МИНИСТЕРСТВО НАУКИ И
ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ВОЛГОГРАДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

А.В. Кизим

Управление проектами разработки систем искусственного интеллекта

Учебное пособие

Волгоград
2021

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЯ
ВЫСШЕГО ОБРАЗОВАНИЯ
ВОЛГОГРАДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

А.В. Кизим

Практикум по управлению проектами разработки систем.

Учебное пособие

Волгоград
2021

Оглавление

Введение

Данное пособие может быть использовано при изучении дисциплин «Управление проектами разработки систем», «Управление проектами разработки систем искусственного интеллекта», а также может быть применено для курсового и дипломного проектирования. Рекомендовано для студентов направлений подготовки: 09.04.01, 09.03.01 и подобных.

1. Жизненный цикл проекта и системы

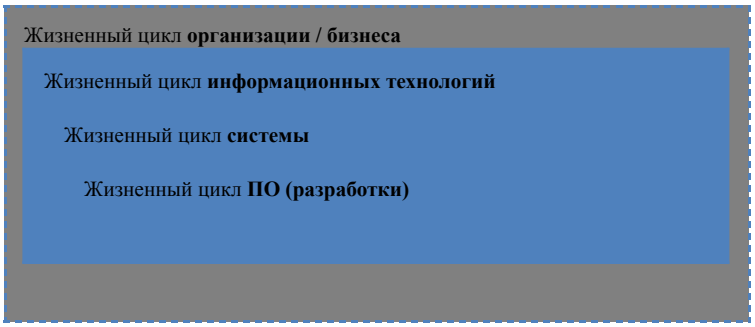
1.1. Жизненный цикл ИС и ПО

Показательно, что при рассмотрении основных вех истории информационных систем, в разные периоды фокус смещался с одного элемента ИС на другой. Так, первые десятилетия развивалось аппаратное, техническое обеспечение, затем со стремительным ростом масштабов разработки программных средств потребовалось срочно принимать необходимые организационные меры, чтобы обеспечить соответствие квалификации и компетенций специалистов минимальным требованиям к пользователям систем. Соответственно, различные элементы информационных систем развивались по разным сценариям, а значит, они имеют и свою специфику жизненных циклов?

Наиболее часто ключевые стороны, задействованные в создании и эксплуатации информационной системы, ассоциируют ее с программным обеспечением, лежащем в основе этой ИС. Тогда в чем основное отличие жизненного цикла информационной системы от жизненного цикла программного обеспечения?

Ответ начинается с самого определения информационной системы, охватывающего большее количество аспектов, нежели просто ПО. Программное обеспечение может быть модернизировано, обновлено до новых версий несколько раз в рамках одного жизненного цикла информационной системы. Однако это будет все та же информационная система, возможно, с теми же инфраструктурными и организационными ресурсами. Могут приходить и уходить сотрудники, обновляться системы хранения данных, но это лишь части информационной системы, которая все равно будет выполнять свои основные функции до момента вывода из эксплуатации.

Интересная для рассмотрения концепция уровней жизненного цикла была предложена в 2005 году Скоттом Амблером, автором практик гибкого моделирования Agile Modeling и концепции Enterprise Unified Process.



Логика рассуждений в данном случае следующая: жизненный цикл бизнеса включает всю деятельность ИТ-департамента, в том числе по разработке, развертыванию, поддержке и сопровождению информационных систем, частью которых является программное обеспечение. Но для того, чтобы проверить эту концепцию, необходимо рассмотреть основные аспекты жизненного цикла ИС подробнее, фокусируясь на основных ее стадиях и их содержании.

Как уже было сказано, **жизненный цикл** информационных систем представляет собой непрерывный процесс из определенных этапов, начинающийся в момент принятия решения о необходимости создания ИС до отказа от ее использования.

Жизненный цикл информационной системы – непрерывный процесс, началом которого становится момент принятия решения о необходимости системы, а завершением – ее изъятие из эксплуатации. Этапы создания системы до момента ввода в эксплуатацию могут рассматриваться как самостоятельные проекты, каждый из которых имеет конкретный результат и ограничения.

