути экстремальное программирование (XP) — это одна из так называемых «гибких» методологий разработки ПО, которая представляет собой небольшой набор конкретных правил, позволяющих максимально эффективно выполнять требования современной теории управления программными проектами.

XP ориентирована  на:

* командную работу с тесными связями внутри команды и с заказчиком;
* разработку наиболее простых работающих решений;
* гибкое адаптивное планирование;
* оперативную обратную связь (путем модульного и функционального тестирования).

Основными принципами XP является разработка небольшими итерациями на основании порции требований заказчика (т.н. пользовательских историй), написание  функциональных тестов до написания  программного кода, постоянное общение  и постоянный рефакторинг кода.

Основными практиками XP являются:

* Планирование процесса
* Частые релизы
* Метафора системы
* Простая архитектура
* Тестирование
* Рефакторинг
* Парное программирование
* Коллективное владение кодом
* Частая интеграция
* 40-часовая рабочая неделя
* Стандарты кодирования
* Тесное взаимодействие с заказчиком

**Сравнение технологий MSF, RUP и XP**

Сопровождение разработки системы и самой системы  регламентируется методологией RUP, однако данная технология достаточно сильно ориентирована на внутрифирменные  инструментальные средства.

Extreme Programming хорошо  подходит для проектных групп  малого размера и для небольших  систем с часто изменяемыми  требованиями. Основная проблема XP — сопровождаемость. В случае  текучки кадров в коллективе  разработчиков значительная часть  проектной информации может быть утеряна из-за практически отсутствующей документации.

Microsoft Solutions Framework является наиболее сбалансированной  технологией, ориентированной на  проектные группы малых и средних  размеров. MSF не накладывает никаких  ограничений на используемый инструментарий и содержит рекомендации весьма общего характера. Однако, эти рекомендации могут быть использованы для построения конкретного процесса, соответствующего потребностям коллектива разработчиков.

**CASE-технологии**

Давно канули в Лету те времена, когда один человек вполне мог справляться с реализацией программного проекта, обеспечивающего функциональность крупных предприятий. Постоянный рост сложности и комплексности не только целей проекта, но и инструментария их реализации приводит к тому, что уже трудно обойтись силами отдельных специалистов, а требуется слаженная работа целой команды.

Для того чтобы  успешно выполнить проект, объект проектирования должен быть прежде всего  правильно и адекватно описан, то есть необходимо построить полноценные и функциональные информационные модели объекта проектирования. До недавнего времени проектирование информационных систем выполнялось главным образом на интуитивном уровне с применением не формализованных методов, которые основывались на практическом опыте, экспертных оценках и дорогостоящих экспериментальных проверках качества функционирования подобных систем. Но, естественно, во время разработки и функционирования информационных систем потребности пользователей могут изменяться или уточняться, что еще более усложняет разработку и сопровождение.

В 1970-80-х годах  при разработке информационных систем широко применялась структурная  методология, предоставляющая в  распоряжение разработчиков строгие  формализованные методы описания ИС и принимаемых технических решений. Эта методология основывалась на наглядной графической технике, иначе говоря, для описания проекта использовались различного рода схемы и диаграммы. Наглядность и строгость средств структурного анализа позволяла разработчикам и будущим пользователям системы с самого начала неформально участвовать в ее создании, обсуждать и закреплять понимание основных технических решений. Однако широкое применение этой методологии и следование ее рекомендациям при разработке конкретных проектов встречалось достаточно редко, поскольку ее практически невозможно реализовать на должном уровне ручным, неавтоматизированным, способом. Вручную очень трудно разработать и графически представить строгие формальные спецификации системы, проверить их на полноту и непротиворечивость, и тем более изменить. Если все же удается создать строгую систему проектных документов, то ее переработка при появлении серьезных изменений практически неосуществима. Если участники проекта пытались прибегнуть к ручной разработке, то перед ними возникали следующие проблемы:

* неадекватная спецификация требований;
* неспособность обнаруживать ошибки в проектных решениях;
* низкое качество документации, снижающее эксплуатационные качества;
* затяжной цикл и неудовлетворительные результаты тестирования.

Как ни странно, все эти аргументы не всегда доходили до разработчиков. Ведь прекрасно известно, что проектировщики информационных систем в самую последнюю очередь используют компьютерные технологии для повышения качества и производительности своей работы.

Но рано или  поздно должны были появиться специализированные программно-технологические средства для разработки проектов, в частности, основанных на информатизации. Ими стали средства, реализующие CASE-технологию создания и сопровождения информационных систем. Термин CASE (Computer-Aided Software Engineering) сегодня понимается достаточно широко.

Первоначальное  значение термина, ограниченное вопросами  автоматизации разработки программного обеспечения (ПО), в настоящее время  приобрело новый смысл, и теперь это понятие охватывает процесс  разработки сложных информационных систем в целом. Теперь под термином CASE-средства понимаются программные средства, поддерживающие процессы создания и сопровождения подобных систем, включая анализ и формулировку требований, проектирование прикладного ПО (приложений) и баз данных, генерацию кода, тестирование, документирование, обеспечение качества, конфигурационное управление и управление проектом и т. д. CASE-средства вместе с системным ПО и техническими средствами образуют полную среду разработки.

Активные исследования в области методологии программирования привели к тому, что программирование приобрело черты системного подхода с разработкой и внедрением языков высокого уровня, методов структурного и модульного программирования, языков проектирования и средств их поддержки, формальных и неформальных языков описаний системных требований и спецификаций и т. д. Кроме того, появлению CASE-технологии способствовали и такие факторы, как:

* подготовка аналитиков и программистов, восприимчивых к концепциям модульного и структурного программирования;
* широкое внедрение и постоянный рост производительности компьютеров, позволившие использовать эффективные графические средства и автоматизировать большинство этапов проектирования;
* внедрение сетевой технологии, предоставившей возможность объединения усилий отдельных исполнителей в единый процесс проектирования путем использования разделяемой базы данных, содержащей необходимую информацию о проекте.

Таким образом, CASE-технология представляет собой методологию  проектирования ИС, а также набор инструментальных средств, позволяющих в наглядной форме моделировать предметную область, анализировать эту модель на всех этапах разработки и сопровождения ИС и разрабатывать приложения в соответствии с потребностями пользователей. Большая часть CASE-средств использует методологию структурного (в основном) или ориентированного анализа и проектирования, использующих спецификации в виде диаграмм или текстов для описания внешних требований, связей между моделями системы, динамики поведения системы и архитектуры программных средств.

Согласно западным исследованиям CASE-технология попала в  разряд наиболее стабильных информационных технологий. Впрочем, CASE-средства, как  и любой инструмент, нужно уметь применять. Существует множество примеров их неудачного внедрения, в результате которых CASE-средства становятся «полочным» ПО (shelfware). В связи с этим необходимо отметить следующее:

* CASE-средства не обязательно дают немедленный эффект; он может быть получен только спустя какое-то время;
* реальные затраты на внедрение CASE-средств обычно намного превышают затраты на их приобретение;
* CASE-средства обеспечивают возможности для получения существенной выгоды только после успешного завершения процесса их внедрения.

Современные CASE-средства охватывают обширную область поддержки многочисленных технологий проектирования информационных систем — от простых средств анализа и документирования до полномасштабных средств автоматизации, покрывающих весь жизненный цикл ПО.

Наиболее трудоемкими  этапами разработки информационных систем являются анализ и проектирование, в процессе которых CASE-средства обеспечивают качество принимаемых технических решений и подготовку проектной документации. При этом большую роль играют методы визуального представления информации. Это предполагает построение структурных или иных диаграмм в реальном масштабе времени, использование многообразной цветовой палитры, сквозную проверку синтаксических правил. Графические средства моделирования позволяют разработчикам в наглядном виде изучать существующую информационную систему, перестраивать ее в соответствии с поставленными целями и имеющимися ограничениями.

В разряд CASE-средств  попадают как относительно дешевые  системы для персональных компьютеров  с ограниченными возможностями, так и дорогостоящие системы для неоднородных вычислительных платформ и операционных сред. Так, современный рынок программных средств насчитывает около 300 различных CASE-средств, наиболее мощные из которых используются практически всеми ведущими западными компаниями.

Обычно к CASE-средствам относят любое программное средство, автоматизирующее ту или иную совокупность процессов жизненного цикла ПО и обладающее следующими особенностями:

* мощные графические средства для описания и документирования ИС, обеспечивающие удобный интерфейс с разработчиком и развивающие его творческие возможности;
* интеграция отдельных компонент CASE-средств, обеспечивающая управляемость процессом разработки информационной системы;
* использование специальным образом организованного хранилища проектных метаданных (репозитория). Интегрированное CASE-средство (или комплекс средств, поддерживающих полный жизненный цикл ПО) содержит следующие компоненты:
* репозиторий, являющийся основой CASE-средства. Он должен обеспечивать хранение версий проекта и его отдельных компонентов, синхронизацию поступления информации от различных разработчиков при групповой разработке, контроль метаданных на полноту и непротиворечивость;
* графические средства анализа и проектирования, обеспечивающие создание и редактирование иерархически связанных диаграмм (DFD, ERD и др.), образующих модели информационной системы;
* средства разработки приложений, включая языки 4GL и генераторы кодов;
* средства конфигурационного управления;
* средства документирования;
* средства тестирования;
* средства управления проектом;
* средства реинжиниринга.

Все современные CASE-средства можно классифицировать по типам и категориям. Классификация  по типам отражает функциональную ориентацию CASE-средств на те или иные процессы жизненного цикла. Классификация по категориям определяет степень интегрированности по выполняемым функциям и включает отдельные локальные средства, решающие небольшие автономные задачи (tools), набор частично интегрированных средств, охватывающих большинство этапов жизненного цикла информационных систем (toolkit) и полностью интегрированные средства, поддерживающие весь жизненный цикл информационных систем и связанные общим репозиторием. Помимо этого CASE-средства можно классифицировать по применяемым методологиям и моделям систем и БД; степени интегрированности с СУБД; доступным платформам.

Классификация по типам в основном совпадает  с компонентным составом CASE-средств  и включает:

* средства анализа (Upper CASE), предназначенные для построения и анализа моделей предметной области (Design/IDEF (Meta Software), BPwin (Logic Works));
* средства анализа и проектирования (Middle CASE), поддерживающие наиболее распространенные методологии проектирования и использующиеся для создания проектных спецификаций (Vantage Team Builder (Cayenne), Designer/2000 (ORACLE), Silverrun (CSA), PRO-IV (McDonnell Douglas), CASE.Аналитик (МакроПроджект)). Выходом таких средств являются спецификации компонентов и интерфейсов системы, архитектуры системы, алгоритмов и структур данных;
* средства проектирования баз данных, обеспечивающие моделирование данных и генерацию схем баз данных (как правило, на языке SQL) для наиболее распространенных СУБД. К ним относятся ERwin (Logic Works), S-Designor (SDP) и DataBase Designer (ORACLE).Средства проектирования баз данных имеются также в составе CASE-средств Vantage Team Builder, Designer/2000, Silverrun и PRO-IV;
* средства разработки приложений. К ним относятся средства 4GL (Uniface (Compuware), JAM (JYACC), PowerBuilder (Sybase), Developer/2000 (ORACLE), New Era (Informix), SQL Windows (Gupta), Delphi (Borland) и др.) и генераторы кодов, входящие в состав Vantage Team Builder, PRO-IV и частично — в Silverrun;
* средства реинжиниринга, обеспечивающие анализ программных кодов и схем баз данных и формирование на их основе различных моделей и проектных спецификаций. Средства анализа схем БД и формирования ERD входят в состав Vantage Team Builder, PRO-IV, Silverrun, Designer/2000, ERwin и S-Designor. В области анализа программных кодов наибольшее распространение получают объектно-ориентированные CASE-средства, обеспечивающие реинжиниринг программ на языке С++ (Rational Rose (Rational Software), Object Team (Cayenne)).