А так как процесс создания и конфигурирования для различных информационных систем включает в себя один набор этапов, то можно говорить о **моделях жизненного цикла.**

Модель ЖЦ ИС – комбинация последовательности этапов жизненного цикла и переходов между ними, необходимых для гарантированного достижения поставленной для реализации проекта цели.

Сами фазы жизненного цикла фиксированы и для различных отраслей человеческой деятельности, по сути, одинаковы:

– Замысел (планирование проекта).
– Анализ и постановка задачи.
– Проектирование.
– Разработка.
– Развертывание и внедрение.
– Эксплуатация.
– Поддержка.
– Модернизация.
– Утилизация.

Можно говорить о том, что средняя продолжительность подобного цикла составляет порядка 15 лет, однако следует учитывать что в зависимости от огромного числа различных факторов, обусловленных спецификой предприятия, отрасли, самой информационной системы сроки физического и морального старения техники и ПО будут значительно отличаться. А значит, еще при проектировании ИС необходимо четко представлять себе возможности ее дальнейшей модернизации, в том числе факторы, которые могут вызвать эту необходимость.

Важно понимать, что переход на новые программные решения – это не утилизация старой и проектирование новой системы. Измениться может практически все: форматы данных, работающий с системой персонал, поддерживающая инфраструктура. Остается только информация, и именно она является связующим звеном, позволяющим говорить о работе с одной и той же информационной системой. Если в системе последние десять лет хранились и обрабатывались данные по незавершенному производству, то могут приниматься решения о переходе на другую платформу, выбор решения другого вендора, для чего проведена конвертация данных, но она все равно останется информационной системой работы с производственными данными.

Порядка трети ИС прекращают свое существование еще на этапе проектирования, причиной чего часто становится несоответствие методов управления проектами (в том числе, анализа) сложности самого проекта, которая постоянно возрастает в условиях рыночной экономики, где число стейкхолдеров постоянно увеличивается, ограничения по срокам и стоимости возрастают, появляются новые конкурирующие компании и продукты. Не всегда уровень выбираемых компаниями технологий анализа / проектирования систем, как и методик управления проектом внедрения соответствует специфике бизнеса и возрастающим требованиям к автоматизации процессов.

Рассмотрим основные модели жизненного цикла, позволяющие при определении правил и условий перехода на следующую стадию нивелировать риски и оптимизировать процесс создания и передачи системы в эксплуатацию.

2. Модели ЖЦИС

2.1. Виды моделей ЖЦИС

2.1.1. Каскадная модель

Каскадная модель жизненного цикла, также называемая моделью «водопада» (waterfall), была разработана еще в 80-х годах, и на протяжении многих лет она считалась стандартом для разработки ПО. Данная модель характеризуется тем, что этапы строго последовательны и переход между ними невозвратный. Это означает, что в рамках каскадной модели переход к следующему этапу (например, от проектирования и сбора требований к разработке и развертыванию) может произойти только по завершении предыдущего этапа. Модель «водопада» была применена одной из первых и одно из ее основных достоинств в возможности планирования сроков и стоимости каждого этапа – однако, на практике разработка системы почти никогда не проходит строго в соответствии с жесткой заранее продуманной схемой. В частности, это касается сбора требований, так как реально при старте проекта требования бывают определены только частично и в дальнейшем уточняются, изменяются и

дополняются. К тому же, если изначально требования были определены неточно, высока вероятность того, что система не будет удовлетворять потребностям заказчика.

2.1.2. Каскадная модель с промежуточным контролем

В качестве одной из вариаций каскадной модели для того, чтобы предусмотреть возможность возвращения к предыдущим этапам для внесения определенных изменений и пересмотра отдельных вопросов, была создана **каскадная модель с промежуточным контролем**.

Она предполагает увеличенное время, отведенное на разработку, за счет проведения промежуточных корректировок между фазами жизненного цикла. В свою очередь, это снижает риски получения некачественного продукта на выходе и повышает надежность системы в целом.

Важно отметить, что согласование результатов в двух описанных моделях происходит только по окончании внедрения – а значит, повышается вероятность получения программного продукта, который морально устареет либо не будет востребован рынком. Еще больше увеличивают риски возможные неточности в исходном техническом задании. В итоге можно говорить о проблеме определенной задержки в получении результата, которая не может быть решена в «каскадном» варианте разработки и внедрения системы.