Вспомогательные типы включают:

* средства планирования и управления проектом (SE Companion, Microsoft Project и др.);
* средства конфигурационного управления (PVCS (Intersolv));
* средства тестирования (Quality Works (Segue Software));
* средства документирования (SoDA (Rational Software)).

Ввиду разнообразной  природы CASE-средств было бы ошибочно делать какие-либо утверждения относительно реального удовлетворения тех или иных ожиданий от их внедрения. Можно перечислить следующие факторы, усложняющие определение возможного эффекта от использования CASE-средств:

* широкое разнообразие качества и возможностей CASE-средств;
* относительно небольшое время использования CASE-средств в различных организациях и недостаток опыта их применения;
* широкое разнообразие в практике внедрения различных организаций;
* отсутствие детальных метрик и данных для уже выполненных и текущих проектов;
* широкий диапазон предметных областей проектов;
* различная степень интеграции CASE-средств в различных проектах.

Одни полагают, будто реальная выгода от использования  некоторых типов CASE-средств может  быть получена только после одно- или  двухлетнего опыта. Другие считают, что воздействие может реально проявиться в фазе эксплуатации жизненного цикла информационных систем, когда технологические улучшения могут привести к снижению эксплуатационных затрат.   
Для успешного внедрения CASE-средств организация должна обладать следующими качествами:

* Технология. Понимание ограниченности существующих возможностей и способность принять новую технологию;
* Культура. Готовность к внедрению новых процессов и взаимоотношений между разработчиками и пользователями;
* Управление. Четкое руководство и организованность по отношению к наиболее важным этапам и процессам внедрения.

Если организация  не обладает хотя бы одним из перечисленных  качеств, то внедрение CASE-средств может  закончиться неудачей, независимо от степени тщательности следования различным  рекомендациям по внедрению.

Для того чтобы  принять взвешенное решение относительно инвестиций в CASE-технологию, пользователи вынуждены производить оценку отдельных CASE-средств, опираясь на неполные и  противоречивые данные. Эта проблема зачастую усугубляется недостаточным знанием всех возможных «подводных камней» использования CASE-средств. Среди наиболее важных проблем выделяют следующие:

• достоверная  оценка отдачи от инвестиций в CASE-средства затруднительна ввиду отсутствия приемлемых метрик и данных по проектам и процессам разработки ПО;   
• внедрение CASE-средств может представлять длительный процесс и не принести немедленной отдачи. Возможно даже краткосрочное снижение продуктивности в результате усилий, затрачиваемых на внедрение. Вследствие этого руководство организации-пользователя может утратить интерес к CASE-средствам и прекратить поддержку их внедрения;   
• отсутствие полного соответствия между теми процессами и методами, которые поддерживаются CASE-средствами, и теми, что используются в данной организации, может привести к дополнительным трудностям;   
• CASE-средства зачастую трудно использовать в комплексе с другими подобными средствами. Это объясняется как различными парадигмами, поддерживаемыми разнообразными средствами, так и проблемами передачи данных и управления от одного средства к другому;   
• некоторые CASE-средства требуют слишком много усилий для того, чтобы оправдать их использование в небольшом проекте, тем не менее, можно извлечь выгоду из той дисциплины, к которой обязывает их применение;   
• негативное отношение персонала к внедрению новой CASE-технологии может быть главной причиной провала проекта.

Пользователи CASE-средств  должны быть готовы к необходимости  долгосрочных затрат на эксплуатацию, частому появлению новых версий и возможному быстрому моральному старению средств, а также постоянным затратам на обучение и повышение квалификации персонала.

Но все же грамотное, продуманное и обоснованное использование CASE-технологии способно принести следующие выгоды:

* высокий уровень технологической поддержки процессов разработки и сопровождения ПО;
* положительное воздействие на некоторые или все из перечисленных факторов: производительность, качество продукции, соблюдение стандартов, документирование;
* приемлемый уровень отдачи от инвестиций в CASE-средства.

**Тестирование программ**

1. Пошаговое и монолитное тестирование.

Монолитное  тестирование заключается в том, что сначала по отдельности тестируется  каждый модуль, а затем все они  комбинируются в рабочую программу и она тестируется.

Пошаговый метод  состоит в том, что каждый модуль для тестирования подключается к  набору ранее отсортированных модулей.

Рассмотрим  пример:

При монолитном подходе сначала тестируются  модули, каждый независимо от других, затем они собираются в программу. Для тестирования каждого модуля требуется специальный модуль — драйвер и 1 или несколько модулей — заглушек.

Модуль —  драйвер, который содержит фиксированные  тестовые данные, вызывает тестируемый  модуль и отображает выходные результаты.

Заглушка - программа, имитирующая работу модуля нижнего уровня. Она может не содержать ничего, кроме сообщения о том, что произошел вход в этот модуль, и возврата управления.

Выводы:

-      Монолитное тестирование требует больших затрат труда.

-       При пошаговом тестировании раньше обнаруживаются ошибки в межмодульных связях.

-     При пошаговом тестировании ошибки в межмодульных связях обнаруживаются легче.

-     Монолитный способ применяется чтобы ускорить сроки сдачи программы.

-     При монолитном  - меньше расход машинного времени.

Категории тестов системных испытаний.

1.     Тестирование удобства использования. Сравниваются цели с содержанием пользовательской документации.

2.      Тестирование на предельных объемах.

3.      Тестирование на предельных нагрузках. Означает поступление пикового объема данных в течение короткого интервала времени.

4.     Тестирование удобства эксплуатации:

-        Справка

-       Значимость входных сообщений программы

-       Понятна ли диагностика ошибок

-      Единообразие стиля пользовательских интерфейсов

-        Содержит ли система опции, число которых чрезмерно или использование которых маловероятно

-       Выдает ли система какие-либо подтверждения на все входные сообщения

5.      Тестирование защиты (от несанкционированного доступа).

6.      Тестирование производительности.

7.      Тестирование требований к памяти.

8.      Тестирование конфигураций оборудования.

9.      Тестирование удобства установки (настройки, инсталляции).

10.  Тестирование надежности.

11.  Тестирование восстановления.

12.  Тестирование удобства обслуживания.

13. Тестирование документации.

14. Тестирование процедур.

15. Выполнение проверки системы непрограммистами.

2. Принципы тестирования.

1. Тестирование - это процесс выполнения программ с целью обнаружения ошибок.
2. Хорошим считается тест, который имеет высокую вероятность обнаружения еще не выявленной ошибки.
3. Удачным считается тест, который обнаруживает еще не выявленную ошибку.
4. Описание предполагаемых значений выходных данных или результатов должно быть необходимой частью тестового набора.
5. Следует избегать тестирования программы ее автором.
6. Тесты для неправильных и непредусмотренных входных данных следует разрабатывать также тщательно как для правильных и предусмотренных.
7. Необходимопроверятьнетол<span class="dash041e\_0431\_044b\_0447\_043d\_044b\_0439\_\_Char" style=" font-family: 'Times New Roman', 'Arial'; font-size

## ЖЦПО.

Для эффективного управления проект должен быть хорошо структурирован. Суть этого процесса сводится к выделению следующих основных элементов:

* фазы жизненного цикла проекта, этапов, работ и отдельных задач;
* организационная структура исполнителей проекта;
* структура распределения ответственности.

*Жизненный цикл* – это последовательность фаз проекта, через которые он должен пройти для гарантированного достижения целей проекта, в нашем случае – для реализации некоторой информационной технологии.

### Адаптация модели жизненного цикла проекта

В данном учебном пособии *по* управлению ИТ-проектами в качестве концептуальной основы используется модель жизненного *цикла* информационных систем (ЖЦ ИС), описанная в стандарте ГОСТ Р ИСО/МЭК 15288. В соответствии с данным стандартом *запуск* каждого нового проекта подразумевает создание (или адаптацию уже имеющейся) модели ЖЦ, состоящей из стадий.

Процесс создания (или адаптации уже имеющейся) модели ЖЦ начинается с определения целей и результатов каждой из стадий, образующих структуру *работ* для детализированного моделирования процессов реализации ИТ. [[10](http://www.intuit.ru/studies/professional_retraining/963/courses/502/literature#literature.10)]

Исходя из допущений базового стандарта, а также типовых этапов ЖЦ ИТ и принятой последовательности их реализации, авторами предлагается следующая модель ЖЦ ИТ, определяющая последовательность изложения материала в книге.

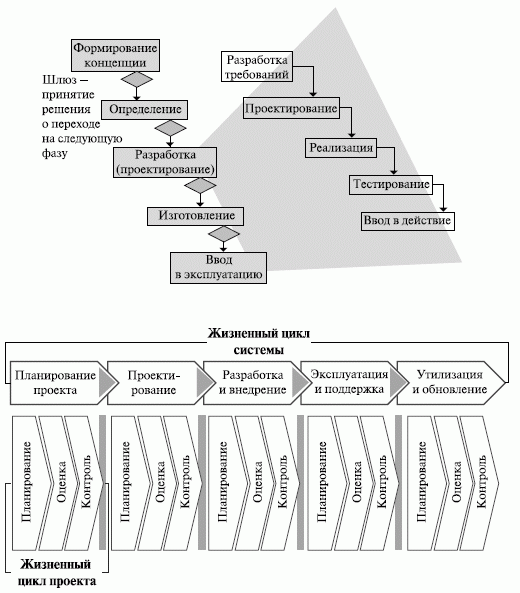
1. Планирование проекта
2. Проектирование
3. Разработка и внедрение
4. Эксплуатация и поддержка
5. Утилизация и обновление

В таблице ниже представлены цели каждой из выделенных стадий ЖЦ (см. [табл. 1.1](http://www.intuit.ru/studies/professional_retraining/963/courses/502/lecture/11390?page=1#table.1.1)).