2.1.3. Спиральная модель

Для нивелирования рисков, связанных с вышеописанной проблемой, была создана **спиральная модель** (еще называемая **итерационной**). Фазы жизненного цикла непоследовательны, то есть допустимо (но не обязательно!) начало работ над следующим этапом до завершения предыдущего. Таким образом, суть спиральной модели состоит в возможности прохождения всех этапов жизненного цикла системы в несколько итераций, каждый раз создавая новый прототип и проверяя актуальность требований, по которым он создавался, внося технические

доработки в интерфейс и функциональность. Подобная гибкость позволяет использовать модель на предприятиях любого масштаба.

Прототип крайне важен в ситуациях, когда требуется разъяснить и уточнить требования, выбрать концептуальное решение или даже в целом определить целесообразность реализации проекта. При этом сам прототип может либо моделировать исключительно пользовательский интерфейс, либо архитектурную сторону системы (логика обработки и хранения данных).

Это означает, что процесс создания системы и само управление проектом будет более гибким и управляемым, с совершенствованием системы на каждом «витке спирали», то есть при выпуске каждой версии. Уменьшаются риски (в том числе, финансовые) для заказчика и спонсора системы, которые могут отказаться от проекта еще на этапе показа первого прототипа в случае абсолютного его несоответствия ожиданиям и потребностям (либо в случае изменения рыночной ситуации).

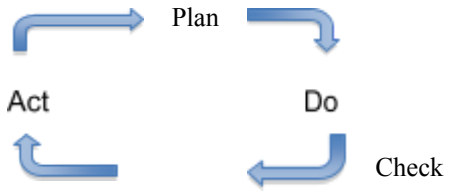
Как правило, подобная модель применяется при разработке нетиповых систем, предоставляя возможность оперативно создать прототип программного продукта для проверки его работоспособности с пользователями и соответственно, быстрого получения комментариев и замечаний к системе.

Цикл Шухарта-Деминга

Определенным прообразом спиральной модели ЖЦ стал классический цикл управления Деминга (PDCA-цикл).

PDCA (Plan – Do – Check – Act) – Цикл качества, еще называемый циклом Деминга. Этот алгоритм управления предполагает четыре основных шага «Планирование – Действие – Проверка – Корректировка» и помимо стандартной области применения в менеджменте предприятия возможно использовать его в управлении внедрением и развитием информационной системы. Таким образом, этап планирования может начинаться как при первичном проектировании системы, так и при старте ее модернизации, когда, как мы говорили, снова необходимо пройти все этапы жизненного цикла системы.

Графически цикл Деминга может быть представлен следующим образом:



При планировании появляется возможность совершенствования процессов и операций, определения и выделения необходимых для них ресурсов. По окончании следующего этапа (выполнения запланированных работ) обязательно проводится мониторинг результатов и сбор данных для аналитики и определения соответствия плановым показателям для принятия дальнейших мер по корректировке проводимых активностей и принципов распределения ресурсов – что, как мы видели, и является основной идеей итерационной модели создания ИС.

Итерационную / спиральную модель невозможно применять для областей, в которых невозможно предварительное тестирование продукта, обладающего неполной функциональностью – в частности, военные разработки, авиастроение, атомная энергетика, ведь любая ошибка стоит жизни. Однако при разработке бизнес-ориентированной информационной системы собственными силами ИТ-подразделения компании данная модель применяется достаточно часто, хотя важно принимать во внимание возможное сопротивление со стороны пользователей, которые хотят видеть требующуюся именно им функциональность в первом же релизе, а не втором

или третьем. Соответственно, на первый план выходит необходимость грамотного управления ожиданиями пользователей.

2.1.4. Модель разработки через тестирование (V-модель)

Приближенная по своей сути к практикам PRINCE2, **V-модель разработки через тестирование** была разработана еще в конце 1980-х годов ведомствами Германии и США, и до сих пор является стандартом немецких правительственных и оборонных проектов. Основной ее принцип состоит в постепенном возрастании степени детализации проекта с течением времени и одновременном проведении «горизонтальных» итераций. Таким образом, результаты каждой из фаз левой стороны буквы V влияют на тестирование и компоновку проекта с правой стороны буквы V: приемо-сдаточные испытания основываются на проведенном анализе требований, интеграционное тестирование – на высокоуровневом описании архитектуры, модульное тестирование – на архитектуре, интерфейсах, алгоритмах и прочих элементах детализированных требованиях к системе.