Приведенные этапы есть стадии *жизненного цикла информационной системы* и не тождественны жизненному циклу проекта. *Жизненный цикл продукта* отражает, что нужно сделать для создания, эксплуатации, поддержки и утилизации данного продукта, а *жизненный цикл* проекта - как нужно организовывать и управлять работой. Фаза ЖЦ продукта может включать в себя все этапы ЖЦ проекта (см. [рис. 1.1](http://www.intuit.ru/studies/professional_retraining/963/courses/502/lecture/11390?page=1#image.1.1) (а, б)), и, в соответствии со стандартом ГОСТ Р ИСО/МЭК 15288 [[10](http://www.intuit.ru/studies/professional_retraining/963/courses/502/literature#literature.10)], предусматривает наличие этапов планирования, оценки и контроля, а также процесса принятия решения - шлюза (см. [рис. 1.1](http://www.intuit.ru/studies/professional_retraining/963/courses/502/lecture/11390?page=1#image.1.1). (а)), через который происходит переход на следующий этап ЖЦ ИС и который является точкой мониторинга качества и точкой принятия решения о целесообразности продолжения проекта [[10](http://www.intuit.ru/studies/professional_retraining/963/courses/502/literature#literature.10)]. Необходимо отметить, что планирование, оценка и *контроль* характерны для любого *цикла* управления (например, цикл Деминга). Таким образом, использование их, в том числе на этапе "*Эксплуатация* и *поддержка*", носящем выраженный *операционный* (не проектный) характер, вполне обосновано.

|  |  |  |
| --- | --- | --- |
| Таблица 1.1. Цели этапов жизненного цикла информационной системы | | |
| Этап (ГОСТ Р ИСО/ МЭК 15288) | Этап (адаптированный) | Цель этапа |
| Замысел | Планирование проекта | Оценка новых возможностей в деловой сфере, разработка предварительных системных требований и проверка их осуществимости. Концептуальное планирование всего ЖЦ ИС |
| Разработка | Проектирование | Создание проекта системы, которая удовлетворяет требованиям приобретающей стороны и может быть реализована, испытана, оценена, применена по назначению, поддержана при применении, в последующем списана и/или обновлена |
| Производство | Разработка и внедрение | Разработка (настройка) системы в соответствии с требованиями приобретающей стороны, тестирование системы, реализация соответствующих организационно-технических мероприятий и развертывание поддерживающих систем, направленных на обеспечение корректной эксплуатации внедренного продукта |
| Применение Поддержка применения | Эксплуатация и поддержка | Использование внедренного продукта в заданных условиях функционирования и обеспечение продолжительной результативности. Осуществление в процессе эксплуатации материально-технического снабжения, технического обслуживания и текущего ремонта, которые обеспечивают непрерывное функционирование рассматриваемой системы и устойчивое предоставление услуг, поддерживающих ее применение |
| Изъятие и списание | Утилизация и обновление | Обеспечение удаления рассматриваемой системы и связанных с нею обслуживающих и поддерживающих организационно-технологических подсистем. Поддержка планирования перехода на новую версию текущей или на абсолютно новую систему |

Рассмотрение каждой стадии ЖЦ ИТ в качестве отдельного проекта позволяет (*по* сути, делает единственно возможным) применять метод планирования *по* принципу набегающей волны, который значительно понижает рискованность проекта и повышает шансы на успех [[9](http://www.intuit.ru/studies/professional_retraining/963/courses/502/literature#literature.9)].



**Рис. 1.1.** (а,б) Примеры соотношения жизненного цикла информационной системы и жизненного цикла проекта

В то же время процессы, выполняемые в рамках одной стадии ЖЦ ИТ, могут иметь взаимосвязи как в рамках данной стадии, так и с процессами других стадий. Очевидно, что для успешного достижения целей проекта необходимо не только управлять каждым процессом в отдельности, но и обеспечить *комплексный подход* к управлению с учетом взаимосвязей, взаимозависимостей как отдельных процессов, так и групп процессов.

С целью структурирования процессы управления проектом принято делить на области знаний. Ниже перечислены области знаний, составляющие процессы проектного управления. Предложенный перечень сформирован на основе рекомендаций лучших мировых практик и содержится в *стандарте управления проектами* [[1](http://www.intuit.ru/studies/professional_retraining/963/courses/502/literature#literature.1),[10](http://www.intuit.ru/studies/professional_retraining/963/courses/502/literature#literature.10), [23](http://www.intuit.ru/studies/professional_retraining/963/courses/502/literature#literature.23)].

1. Управление интеграцией
2. Управление содержанием
3. Управление сроками
4. *Управление стоимостью*
5. Управление качеством
6. Управление рисками
7. Управление человеческими ресурсами
8. Управление коммуникациями
9. Управление конфигурацией

Описание содержания каждой из перечисленных выше областей знаний и соответствующих им процессов приводится в [табл. 1.2](http://www.intuit.ru/studies/professional_retraining/963/courses/502/lecture/11390?page=1#table.1.2).

|  |  |  |
| --- | --- | --- |
| Таблица 1.2. Области знаний проектного управления | | |
| Область знаний | Описание | Процессы |
| Управление интеграцией | Управление интеграцией включает в себя процессы и действия, необходимые для определения, уточнения, комбинирования, объединения и координирования различных процессов и действий по управлению проектом в рамках групп процессов управления проектом. Таким образом, цель данного процесса состоит в достижении эффективного взаимодействия процессов управления проектами, обеспечивающих достижение целей проекта. Эффективное взаимодействие на стадии планирования заключается в формировании базовой линии проекта1 (project baseline), на стадии оценки - в сравнении с базовой линией и корректировке в соответствии с ней на стадии контроля | Разработка ТЭОпроекта.  Разработка устава проекта.  Разработка плана управления проектом.  Руководство и управление исполнением проекта.  Осуществление интегрированного управления изменениями.  Оценка альтернатив развития проекта.  Планирование закрытия проекта и перехода в стадию эксплуатации.  Завершение проекта. |
| Управление содержанием | Управление содержанием включает в себя процессы и действия, обеспечивающие включение в проект всех тех и только тех работ, которые необходимы для успешного выполнения проекта. Оно непосредственно связано с определением и контролем того, что включено или не включено в проект [[1](http://www.intuit.ru/studies/professional_retraining/963/courses/502/literature#literature.1),[23](http://www.intuit.ru/studies/professional_retraining/963/courses/502/literature#literature.23)] | Формирование требований проекта.  Формирование ИСР.  Определение содержания проекта.  Определение результатов всех стадий жц ис.  Оценка реализуемости требований проекта.  Подтверждение содержания проекта.  Определение уточненных системных требований.  Мониторинг содержания и объема проекта.  Оценка готовности пользователей к работе в системе.  Планирование обучения конечных пользователей |
| Управление сроками | Управление сроками проекта включает в себя процессы, обеспечивающие своевременное завершение проекта [[23](http://www.intuit.ru/studies/professional_retraining/963/courses/502/literature#literature.23)] | Формирование списка работ проекта.  Определение последовательности работ проекта.  Оценка трудоемкости и продолжительности работ.  Разработка базовогорасписания проекта.  Контроль и управлениерасписанием проекта |
| Управление стоимостью | Управление стоимостью проекта объединяет процессы, выполняемые в ходе планирования, разработки бюджета и контролирования затрат и обеспечивающие завершение проекта в рамках утвержденного бюджета [[23](http://www.intuit.ru/studies/professional_retraining/963/courses/502/literature#literature.23)] | Оценка стоимости проекта.  Разработка сметы проекта.  Разработка базового плана по стоимости.  Управление стоимостью проекта |
| Управление качеством | Процессы управления качеством проекта объединяют все осуществляющиеся в исполняющей организации операции, определяющие политику, цели и распределение ответственности в области качества таким образом, чтобы проект удовлетворял тем нуждам, для которых он был предпринят. Управление качеством осуществляется посредством системы управления, предусматривающей определенные правила, процедуры и процессы по планированию качества, обеспечению качества и контролю качества, а также операции по их совершенствованию | Формирование программы качества проекта.  Формирование базовой линии требований проекта.  Управление требованиями проекта.  Осуществление обеспечения качества.  Тестирование.  Приемка результатов |
| Управление риском | Процесс управления рисками тесно связан с общим жизненным циклом проекта. На ранних этапах преобладают риски, связанные с бизнесом, рамками проекта, требованиями к конечному продукту и проектированием этого продукта. На стадии реализации доминируют технологические риски, далее возрастает роль рисков, связанных с поддержкой и сопровождением системы. На протяжении всего жизненного цикла проекта возникают новые риски, что требует проведения дополнительных операций анализа и планирования.Согласно ГОСТ Р ИСО/ МЭК 15288-2005 [[10](http://www.intuit.ru/studies/professional_retraining/963/courses/502/literature#literature.10)] цель процесса управления рисками заключается в снижении последствий отрицательного воздействия вероятных событий, которые могут явиться причиной изменений качества, затрат, сроков или ухудшения технических характеристик. В ходе данного процесса проводятся определение, оценка, обработка и мониторинг рисков, возникающих в течение полного жизненного цикла, а также вырабатывается реакция на каждый риск в терминах реализации соответствующих мер противодействия риску или его принятия | Планирование управления рисками.  Идентификация рисков.  Качественный анализ рисков.  Количественный анализ рисков.  Планирование реагирования на риски.  Мониторинг и управление рисками |
| Управление человеческими ресурсами | Управление человеческими ресурсами проекта - это процесс обеспечения эффективного использования человеческих ресурсов проекта, к которым относятся все участники проекта (спонсоры, заказчики, команда проекта, субподрядчики, подразделения компании и другие участники проекта [[13](http://www.intuit.ru/studies/professional_retraining/963/courses/502/literature#literature.13),[17](http://www.intuit.ru/studies/professional_retraining/963/courses/502/literature#literature.17)]) | Планирование человеческих ресурсов.  Набор команды проекта.  Оценка доступности.  Развитие и оценка команды проекта.  Организация инфраструктуры проекта |
| Управление коммуникациями | Управление коммуникациями проекта - это процесс идентификации и эффективного обеспечения всех участников проекта информацией о проекте, а также создания единого образа проекта внутри организации | Идентификация участников проекта.  Формирование стратегии и плана коммуникаций.  Реализация плана коммуникаций и сбор обратной связи |
| Управление конфигурацией | Управление конфигурацией - процесс управления аппаратными средствами, программным обеспечением, данными, а также документацией в ходе разработки, тестирования и использования информационных систем. Цель процесса управления конфигурацией состоит в установлении и поддержании целостности всех идентифицированных выходных результатов проекта или процесса обеспечения доступа к ним любой заинтересованной стороны.Задачи управления конфигурацией проекта:  определение стратегии управления конфигурацией, включающей следующие вопросы:  - определение полномочий на запрет или разрешение доступа, реализацию и контроль изменений элементов конфигурации;  определение места и условий хранения элементов конфигурации, включая требования к окружающей среде, а в случае информации - требования к хранению носителей информации в соответствии с назначенными уровнями целостности, защищенности и безопасности;  определение критериев или событий, соответствующих началу контроля конфигурации и сопровождения базовых линий в процессе эволюции конфигураций;  определение стратегии аудита и ответственности за гарантии непрерывной целостности и защищенности информации, описывающей конфигурацию;  идентификация элементов, которые необходимо контролировать в процессе управления конфигурацией;  поддержка информации о конфигурации на приемлемом уровне целостности и защищенности. Для этого рекомендуется:  поддерживать записи о конфигурации в течение всего жизненного цикла и архивировать их в соответствии с соглашениями, законодательством или передовым производственным опытом;  описывать конфигурацию в соответствии с производственным или технологическим стандартами там, где это возможно.  регистрировать обоснования для изменений базовой линии конфигурации и связанные с этим данные о соответствующих разрешениях;  гарантирование того, что изменения базовой линии конфигурации соответствующим образом идентифицируются, записываются, оцениваются, утверждаются, проводятся и верифицируются. Для этого рекомендуется:  регистрировать этапы конфигурации;  управлять выполнением записей, изменениями и утверждениями текущего статуса конфигурации и статуса всех предыдущих конфигураций для подтверждения корректности, своевременности, целостности и защищенности информации;  проводить аудит для проверки соответствия базовой линии УК требованиям к результатам проекта | Идентификация объектов управления конфигурацией.  Планирование инфраструктуры стадии разработки.  Установление базовой линии конфигурации проекта.  Оценка соответствия базовой линии конфигурации.  Контроль конфигурации выделенных элементов проекта.  Обеспечение целостности элементов конфигурации.  Реконфигурация инфраструктуры проекта |

Каждый из этапов ЖЦ ИТ и ЖЦ проекта предусматривает совокупность задач, с полной матрицей которых можно ознакомиться в Приложении 1.

В рамках конкретных проектов предложенные этапы ЖЦ ИТ, а также и отдельно взятые процессы ЖЦ ИТ могут быть индивидуально отобраны, идентифицированы и, при необходимости, модифицированы для достижения измененных целей и результатов соответствующих стадий.

Сделанные изменения должны быть задокументированы. Общие требования к процедуре модификации таковы: любой новый процесс жизненного *цикла* определяется и документируется в терминах его назначения, целей и результатов. Ответственным за такого рода модификации является, как правило, руководитель соответствующего проекта. В то же время утверждение адаптированной, сокращенной или дополненной модели ЖЦ ИС обычно производит *офис* управления проектами или иная организационная *единица*, в круг обязанностей которой входит поддержание целостности и актуальности корпоративной методологии управления проектами [[8](http://www.intuit.ru/studies/professional_retraining/963/courses/502/literature#literature.8)].

Приведенная процедура и *шаблон* документирования модификации ЖЦ ИТ являются одним из возможных вариантов оформления соответствующих действий над ЖЦ ИТ.

#### Процедура адаптации модели ЖЦ ИС

При адаптации модели ЖЦ ИТ в интересах организации или проекта в соответствии с применяемыми политикой и процедурами должны выполняться следующие действия.

1. Руководителем проекта (РП) определяются и документируются обстоятельства, воздействующие на адаптацию. Эти воздействия включают (но не ограничиваются) перечисленное ниже:
   * стабильность и разнообразие среды функционирования;
   * коммерческие или эксплуатационные риски, касающиеся заинтересованных сторон;
   * новизну, размеры и сложность;
   * дату начала и продолжительность применения;
   * вопросы целостности, такие как безопасность, защищенность, секретность, удобство применения,
   * доступность;
   * вновь возникающие технологические возможности;
   * бюджетный профиль и доступные организационные ресурсы;
   * готовность предоставления услуг обеспечивающими системами.
2. При наличии свойств, критичных по отношению к системе, руководитель проекта должен учесть структуры ЖЦ, которые рекомендованы или установлены в качестве обязательных стандартами, соответствующими области критичности.
3. Далее руководитель проекта собирает входные данные от следующих заинтересованных сторон проекта:
   * правообладатели системы;
   * заинтересованные стороны соглашения, заключенного организацией;
   * стороны, вносящие вклад в *организационные функции*.
4. Руководитель проекта определяет новую (модифицированную) модель жизненного цикла системы в терминах стадий, их назначения, целей и результатов, которые достигаются вследствие применения процессов жизненного цикла в пределах каждой стадии.
5. Проектный офис принимает решение об адаптации базовой модели.
6. Модификация ЖЦ ИС приобретает локальный (для одного проекта и для одной (под)системы) или общекорпоративный характер по решению проектного офиса, по результатам апробации предложенной РП модификации.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Таблица 1.3. Шаблон адаптации модели жизненного цикла информационной системы | | | | | | | |
| Описание причины: | | | | | | | |
| Действия | Базовый | | Модифицированный | | Характеристики модифицированного этапа/процесса | | |
| Этап | Процесс | Этап | Процесс | Назначение | Цель | Результат |
| \_ |  |  |  |  |  |  |  |
| \_ | | | | | | | |
| Дата подачи заявки (руководитель проекта): | | | | | | | |
| Дата принятия решения (проектный офис): | | | | | | | |
| Дата начала применения: | | | | | | | |

### Разработка технико-экономического обоснования

Традиционно основной целью подготовки технико-экономического обоснования ( *ТЭО* ) ИТ-проекта является получение финансирования на реализацию соответствующей инициативы. Кроме того, корректно составленное *ТЭО* может решать следующие задачи:

* приоритизация проектов в условиях ограниченных финансовых, человеческих и прочих ресурсов;
* определение совокупности организационно-технологических мероприятий по обеспечению заявленных бизнес-выгод от реализации проекта;
* обеспечение заинтересованности руководителей бизнес-подразделений в проекте;
* формирование основы для оценки соответствия результатов проекта и первоначальных планов.

Согласно последним исследованиям 75% компаний ставит именно такие цели при подготовке *ТЭО*, в то же время всего лишь 40% из них считают, что используемые ими методы позволяют получить корректную оценку эффективности внедряемого ИТ-решения [[7](http://www.intuit.ru/studies/professional_retraining/963/courses/502/literature#literature.7)].

Помимо обозначенных задач *ТЭО* может обеспечивать входную информацию для устава проекта, рассматриваемый в данной книге как ключевой документ интегрированного управления проектом. Для того чтобы *ТЭО* обеспечивал качественную информацию, рекомендуется следующим образом структурировать идентифицированные бизнес-выгоды ИТ-проекта (см. [табл. 1.4](http://www.intuit.ru/studies/professional_retraining/963/courses/502/lecture/11390?page=2#table.1.4).).

В соответствии с предлагаемым подходом [[7](http://www.intuit.ru/studies/professional_retraining/963/courses/502/literature#literature.7)] бизнес-выгоды можно классифицировать *по* двум факторам: (1) характеру воздействия на бизнес и (2) степени определенности. Таким образом, каждая выгода *по* проекту размещается "на пересечении" соответствующих значений двух обозначенных факторов.

Использование матрицы структурирования выгод начинается с определения характера воздействия на бизнес каждой из них. Определены три типа воздействия.

1. *Создание новых возможностей*: функциональность информационной системы, ранее не доступная компании, ее контрагентам или иным заинтересованным сторонам.
2. *Повышение эффективности операций*:функциональность новой информационной системы позволяет выполнять существовавшие до нее операции гораздо более эффективно.
3. *Отказ от операций*:информационная система позволяет отказаться от выполнения операций, утративших свою актуальность для бизнеса компании в связи с изменением бизнес-процессов.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Таблица 1.4. Матрица структурирования выгод ИТ-проекта (адаптировано из [[7](http://www.intuit.ru/studies/professional_retraining/963/courses/502/literature#literature.7)]) | | | | |
|  | | **ХАРАКТЕР ВОЗДЕЙСТВИЯ НА БИЗНЕС** | | |
| Создание новых возможностей | Повышение эффективности операций | Отказ от операций |
| СТЕПЕНЬ ОПРЕДЕЛЕННОСТИ | Финансовые |  |  |  |
| Количественные |  |  |  |
| Измеримые |  |  |  |
| Качественные |  |  |  |

После определения характера воздействия необходимо классифицировать каждую бизнес-выгоду *по* степени определенности (от менее определенных к более определенным): наблюдаемые (качественные), измеримые, количественные, финансовые.

1. *Качественные*: выгоды, которые могут быть зафиксированы на уровне экспертного мнения или суждения. В то время как данный тип выгод вполне допустим, необходимо всегда предупреждать ситуацию, когда без четкого значения выгоды на этапе планирования очень сложно определить степень ее реализации на момент принятия результатов проекта. В связи с этим рекомендуется разрабатывать четкие критерии реализации качественных выгод в самом начале проекта и, по возможности, собирать дополнительную информацию для "переноса" качественных выгод в более объективные категории.
2. *Измеримые*:выгоды данного типа поддаются измерению. В распоряжении аналитика есть инструменты и техники, например, ключевые показатели эффективности, позволяющие измерить их значение до внедрения. Для данного типа бизнес-выгод характерна невозможность оценить значение соответствующего показателя после внедрения.
3. *Количественные*:аналогично измеримым, количественные выгоды характеризуются наличием показателей, позволяющих измерить их значение до выполнения проекта. Но, в отличие от измеримых, значение показателей количественных бизнес-выгод на момент окончания проекта можно оценить с высокой степенью точности.
4. *Финансовые*:это тип бизнес-выгод, которые могут быть выражены в терминах финансовых показателей. Отнесение бизнес-выгоды к данной категории должно производиться только в том случае, если в распоряжении аналитика имеется достаточно достоверная информация о финансовой оценке соответствующих показателей. Очевидно, финансовые выгоды есть результат "обогащения" количественных бизнес-выгод финансовыми данными. Агрегированные финансовые выгоды проекта образуют базу для построения финансовой модели проекта (ROI-модель) и расчета инвестиционных показателей: NPV, *IRR*, периода окупаемости.

Выбор той или иной категории для конкретной бизнес-выгоды производится на основе доступной информации о ней до момента реализации инвестиций. Каждая бизнес-выгода на момент ее идентификации относится к наименее определенной категории - наблюдаемая. *По* ходу анализа необходимо максимальное количество бизнес-выгод перенести в финансовую категорию для построения экономической модели окупаемости проекта, кроме доходной части, в которой должна быть отражена и расходная. В качестве инструмента оценки стоимости проекта и системы авторы рекомендуют использовать модель совокупной стоимости владения системы (*TCO*), рассмотрение которой будет произведено в разделе, посвященном *управлению стоимостью* проекта.

### Формирование бизнес-цели проекта

Бизнес-цель - это описание фактора, побуждающего к выполнению проекта. Ее формирование производится на стратегическом уровне, то есть *бизнес-цель* выступает в качестве связующего звена между глобальными задачами, стоящими перед организациями, и планируемым к реализации проектом. При отходе от стратегического видения происходит смещение бизнес-цели в сторону тактических и даже операционных задач, на уровне которых целью проекта видится "просто выдать продукт", а не достичь какой-либо тактической цели, поддерживающей стратегические цели организации. Этого нельзя допускать: *бизнес-цель* проекта должна всегда носить тактический или стратегический характер, но в то же время быть предельно точной и ясной (очень редко удается применить широко известный метод *SMART* к построению *бизнес-цели проекта* ).

Так, например, *бизнес-целью проекта* *по* приобретению и установке нового производственного оборудования является не покупка и установка оборудования, а устранение узкого места в производственном процессе и обеспечение надлежащих объемов выпуска, гарантирующих удовлетворение спроса и завоевание определенной доли рынка. Аналогично, проект внедрения информационной системы имеет своей бизнес-целью не разворачивание технических средств, а создание информационно-технологического фундамента для поддержки принятия руководством компании своевременных управленческих решений, направленных на обеспечение ее развития и роста.