Важна гибкость данной модели, так как по сути она адаптируема под любой тип организации. Промежуточные результаты проверяются на ранних стадиях и минимизация рисков достигается благодаря простому соотнесению фаз / итераций. V-модель не рассматривает непосредственно стадию обслуживания и утилизации, учитывая лишь активности по подготовке к ним.

Рассмотрев основные особенности моделей жизненного цикла, перейдем к фазам **проекта по созданию и внедрению** интегрированной **системы управления** как совокупности этапов по созданию, настройке, доработке и внедрению отдельных функциональных модулей системы, выполнение которых необходимо и достаточно для создания системы управления, соответствующей заданным требованиям.

2.2. Стандарты жизненного цикла ИС

2.2.1. ГОСТ 34.601-90

ГОСТ 34.601-90 Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания.

Отличается высокой степенью формализации (как и ГОСТы 19-й серии) и по умолчанию, таким образом предполагает каскадный подход. На сегодняшний день ГОСТ многократно становился основой для доработок и частичного использования в других стандартах / методологиях и в целом в исходном виде не является исчерпывающим как единственный источник информации для выполнения проекта разработки / внедрения.

Стадии	Этапы
Формирование требований к АС	1.1. Обследование объекта и обоснование необходимости создания АС 1.2. Формирование требований пользователя к АС 1.3. Оформление отчета о выполненной работе и заявки на разработку АС (тактико-технического задания)

Разработка концепции АС	2.1. Изучение объекта 2.2. Проведение необходимых научно-исследовательских работ 2.3. Разработка вариантов концепции АС, удовлетворяющего требованиям пользователя 2.4. Оформление отчета о выполненной работе
Техническое задание	3.1 Разработка и утверждение технического задания на создание АС
Эскизный проект	4.1. Разработка предварительных проектных решений по системе и ее частям 4.2. Разработка документации на АС и ее части
Технический проект	5.1. Разработка проектных решений по системе и ее частям 5.2. Разработка документации на АС и ее части 5.3. Разработка и оформление документации на поставку изделий для комплектования АС и (или) технических требований (технических заданий) на их разработку 5.4. Разработка заданий на проектирование в смежных частях проекта объекта автоматизации
Стадии	Этапы
Рабочая документация	6.1. Разработка рабочей документации на систему и ее части 6.2. Разработка или адаптация программ
Ввод в действие	7.1. Подготовка объекта автоматизации к вводу АС в действие 7.2. Подготовка персонала 7.3. Комплектация АС поставляемыми изделиями (программными и техническими средствами, программно-техническими комплексами, информационными изделиями) 7.4. Строительно-монтажные работы 7.5. Пусконаладочные работы 7.6. Проведение предварительных испытаний 7.7. Проведение опытной эксплуатации 7.8. Проведение приемочных испытаний
Сопровождение АС	8.1. Выполнение работ в соответствии с гарантийными обязательствами 8.2. Послегарантийное обслуживание

Благодаря своей структурированности, ГОСТ 34.601-90 до сих пор служит самодостаточной базой для адаптации, которую можно адаптировать к конкретным условиям деятельности предприятия. Приложение к стандарту содержит детальное описание работ, включая списки формируемых по завершении этапа документов.

ПРИМЕР

На этапе 7.6 «Проведение предварительных испытаний» осуществляют:

а) Испытания АС на работоспособность и соответствие техническому заданию в соответствии с программой и методикой предварительных испытаний;

б) Устранение неисправностей и внесение изменений в документацию на АС, в том числе эксплуатационную в соответствии с протоколом испытаний;

в) Оформление акта о приемке АС в опытную эксплуатацию.

Также в стандарте приведен список основных типов организаций, участвующих в работах по созданию АС, что позволяет сформировать понимание сути процесса.