Бизнес-цель должна быть достаточно веской, чтобы организация решилась перейти к разработке устава проекта, документа, в соответствии с лучшими практиками инициирующего выполнение проекта. В качестве инструмента, позволяющего определить необходимость реализации проекта, может быть использовано *ТЭО*, или бизнес-кейс, проекта.

### Разработка устава проекта

*Устав проекта* - это инструмент, который формально авторизует проект и является звеном, соединяющим предстоящий проект с текущей работой организации. Данный документ обычно отражает ситуацию со стороны организации-заказчика, выпускается руководителем, внешним *по* отношению к проекту, и назначает менеджера проекта, наделяя его полномочиями на использование в проекте ресурсов организации. Это особенно актуально в функционально-ориентированных и *матричных организациях*, т.е. в тех компаниях, где менеджеры не имеют непосредственной власти над членами проектной команды и другими ресурсами, но несут ответственность за выполнение проекта. Для того чтобы устав имел силу в подобной ситуации, издающий его руководитель, или *спонсор проекта*, должен находиться на том уровне, который подразумевает наличие контроля над ресурсами. Часто датой начала проекта считается день, следующий за подписанием устава.

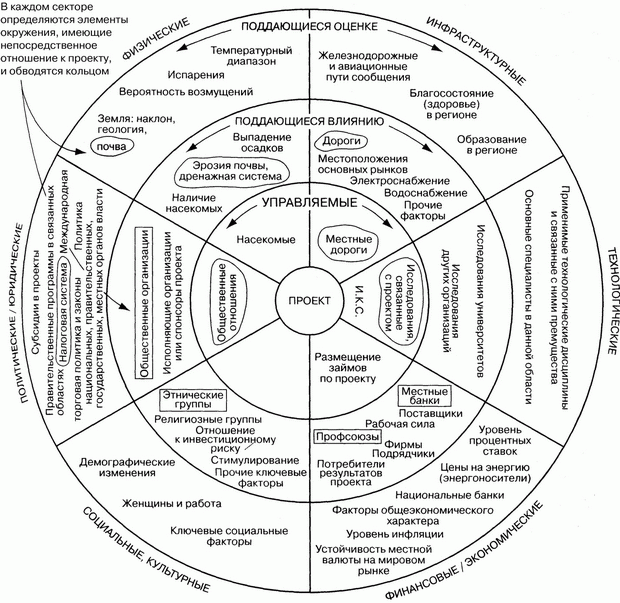
Процесс разработки устава проекта уже подразумевает, что компания заинтересована в достижении какой-то цели или решении имеющейся проблемы и готова выделять под это ресурсы. Следовательно, со стороны организации-заказчика есть мотив инвестировать средства и ресурсы в генерацию такой информации, которая позволит разработать корректный *устав проекта*. К информации, имеющей ключевое *значение* для составления устава, относятся:

* стратегические и тактические цели организации-заказчика;
* формулировка требований организации-заказчика;
* *ТЭО* ;
* контракт;
* внутрикорпоративная методология управления проектами и соответствующие политики.

Решение о выполнении проекта - итог процесса отбора проектов, основанного на информации, которая изложена в вышеуказанных документах. Таким образом, крайне важно давать прямую ссылку в соответствующих разделах устава на них с тем, чтобы придать уставу больший *вес*.

В [табл. 1.5](http://www.intuit.ru/studies/professional_retraining/963/courses/502/lecture/11390?page=2#table.1.5) приведены требования к уставу проекта - перечислены обязательные *разделы* с необходимыми рекомендациями и пояснениями к их наполнению.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Таблица 1.5. Требования к уставу проекта | | | | | |
| **№** | **Раздел** | | **Пояснения** | | |
| **1**. | **Название проекта** | | Каждый проект должен иметь название, отражающее его суть и в то же время достаточно яркое для привлечения внимания | | |
| **2**. | **Бизнес-причинавозникновенияпроекта** | | Производственная необходимость, или самое общее описание проекта и требований к продукту, производство которого является результатом выполнения проекта. Формулировка причины фактически дает ответ на вопрос, зачем выполняется данный проект. Причины возникновения проекта могут основываться на требованиях рынка, техническом прогрессе, юридических требованиях или государственном стандарте | | |
| **3**. | **Бизнес-цель** | | Сформулирована заказчиком, исходя из стратегических и тактических целей компании, см. раздел "Формирование*бизнес-цели проекта* " | | |
| **4**. | **Требования**,**удовлетворяющиепотребности**,**пожелания** **и ожиданиязаказчика**,**спонсора** **и другихучастниковпроекта** | | Видение организацией-заказчиком, как правило, высокоуровневое, способов достижения поставленной бизнес-цели или решения существующей проблемы. Проект считается успешным, если ожидания заказчика и участников проекта оказались выполненными, следовательно, к моменту формирования устава проекта его участники должны быть идентифицированы. Все задокументированные в уставе требования должны быть учтены при выполнении стоимостной оценки проекта | | |
| **5**. | **Расписание основных контрольных событий** | | На этапе формирования устава должно быть обязательно указано время начала и завершения проекта; при необходимости отмечаются ключевые *вехи проекта*, принципиальные для организации-заказчика. Вообще рекомендуется ограничить количество *контрольных событий* теми, которые абсолютно необходимы, т.е. обычно тремя-пятью. Иными словами, принимая во внимание цель устава и соответствующий уровень детализации, совершенно излишне разрабатывать длинный список событий - это только создаст дополнительные ограничения для выбора методологии реализации проекта. Кроме того, организации, придающие значение себестоимости, имеют тенденцию указывать для основных событий специфику бюджета ресурсов или бюджета средств [[18](http://www.intuit.ru/studies/professional_retraining/963/courses/502/literature#literature.18)] | | |
| **6**. | **Участники проекта** | | Перечисление заинтересованных сторон проекта, иными словами, круга лиц и организаций, на которых оказывает воздействие реализация данного проекта и которые сами могут воздействовать на него. Подробнее об участниках проекта см. раздел "Идентификация участников проекта" | | |
| **7**. | **Окружение проекта** | | Перечисление всех организационных факторов, характеризующих обстановку вокруг проекта и на рынке. Также необходимо указать благоприятные и неблагоприятные особенности среды, в которой проект будет выполняться (внутри и вне компании), и способность организации-исполнителя к его осуществлению, а организации-заказчика - к использованию его результатов. Далее (см. [рис. 1.2](http://www.intuit.ru/studies/professional_retraining/963/courses/502/lecture/11390?page=3#image.1.2)) будет показан один из эффективных способов выполнения комплексного анализа окружения и участников проекта. При использовании этого подхода сначала определяется достаточно большое число факторов, *действующих в окружении проекта*; они заносятся в соответствующий сектор. Затем выделяются наиболее критичные из них (прямоугольники - участники, овалы - факторы окружения) [[20](http://www.intuit.ru/studies/professional_retraining/963/courses/502/literature#literature.20)] | | |
| **8**. | **Допущения относительно организации и окружения, а также внешние допущения** | | Набор условий, которые должны быть выполнены наряду с созданием продукта проекта, для достижения результата проекта. Допущения обуславливают риски проекта; во время проекта происходит их мониторинг. Пример допущений:   * o *компетенции команды проекта достаточно для выполнения предпроектного обследования;* * o *организацией-заказчиком будет выделен персонал для выполнения работ по поддержке проекта*.   Обратите внимание, что при составлении устава проекта допущения формулируются со стороны организации-заказчика об организации-исполнителе | | |
| **9**. | **Ограничения относительно организации и окружения, а также внешние ограничения** | | Ограничение указывает на условие, которое нельзя нарушать в процессе создания продукта проекта, или условие, которому ни при каких обстоятельствах не должен удовлетворять продукт проекта. Ограничения к тому же указывают на возможности команды проекта по выбору вариантов для выполнения любых проектных работ [[11](http://www.intuit.ru/studies/professional_retraining/963/courses/502/literature#literature.11)]. Пример ограничений проекта:   * *увеличение стоимости проекта не более чем на 10%;* * *не менее 40% членов команды проекта, предоставляемых исполнителем, заняты на 100% в проекте*.   Обратите внимание, что при составлении устава проекта ограничения формулируются со стороны организации-заказчика об организации-исполнителе и о проекте в целом | | |
| **10**. | **Объем денежных средств, выделенных на достижение бизнес-цели** | | На данном этапе указывается сумма средств, которую организация-заказчик готова выделить на достижение сформулированной *бизнес-цели проекта*. Указанная сумма является результатом определения порядка величины и ошибка в оценке может составлять от ~ -20% до +100% [[18](http://www.intuit.ru/studies/professional_retraining/963/courses/502/literature#literature.18)] | | |
| **11**. | **Назначениеруководителейпроекта и общееопределениеполномочийключевых членовпроектнойкоманды: РП**,**спонсор**,**координатор** | | Руководитель проекта назначается уставом проекта и формально приступает к выполнению своих обязанностей на следующий день после подписания устава проекта. Руководитель, или менеджер, проекта несет основную ответственность за общее планирование, направление и контроль проекта в течение всех фаз его жизненного цикла, ставя целью получение желаемого результата в рамках утвержденного бюджета и расписания. Основная задача руководителя проекта - объединение усилий всех лиц, участвующих в проекте. Для решения этой задачи менеджер проекта наделяется полномочиями по проекту, т.е. правом отдавать функциональным лидерам проекта распоряжения, необходимые для планирования, исполнения, мониторинга, оценивания и контроля работ, которые должны быть выполнены по данному проекту. Руководство проектом также включает в себя получение информации, необходимой для планирования, мониторинга, оценивания и контроля проекта [[8](http://www.intuit.ru/studies/professional_retraining/963/courses/502/literature#literature.8),[18](http://www.intuit.ru/studies/professional_retraining/963/courses/502/literature#literature.18)]. Роль спонсора проекта обычно берет на себя (не назначается!!!) менеджер высшего звена, который действует от лица руководства компании, финансирующей или исполняющей проект2. Ключевая задача спонсора заключается в обеспечении ресурсов проекта, в том числе административных, а также в обеспечении связи между проектом и руководством организации-заказчика. На проекте спонсор является лицом, принимающим те решения, которые находятся за пределами полномочий руководителя проекта, например:   * *утверждать бизнес-цели проекта* *, включая расписания и бюджет, и вносимые в них изменения;* * *назначать и утверждать менеджера проекта, а также утверждать соответствующую должностную инструкцию и порядок подчинения;* * *формировать стратегические указания для менеджера проекта по ходу отслеживания результатов проекта;* * *вносить и утверждать основные изменения по проекту и решения, касающиеся выделения ресурсов;* * *принимать решения о внесении изменений в базовую линию проекта*.   Роль спонсора проекта обычно не предполагает работы с полной занятостью вне зависимости от размера проекта [[8](http://www.intuit.ru/studies/professional_retraining/963/courses/502/literature#literature.8),[18](http://www.intuit.ru/studies/professional_retraining/963/courses/502/literature#literature.18)]. Администратор (координатор) проекта - это специфическая функция на проекте, которая необходима для поддержки работ, связанных с администрированием и документированием функционирования *проектной организации* и обеспечением инфраструктуры проекта. Работа администратора имеет своей ключевой задачей поддержку руководителя проекта на операционном уровне с целью его высвобождения для интеллектуально-сложных задач. В обязанности координатора проекта может входить: администрирование проектных контрактов и договоров на протяжении всего ЖЦ, организация периодического сбора статуса выполнения проекта и т.п. сбор статуса - словосочетание, не несущее смысла, если только это не специфический термин. Я прошу обратить на него внимание. М/б, сбора информации о текущем статусе? Формировать всю команду и тем более сразу указывать имена всех ее членов не принято - функциональные руководители обычно выделяют для проекта своих подчиненных, только когда руководитель проекта составит план потребности в ресурсах, после определения состава работ проекта, и отправит официальный запрос на ресурсы, утвержденный спонсором проекта [[18](http://www.intuit.ru/studies/professional_retraining/963/courses/502/literature#literature.18)] | | |
| Таблица 1.6. Шаблон листа управления документом | | | | | |
| **Авторы** | | |  | | |
| **Файл** | | |  | | |
| **Дата создания** | | |  | | |
| **Дата последнего редактирования** | | |  | | |
| **Количество страниц** | | |  | | |
| **Версия** | | **Дата изменения** | **Краткое описание изменения** | **Автор изменения** | **Подпись** |
| **01** | |  |  |  |  |
| **02** | |  |  |  |  |
| **Согласование документа** | | | | | |
| **Замечания** | | | | | |
| **№** | | **Дата поступления** | **Наименование документа** | **Автор замечания** | **Подпись** |
| **1**. | |  |  |  |  |
| **2**. | |  |  |  |  |
| **Обработка замечаний** | | | | | |
| **№** | | **Дата обработки** | **Версия документа, учитывающая замечание** | **Исполнитель** | **Подпись** |
| **1**. | |  |  |  |  |
| **2**. | |  |  |  |  |

[](http://www.intuit.ru/EDI/29_06_15_4/1435529976-29105/tutorial/907/objects/1/files/01_02.gif)

**Рис. 1.2.** Модель комплексного анализа участников и окружения проекта (Burnett, 1980). На рисунке ИКС - исследовательская команда спонсора

После подготовки устава в соответствии с предложенным шаблоном рекомендуется произвести проверку его корректности. *Автор* устава, как правило, *спонсор проекта*, должен обязательно убедиться, что:

* информация в уставе, а также сделанные в нем допущения соответствуют исходным данным, которые содержатся в стратегических и тактических целях организации-заказчика, задокументированных предварительных требованиях организации-заказчика, *ТЭО*, контракте и внутрикорпоративных политиках и методологиях;
* все разделы предложенного формата устава проекта заполнены в соответствии с рекомендациями;
* в самом документе отсутствуют противоречия;
* список рассылки устава проекта включает все функциональные группы, сотрудники которых должны войти в проектную команду, а в идеале - и всех участников проекта внутри организации-заказчика.

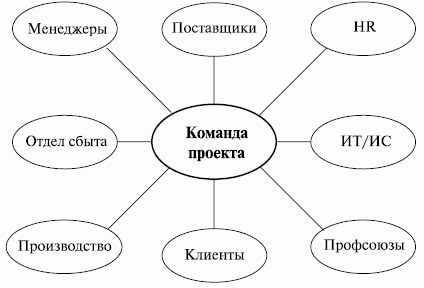
*Устав проекта* является установочным документом, описывающим *связь* проекта с операционной деятельностью компании. *По* этой причине внесения значительных изменений в текст данного документа не рекомендуется, а при возникновении такой обоснованной необходимости стоит разработать новый *устав проекта*, более полно отвечающий реалиям реализуемого проекта. В то же время для обеспечения контроля версионности в процессе разработки устава проекта необходимо использовать т.н. *лист* управления документом, представленный в [табл. 1.6](http://www.intuit.ru/studies/professional_retraining/963/courses/502/lecture/11390?page=3#table.1.6).

### Идентификация и анализ участников проекта

Заинтересованная сторона (участник, или стейкхолдер3 ) - это любое лицо, которое само оказывает влияние на проект или подвергается влиянию проекта и результатов его реализации [[5](http://www.intuit.ru/studies/professional_retraining/963/courses/502/literature#literature.5)].

На начальной фазе ЖЦ ИС, фазе планирования, целевой группой всегда является руководство компании, на которое следует обращать особое внимание и наиболее тесно взаимодействовать с ним. Кроме того, на данной фазе руководство компании будет и единственной точкой опоры проекта в организации, поэтому нужно четко себе представлять, чем отличаются руководители среднего звена от прочих сотрудников [[5](http://www.intuit.ru/studies/professional_retraining/963/courses/502/literature#literature.5)].

Вообще процесс идентификации заинтересованной стороны стоит начинать с построения карты участников проекта, на которой мы уже сразу можем произвести классификацию участников проекта*по* различным категориям. В качестве примера предлагается следующий вариант карты заинтересованных сторон проекта, представленный на [рис. 1.3](http://www.intuit.ru/studies/professional_retraining/963/courses/502/lecture/11390?page=3#image.1.3). При разработке карты заинтересованных сторон проекта всегда стоит помнить следующие рекомендации



**Рис. 1.3.** Пример карты участников проекта (адаптировано из [5])

1. Организация не является единым целым, но представляет собой совокупность отношений между различными заинтересованными сторонами. Построение карты заинтересованных сторон есть первый шаг на пути выявления взаимосвязей между ними.
2. Необходимо выявить всех участников проекта, и в этом аспекте построение красивых и однозначных схем является вторичным по отношению к значимости формирования исчерпывающего списка.
3. Карта заинтересованных сторон не является статической, по мере продвижения проекта она будет уточняться: изначально включенные участники могут быть исключены из рассмотрения, а на поздних этапах могут быть идентифицированы новые.

После выполнения идентификации заинтересованных сторон следует *анализ* участников проекта, в рамках которого необходимо выяснить уровень воздействия каждого из стейкхолдеров на проект и произвести оценку их вовлеченности в проект.

Для анализа воздействия участников на проект рекомендуется использовать *шаблон*, приведенный в [табл. 1.7](http://www.intuit.ru/studies/professional_retraining/963/courses/502/lecture/11390?page=3#table.1.7).

|  |
| --- |
| Таблица 1.7. Анализ воздействия участников проекта (адаптировано из [5]) |
| **http://www.intuit.ru/EDI/29_06_15_4/1435529976-29105/tutorial/907/objects/1/files/01_07sm.gif** |

**Анализ воздействия** производится в разрезе двух аспектов.

1. Степень организационного влияния участника проекта

Степень участия заинтересованной стороны проекта в принятии стратегически важных для компании решений, ее влияние на реализацию различных инициатив. Крайне важно при подобном анализе не упустить из рассмотрения и неформальных лидеров организации.

1. Воздействие участников на реализуемый ИТ-проект

Данный показатель характеризует, как конкретный участник может повлиять на проект, насколько важна его *поддержка* и опасно его неприятие результатов проекта.

Иногда результаты данного анализа могут быть довольно неожиданными: агенты изменений из отделов, "отдаленных" от проекта, требуют гораздо большего внимания, чем сотрудники отдела, реализующего проект, которые оказались в нижнем левом углу. Данный *анализ* позволяет правильно расставить приоритеты и интенсивность использования типично ограниченных ресурсов, в том числе на реализацию стратегию коммуникаций.



Если *анализ* воздействия участников позволяет приоритизировать использование ограниченных ресурсов проекта, то оценка вовлеченности позволяет определить степень сопротивления различных участников проекта, которое характеризуется в разрезе двух аспектов [[5](http://www.intuit.ru/studies/professional_retraining/963/courses/502/literature#literature.5)].

1. Доверие

Насколько спонсор(ы) и прочие ключевые участники готовы участвовать в проекте и работать с командой до получения итогового результата, насколько можно рассчитывать на их поддержку в критический момент на проекте?

1. Согласие

Получилось ли достичь согласия с этим конкретным участником (группой участников проекта)? Разделяют ли они точку зрения руководителя проекта?

Поскольку на фазе планирования проекта мы в большей степени говорим именно о руководителях высшего звена - о потенциальных *спонсорах проекта*, то и результаты данного анализа будут в большей степени применимы именно к этой категории сотрудников.

В зависимости от того, в каком из четырех квадрантов образовавшейся матрицы оказывается тот или иной участник (*группа* участников), они относятся к нижеуказанным группам, принципы работы с которыми весьма разнятся. Руководителю проекта необходимо установить конструктивный диалог с двумя группами, обладающими высоким уровнем доверия, и сохранять разумную пропорцию их участия.

1. Союзники

Поддерживают имеющиеся ожидания по проекту, в основном разделяют видение и, скорее всего, заинтересованы в результатах.

1. Конкуренты

Они заставляют команду руководства проекта постоянно конкурировать за ресурсы, обосновывать значимость проекта и искать наиболее оптимальные и менее затратные способы реализации запланированного - по сути, таким образом, выявляя в последних лучшее.

Спонсоры и ключевые участники проекта, обладающие низким уровнем доверия, характеризуются также низким уровнем готовности работать над проектом. Две группы, которые относятся к данной категории, - это партнеры и противники.

1. Партнеры

На первый взгляд, с ними удалось прийти к согласию, но первые их действия свидетельствуют о нерешительности и несоответствии заявленному. Рекомендуется с каждым участником данной категории провести личную беседу, касающуюся его роли на проекте, касающееся их обязательств, для идентификации причин, не позволяющих им действовать более решительно и организованно.

1. Противники

В отличие от партнеров признают свою неготовность действовать - но *конфликт* с людьми, открыто выражающими свою позицию, маловероятен, поэтому действия *по* вовлечению их в проект аналогичны действиям *по* отношению к партнерам: общение, беседа *по* вопросам их обязанностей и ответственности.

На крупных проектах рекомендуется создавать базу данных участников проекта, в которой будет храниться *информация* о сотрудниках, способных, так или иначе, оказать влияние на результаты реализации проекта. Хранимая *информация* включает в себя:

* ФИО и контактную информацию сотрудника;
* сведения об организационной единице, в которой работает сотрудник;
* определение функциональной *роли сотрудника*;
* категорию получаемых сообщений и их историю.

Наличие такой *базы данных* позволяет менеджеру проекта, во-первых, держать информацию об участниках проекта всегда под рукой, а во-вторых, избегать неловких ситуаций, когда *менеджер*забывает с кем-то лично побеседовать.

### Формирование требований проекта

Данный процесс направлен на изучение требований заказчика, которые должны быть уже отражены в уставе проекта, и перенос их в более конкретные термины требований проекта, на основе которых уже формируется *список* проектных *работ* и *программа* качества проекта. Для получения корректной информации необходимо в самом начале правильно организовать ее сбор, который на ИТ-проектах чаще всего реализуется в форме интервью с заказчиком (см. [рис. 1.5](http://www.intuit.ru/studies/professional_retraining/963/courses/502/lecture/11390?page=4#image.1.5)).

#### Организация и проведение результативного интервью

1. Подготовка

Изначально необходимо точно сформулировать основные идеи, которые определяют цель визита к заказчику. Участники команды проекта должны очень ясно представлять себе, какого рода информация нужна и каким образом она будет использована, с тем чтобы точно определить целевую аудиторию. Только после этого необходимо переходить к разработке списка вопросов, которые будут заданы заказчику.