ПРИМЕР

Организация-заказчик (пользователь), разработчик, поставщик, генпроектировщик, …

2.2.2. ISO/IEC 12207:2008 (ГОСТ Р ИСО/МЭК 12207-2010)

Международный стандарт: **ISO/IEC 12207:2008 Information technology – Software life cycle processes (Информационные технологии. Процессы жизненного цикла программного обеспечения).**

Российский аналог: **ГОСТ Р ИСО/МЭК 12207-2010 «Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств».**

Базируясь на процессном подходе, ISO 12207 определяет необходимость документирования основных результатов процесса, но не ограничивает их содержание и тем более последовательность, а также не противоречит применению итераций в разработке. Данный стандарт стал основой для дальнейшей детализации в некоторых методологиях разработки ПО (в частности, Rational Unified Process), однако сам по себе лишь устанавливает

структуру основных, вспомогательных и организационных процессов ЖЦ **программных средств**, определяя необходимые в их рамках работы и задачи. Таким образом формируется единое понимание жизненного цикла (и единая терминология) между заказчиком, разработчиком / подрядчиком и другими стейкхолдерами. С другой стороны, ISO 12207:2008 рассматривает лишь программные средства и соответствующие организационные процессы, не рассматривая аппаратную составляющую.

В РФ был разработан и принят идентичный ISO 12207 стандарт **ГОСТ Р ИСО/МЭК 12207-2010.** Основной идеей разработчиков ГОСТ 12207 являлось создание единого общекорпоративного стандарта, которым было бы возможно воспользоваться при возникновении любой задачи из тех, которые описаны в документе (будь это обучение пользователей, поставка ПО или любая другая активность в рамках ЖЦ).

Стандарт предполагает, что процессы состоят из работ, для которых определены задачи (а также цели и результаты). Тем не менее, допускается адаптация процессов к особенностям организации (например, при больших масштабах проекта изменять состав определенных задач или работ). Как правило, это возможно сделать в рамках существующих на предприятии процессов.

ПРИМЕР

Говоря о процессе поставки ПО (п. 6.1.2 стандарта), предполагаются следующие виды работ:
6.1.2.3.1 Идентификация возможностей
6.1.2.3.2 Предоставление заявки поставщикам
6.1.2.3.3 Согласование проекта
6.1.2.3.4 Выполнение контракта
6.1.2.3.5 Поставка и поддержка продукта (услуги)
6.1.2.3.6 Закрытие
Несомненно, что у организации – заказчика существуют свои корпоративные регламенты проведения закупок (например, согласования заявок, формирования годовой программы закупок по подразделению ИТ, определения критериев выбора поставщиков услуг и т.д.). Соответственно, компания будет осуществлять приведенные в ГОСТ активности, но на операционном уровне она будет это делать в соответствии со своими внутренними регламентами и процессами.

Приложения стандарта содержат (помимо эталонной модели процессов, их описаний и видов, а также истории разработки) отдельно выделенный процесс адаптации. Приведенные в нем рекомендации по переходу от стандарта к реалиям определенного предприятия в основном концентрируются на выборе из всего приведенного множества процессов тех работ, которые необходимы для реализации конкретного программного проекта. Однако практические рекомендации по организации внедрения ГОСТ 12207-2010 остаются за границами самого документа.

2.2.3. ISO/IEC 15288 (ГОСТ Р ИСО/МЭК 15288-2005)

Международный стандарт: **ISO/IEC 15288:2005 Systems engineering. System life cycle processes (Системотехника. Процессы жизненного цикла системы).**

Российский аналог: **ГОСТ Р ИСО/МЭК 15288-2005 Информационная технология. Системная инженерия. Процессы жизненного цикла систем.**

Достаточно «молодой» стандарт системной инженерии (впервые представленный в 2002 году), ISO/IEC 15288 фокусируется на вопросах жизненного цикла системного уровня, в особенности тейлоринге (tailoring) – по сути, настройке и адаптации ЖЦ к конкретным требованиям и ограничениям.

В отличие от рассмотренного ранее стандарта, ISO 15288 распространяется на системы в целом, охватывая такие их элементы, как: «технические средства, программные средства, люди, процессы (например, процесс

оценки), процедуры (например, инструкции оператора), основные средства и природные ресурсы (например, вода, объекты живой природы, минералы)» [50, с. 4]. Согласно данному стандарту, любой процесс ЖЦ может быть начат в любой момент, без ограничения порядка использования и последовательности (в том числе, параллельном выполнении нескольких процессов).