Основная рекомендация для проведения интервью для формирования функциональных требований к системе - использовать подход, применяемый в структурном моделировании, то есть постараться задать вопросы и получить информацию в разрезе, как показано на [рис. 1.4](http://www.intuit.ru/studies/professional_retraining/963/courses/502/lecture/11390?page=4#image.1.4).

* + "Большая картинка"
    - Место (под)процесса в сквозном процессе
    - Интерфейс с предшествующим процессом
    - Интерфейс с последующим процессом
  + Описание процесса
    - Что? (типовой результат и его потребитель)
    - Как? (последовательность шагов и показатели эффективности)
    - Кто? (роли и присущая им квалификация)
    - Чем? (используемые средства, инструменты, расходуемые ресурсы)
  + Документы
    - Входящие документы
    - Исходящие документы
    - Регламентирующие документы

1. Проведение

Рекомендуется планировать интервью исходя из максимальной продолжительности в 1,5 часа. Учитывая весьма ограниченное время, необходимо всегда приходить в указанный час, заранее продумать маршруты пути к месту встречи. Каждый участвующий в процессе общения с заказчиком должен оперировать одним и тем же набором вопросов в одинаковой последовательности, чтобы обеспечить порядок проведения переговоров и удобство при дальнейшем формировании протокола интервью.

[](http://www.intuit.ru/EDI/29_06_15_4/1435529976-29105/tutorial/907/objects/1/files/01_04.gif)

**Рис. 1.4.** Принцип структурирования информации о бизнес-процессе

На практике для проведения интервью часто составляют команды по два человека, это удобно с той точки зрения, что один задает вопросы, а другой делает пометки. В то же время присутствие на встрече более двух интервьюеров способно значительно повредить общению.

1. Дальнейшие действия

После завершения всех встреч нужно собрать полученную информацию в единый отчет, который будет полезен проекту. Обычно это реализуется при тесном взаимодействии всех участников интервью, в том числе и интервьюированных представителей заказчика, которые согласуют подготовленный сводный протокол интервью.

В качестве отчета по интервью готовится сводный протокол, который затем отправляется на согласование. Пример шаблона протокола представлен в [табл. 1.8](http://www.intuit.ru/studies/professional_retraining/963/courses/502/lecture/11390?page=4#table.1.8).

Крайне важным моментом является "встраивание" информации, полученной на интервью от заказчика, в проектную документацию - иначе вся собранная информация не имеет никакого значения для проекта. Для того чтобы проверить, были ли учтены *требования заказчика* в проектных спецификациях, рекомендуется ответить на следующие вопросы:

* + Определены ли факторы ценности для заказчика?
  + Усвоила ли команда проекта эти факторы?
  + Были ли факторы ценности интегрированы в процессы и проектные продукты?

Инструментом, который позволяет обеспечить положительные ответы на эти вопросы, является функция качества или "дом качества", речь о котором пойдет ниже, в разделе, посвященном формированию требований проекта.

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица 1.8. Шаблон протокола интервью | | | |
| **УПРАВЛЕНИЕ ДОКУМЕНТОМ** | | | |
| Автор |  |  |  |
| Дата создания |  |  |  |
| **ИНФОРМАЦИЯ О ВСТРЕЧЕ** | | | |
| Время и дата |  |  |  |
| Порядковый номер |  |  |  |
| Адрес/ место |  |  |  |
| **УЧАСТНИКИ ВСТРЕЧИ** | | | |
| Со стороны заказчика | *[ФИО, должность]* | | |
| Со стороны исполнителя | *[ФИО, должность]* | | |
| **РЕЗУЛЬТАТЫ ОБСУЖДЕНИЯ** | | | |
| Пункт повестки/ вопрос | Результаты обсуждения | Ответственный | Сроки выполнения |
| . |  |  |  |
| . |  |  |  |
| . |  |  |  |
| **СТАТУС ПРОТОКОЛА** | | | |
| Согласовано | *[ФИО, должность]* | | |
| Утверждено | *[ФИО, должность]* | | |
| **ИНФОРМАЦИЯ О СЛЕДУЮЩЕЙ ВСТРЕЧЕ** | | | |
| Время/ Дата |  |  |  |
| Место |  |  |  |

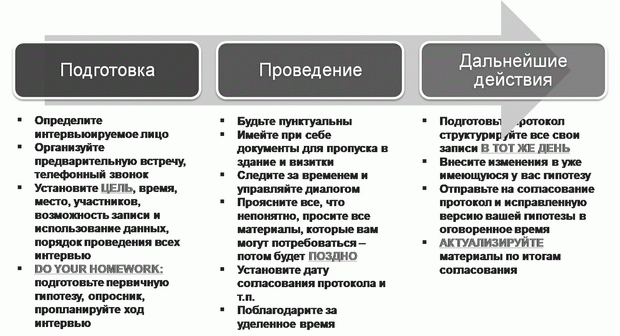
#### Использование функции качества

**Функция качества** - это инструмент для работы с заказчиком, который позволяет встроить его требования в проект. Цель этого инструмента - убедиться, что *требования заказчика* интегрированы в каждую часть проекта, от определения (1) требований проекта и (2) установления характеристик решения до формирования (3) проектных работ и выстраивания (4) *программы обеспечения качества*.

Процесс построения "дома качества" - предельно сложная процедура, особенно в случае крупных проектов. Тем не менее, этот инструмент довольно удобен в использовании и значительно повышает качество процесса управления требованиями проекта.

На [рис. 1.6](http://www.intuit.ru/studies/professional_retraining/963/courses/502/lecture/11390?page=4#image.1.6) отражена типовая структура "дома качества". Его заполнение производится в несколько этапов.

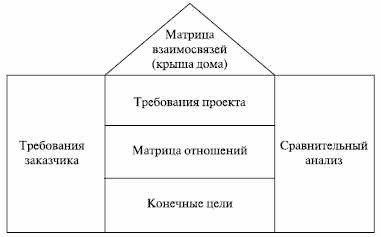
1. Подготовка требований заказчика

[](http://www.intuit.ru/EDI/29_06_15_4/1435529976-29105/tutorial/907/objects/1/files/01_05.gif)

**Рис. 1.5.** Схема и рекомендации по проведению интервью

Требования заказчика - важнейшая информация при построении "дома качества". Как правило, это самый сложный этап, поскольку необходимо выяснить наиболее значимые условия заказчика. Основной объем информации о его потребностях был выявлен в процессе интервьюирования и на этапе подготовке *ТЭО* и устава проекта. При разработке функции качества рекомендуется ограничивать функцию качества 10-ю требованиями заказчика, в противном случае использование инструмента становится громоздким.

1. Определение требований проекта



**Рис. 1.6.** Функция качества проекта ("домик" качества)

В отличие от требований заказчика, требования проекта сформулированы в терминах конкретных действий, при помощи которых команда планирует и реализует проект. Сформулированные таким образом требования проекта должны быть доступны для выполнения измерения и контроля достижения по завершению проекта. Следует различать два вида требований проекта: условия заказчика и предложения команды для их выполнения, как правило, содержащиеся в соответствующей методологии.

1. Формирование матрицы взаимосвязей

На этом этапе происходит проверка отсутствия взаимных противоречий между сформулированными требованиями проекта, иными словами, происходит их попарное сравнение. Для обозначения связи могут использоваться разные символы: так, для показа положительной связи часто используют o, а для показа отрицательной - o. Идентифицированные отношения демонстрируют столкновение требований и возможность нахождения компромисса между ними, что позволяет увидеть условия проекта в совокупности, а не по отдельности.

1. Формирование матрицы отношений

Заполнение матрицы отношений есть ключевой шаг построения "дома качества". Смысл ее заполнения состоит в том, чтобы убедиться, что все *требования заказчика* будут удовлетворены предложенными требованиями проекта. На пересечении соответствующего *требования заказчика* и требования проекта при наличии положительной связи ставится отметка, например, крестик. Если требование заказчика не поддерживается ни одним требованием проекта, значит, удовлетворение первого может вызвать ряд проблем. В обратной ситуации, когда проектное требование не соотносится ни с одним требованием заказчика, говорят об избыточности данного проектного требования. На крупных проектах иногда усложняют отношения между требованиями заказчика и проекта и вместо крестика используют числовые значения, характеризующие степень влияния требований проекта на реализацию заданного *требования заказчика* [[5](http://www.intuit.ru/studies/professional_retraining/963/courses/502/literature#literature.5)].

1. Субъективная оценка через сравнительный анализ

На данном этапе происходит присвоение степени важности каждому требованию заказчика и проект сравнивается с другими проектами и/или текущим status quo. Сравнение выявляет сильные и слабые стороны проекта по отношению к аналогичным инициативам, определяет возможности для улучшения.

1. Объективная оценка через установку конечных целей

Для обеспечения измерения и последующего контроля реализации требований проекта (а через них - и обеспечения требований заказчика) необходимо задать измеримые конечные цели по каждому требованию проекта, тем самым обеспечив объективную основу для управления ими.

Владимир Грекул, Нина Коровкина, Юрий Куприянов

Дополнительный материал 1. Матрица задач жизненного цикла ИС

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Этапы жизненного цикла информационных систем (ЖЦ ИС)** | | **Замысел (планирование проекта)** | | | **Принятие решений** |
| Стадия замысла выполняется для оценки новых возможностей в деловой сфере, разработки предварительных системных требований и осуществимых проектных решений. | | |
| **Этапы жизненного цикла проекта** | | **Планирование** | **Оценка** | **Контроль** |
| 1. | Управление интеграцией | Адаптация модели ЖЦ. Разработка *ТЭО* (бизнес-причина, выбор способа решения, эффекты по категориям). Формирование *бизнес-цели проекта*. Разработка устава проекта. Разработка*плана управления проектом* |  | Руководство и управление исполнением проекта. Осуществление интегрированного управления изменениями. Завершение стадии планирования проекта | Санкционирование перехода на стадию разработки. Документация всех принятых решений на данной стадии ЖЦ |
| 2. | Управление содержанием | Формирование требований проекта Формирование *ИСР*. Определение содержания проекта. Определение результатов всех стадий ЖЦ | Оценка реализуемости требований проекта | Подтверждение содержания проекта |  |
| 3. | Управление сроками | Формирование списка работ проекта. Определение последовательности работ проекта. Оценка трудоемкости и продолжительности работ. Базовое *расписание проекта*. Разработка расписания (детальное расписание стадии замысла и предварительное остальных стадий ЖЦ ИС) | Оценка реализуемости проектного расписания | Контроль *расписания проекта* | Принятие решение об адаптации модели ЖЦ |
| 4. | *Управление стоимостью* | Концептуальная оценка стоимости проекта. Формирование сметы стадии замысла | Уточнение оценки стоимости проекта | Уточнение и корректировка оценки стоимости проекта | Принятие решения о реализуемости проекта в рамках выделенной суммы |
| 5. | Управление качеством | Формирование критериев завершения стадии разработки и перехода к стадии разработки. Формирование критериев завершения стадии разработки. Определение результатов стадий в модели жизненного цикла системы. Формирование программы качества проекта. Формирование базовой линии требований проекта | Оценка соответствия полученных результатов критериям | Реализация программы качества | Принятие решения о соответствии качеству и корректировке результатов стадии или критериев |
| 6. | Управление рисками | Планирование управления рисками. Идентификация рисков проекта | *Качественный анализ рисков*. Количественный анализ рисков | Управление рисками стадии замысла | Документация реализованных рисков стадии замысла |
| 7. | Управление человеческими ресурсами | Планирование человеческих ресурсов Оценка доступности ресурсов. Формирование предварительного график потребности в человеческих ресурсах. Формирование предварительного описания должностных обязанностей членов проектной команды. Разработка организационной структуры проекта. Набор команды проекта | Оценка соответствия компетенций сотрудников стоя- щим перед ними задачам |  | Определение круга лиц, принимающих решение. Принятие решения о реализуемости проекта с точки зрения доступных человеческих ресурсов |
| 8. | Управление коммуникациями | Идентификация участников проекта. Планирование проектных коммуникаций | Оценка ожиданий участников проекта. Реализация *плана коммуникаций* | Получение обратной связи. Управление ожиданиями. Подготовка отчета | Определение стратегии принятия решений |
| 9. | Управление конфигурацией | Определение объектов управления конфигурацией. Планирование инфраструктуры стадии разработки. Установление базовой линии конфигурации проекта |  | Разработка документации по верификации и валидации проекта. Обеспечение доступа на протяжении всего жизненного цикла к статусу элементов конфигурации |  |
| **Этапы жизненного цикла информационных систем (ЖЦ ИС)** | | **Разработка (проектирование)** | | | **Принятие решений** |
| Стадия разработки осуществляется с целью создания такой рассматриваемой системы, которая удовлетворяет требованиям приобретающей стороны и может быть создана, испытана, оценена, применена по назначению, поддержана при применении и списана | | |
| **Этапы жизненного цикла проекта** | | **Планирование** | **Оценка** | **Контроль** |
| 1. | Управление интеграцией | Уточнение *плана управления проектом* | Мониторинг исполнения*плана управления проекта*. | Осуществление интегрированного управления изменениями. | Санкционирование перехода на стадию производства. |
| Оценка альтернатив развития проекта | Завершение фазы разработки | Документация всех принятых решений на данной стадии ЖЦ |
| 2. | Управление содержанием | Определение уточненных системных требований. Определение границ (технических, организационных, функциональных и т.д.) проекта | Оценка отклонений от базовой линии требований. Мониторинг содержания и объема проекта. Оценка готовности пользователей к работе в системе | Контроль содержания проекта. Планирование обучения конечных пользователей |  |
| 3. | Управление сроками | Разработка детальных планов стадии разработки и производства | Оценка достижения контрольных *вех проекта* | Контроль *расписания проекта* |  |
| 4. | *Управление стоимостью* | Формирование сметы проекта. Контрольная оценка стоимости проекта. Уточнение стоимостных оценок последующих стадий. Разработка бюджета оставшихся стадий проекта. Оценка ССВ | Оценка соответствия уточненным стоимостным оценкам | *Управление стоимостью*проекта |  |
| 5. | Управление качеством | Формирование критериев завершения стадии производства. Формирование критериев непригодности и списания систе-мы(документация по верификации и валидации). Уточнение базовой линии требований к системе. | Оценка соответствия установленным критериям качества | Управление требованиями проекта. Осуществление обеспечения качества | Оценка соответствия критериям перехода на следующую стадию. Подтверждение того, что система соответствует всем требованиям правообладате- |
|  |  | Уточнение результатов последующих стадий |  |  | лей и системным требованиям, может применяться по требованию |
| 6. | Управление рисками | Идентификация рисков для данной стадии | Оценка реализуемости рисков. Мониторинг рисков. Обновление журнала управления рисками | Управление рисками | Документация реализованных рисков стадии разработки |
| 7. | Управление человеческими ресурсами | Разработка базового плана потребности в человеческих ресурсах. Планирование обучения членов проектной команды | *Оценка адекватности*выделенных ресурсов решаемым задачам | Обучение и контроль качества произведенного обучения проектной команды |  |
| 8. | Управление коммуникациями | Уточнение архитектуры рассматриваемой системы, состоящей в т.ч. из людей и их интерфейсов | Оценка воздействия на участников проекта. Оценка отношения участников к реализуемому проекту | Сбор обратной связи участников проекта. Управление ожиданиями участников проекта. Обеспечение распространения информации о результатах проекта |  |
| 9. | Управление конфигурацией |  | Оценка соответствия базовой линии конфигурации | Контроль конфигурации выделенных элементов проекта. Обеспечение целостности элементов конфигурации | Утверждение и документация изменений в базовой линии конфигурации проекта |
| **Этапы жизненного цикла информационных систем (ЖЦ ИС)** | | **Производство (разработка и внедрение)** | | | **Принятие решений** |
| Цель стадии производства состоит в производстве или изготовлении продукта, испытании продукта и производстве соответствующих необходимых поддерживающих и обеспечивающих систем | | |
| **Этапы жизненного цикла проекта** | | **Планирование** | **Оценка** | **Контроль** |
| 1. | Управление интеграцией | Планирование закрытия проекта и перехода в стадию эксплуатации |  | Приемка проекта. Закрытие проекта | Санкционирование перехода на стадию применения. Документация всех принятых решений на данной стадии ЖЦ |
| 2. | Управление содержанием | Планирование тестирования и *перехода в продуктивную эксплуатацию*. Планирование концепции последующих стадий (план *перехода в продуктивную эксплуатацию*) | Оценка соответствия базовому плану по содержанию. Оценка возможности улучшения проекта | Управление требованиями проекта |  |
| 3. | Управление сроками | Планирования эксплуатации и утилизации системы |  | Контроль *расписания проекта*. Корректировка*расписания проекта* по итогам реализации двух первых стадий |  |
| 4. | *Управление стоимостью* | Уточнение стоимостных оценок последующих стадий. Уточнение ССВ внедряемой системы | Оценка соответствия уточненным стоимостным оценкам | *Управление стоимостью*проекта |  |
| 5. | Управление качеством | Определение критериев завершения стадий применения | Оценка соответствия системы критериям качества |  | Удовлетворение критериев завершения стадии производства |
| 6. | Управление рисками | Идентификация рисков данной стадии | Оценка реализуемости рисков, контроль | Управление рисками на данной стадии | Документация реализованных |
|  |  |  | статуса идентифицированных рисков. Мониторинг рисков. Обновление журнала управления рисками |  | рисков стадии производства |
| 7. | Управление человеческими ресурсами | Планирование потребности и квалификации ресурсов на последующей стадии. Планирование развития карьеры членов команды проекта внедрения | *Оценка адекватности*выделенных ресурсов решаемым задачам | Управление человеческими ресурсами | Принятия решение о дальнейшей работе членов проектной команды |
| 8. | Управление коммуникациями |  | Оценка удовлетворенности участников проекта результатами внедрения системы. Анализ возможности повышения эффективности взаимодействия с заказчиком за счет реализации обратной связи | Управление ожиданиями участников проекта. Обеспечение распространения информации о результатах проекта | Обеспечение сохранности проектной документации после окончания проекта |
| 9. | Управление конфигурацией | Подготовка инфраструктуры для стадии эксплуатации и поддержки | Оценка потребности реконфигурации инфраструктуры системы | Реконфигурация инфраструктуры проекта. Обеспечение целостности элементов конфигурации | Утверждение и документация изменений в базовой линии конфигурации проекта |
| **Этапы жизненного цикла информационных систем (ЖЦ ИС)** | | **Применение (эксплуатация и поддержка)** | | | **Принятие решений** |
| Стадия применения осуществляется для того, чтобы использовать продукт, предоставлять услуги в заданных условиях функционирования и гарантировать продолжительную результативность.Стадия поддержки применения осуществляется с целью осуществления материально-технического снабжения, технического обслуживания и текущего ремонта, которые обеспечивают непрерывное функционированиерассматри-ваемой системы и устойчивое предоставление услуг, поддерживающих ее применение | | |
| **Этапы жизненного цикла проекта** | | **Планирование** | **Оценка** | **Контроль** |
| 1. | Управление интеграцией |  |  |  | Санкционирование перехода на стадию утилизации |
| 2. | Управление содержанием |  | Выявление и оценка новых возможностей для развития системы |  |  |
| 3. | Управление сроками | Планирование стадии изъятия и списания |  |  |  |
| 4. | *Управление стоимостью* |  | Мониторинг ССВ системы с целью определения соответствия задачам использования | *Управление стоимостью*системы |  |
| 5. | Управление качеством |  | Оценка рабочих характеристик системы | Фиксирование удовлетворения критериев завершения данной стадии. Удовлетворение критерие в перехода на следующую стадию |  |
| 6. | Управление рисками |  | Оценка реализуемости рисков, контроль статуса идентифицированных рисков. Мониторинг рисков Обновление журнала управления рисками | Информирование о рисках всех заинтересованных сторон | Документация реализованных рисков стадии применения |
| 7. | Управление человеческими ресурсами | Планирование персонала (конечных пользователей) с необходимым уровнем компетенции | Оценка компетентности конечных пользователей |  |  |
| 8. | Управление коммуникациями |  |  | Контроль эффективности обратной связи. Обеспечение распространения информации о результатах проекта | Документация всех принятых решений на данной стадии ЖЦ |
| 9. | Управление конфигурацией |  |  |  |  |
| **Этапы жизненного цикла информационных систем (ЖЦ ИС)** | | **Изъятие и списание (утилизация и обновление)** | | | **Принятие решений** |
| Стадия изъятия и списания осуществляется с целью обеспечить удаление рассматриваемой системы и связанных с нею обслуживающих и поддерживающих служб из среды применения, непосредственно оперировать самой списываемой системой и поддерживать процесс ее изъятия и списания | | |
| **Этапы жизненного цикла проекта** | | **Планирование** | **Оценка** | **Контроль** |
| 1. | Управление интеграцией | Разработка *ТЭО* перехода на новую систему и отказа от текущей (бизнес-причина, выбор способа решения, эффекты по категориям) |  |  | Документация всех принятых решений на данной стадии ЖЦ |
| 2. | Управление содержанием |  |  |  |  |
| 3. | Управление сроками | Планирование передачи функций новой рассматриваемой системе |  |  |  |
| 4. | *Управление стоимостью* | Концептуальная оценка стоимости проекта по замене системы |  |  |  |
| 5. | Управление качеством |  |  |  | Констатирование удовлетворения критериям завершения стадии изъятия и списания и всего ЖЦ проекта |
| 6. | Управление рисками |  | Оценка реализуемости рисков, контроль статуса идентифицированных рисков. Мониторинг рисков. Обновление журнала управления рисками |  |  |
| 7. | Управление человеческими ресурсами |  |  |  |  |
| 8. | Управление коммуникациями | Планирование информирования о целях и задачах проекта апгрейда/миграции на новую систему |  |  |  |
| 9. | Управление конфигурацией |  |  |  |  |