Важно также отметить высокий уровень абстракции ISO 15288 в сравнении с ISO 12207, так как данный стандарт не приводит ролей, конечных результатов в виде списка выходных документов, либо же состава работ, лишь оставаясь на уровне концепции.

ПРИМЕР

В процессе управления ресурсами (п. 5.3.5 Стандарта) в качестве основных пунктов деятельности приводятся следующие:

a) определять и обеспечивать поддержку инфраструктуры ресурсов, необходимой для выполнения организацией требований настоящего стандарта и осуществления поддержки проекта;

b) получать ресурсы, за исключением персонала, необходимые для внедрения и осуществления проектов;

c) проявлять заботу о персонале, занятом в осуществлении текущих проектов;

d) стимулировать персонал, например, посредством предоставления возможности карьерного роста или при помощи системы поощрений;

e) контролировать области взаимодействия нескольких проектов для разрешения связанных с графиками их реализации конфликтов.

Очевидно, что формулировки «… обеспечивать поддержку инфраструктуры ресурсов…» или «проявлять заботу о персонале…» не являются очень конкретными и могут толковаться по-разному. В связи с этим, важно иметь поддержку высшего руководства для использования данного стандарта в проекте создания и эксплуатации системы. При этом особенность ISO 15288 в том, что использоваться он может как со стороны заказчика, так и со стороны исполнителя.

Стандарт содержит четыре основные группы процессов (предприятия, соглашения, проекта и технические), описывающие соответственно вспомогательные корпоративные процессы, взаимодействие с контрагентами, управление проектом и саму реализацию системы.

Важной положительной чертой стандарта является его связь с бизнессстороной проекта создания системы за счет групп процессов предприятия и соглашения. Благодаря наличию подобных разделов стандарта появляется связь с соответствующими корпоративными функциями и для бизнеса становится более понятным место процессов ЖЦ в процессах организации в целом.

2.3. Системный подход

Вне зависимости от того, рассматривается ли проект управления внедрения ИС или речь идет об управлении всей организацией в целом, важно иметь общее системное представление о ее деятельности, раскрывая так называемый «черный ящик».

«**Системный подход** является отдельным направлением методологии научного познания и социальной практики, в основе которого лежит рассмотрение объектов как систем. Это ориентирует исследование на раскрытие целостности объекта, выявление многообразных типов связей в нем и сведение их в единую теоретическую картину».

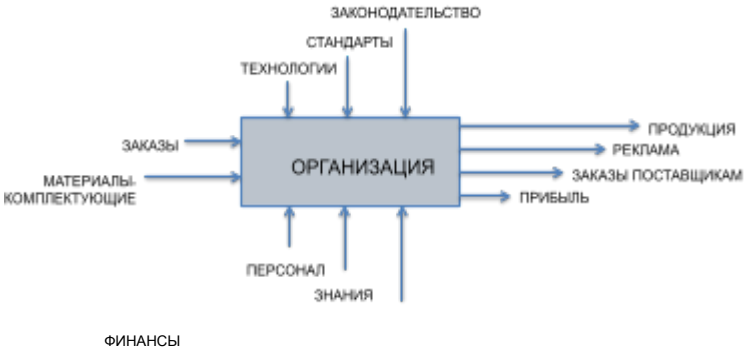
Важным свойством, характеризующим системный анализ, является эмержентность, возрастание вероятности отличия свойств целого от свойств его компонентов по мере усложнения системы. И действительно, чем крупнее организация (которую мы рассматриваем как систему), тем больше зависимости между ее элементами влияют на общее направление ее развития.

ГОСТ Р ИСО/МЭК 15288-2005, подробно рассматриваемый в разделе стандартов и посвященный процессам ЖЦ систем, не только представляет из себя хороший пример единого системного подхода, рассматривая процессы соглашения, предприятия, проекта и технические процессы с точки зрения предприятия, а не разработчиков, но и сам в приложении содержит крайне наглядные представления понятий «системы» и «среды использования». Для успешного применения подобного системного подхода необходимо уметь рассматривать системы не только «изнутри» или «снаружи», но и «сбоку».

ПРИМЕР

Как мы уже говорили, информационная система может представляться с точки зрения структуры (набор программных модулей, аппаратного и организационного обеспечения) или, к примеру, с точки зрения функциональности (те функции и возможности, которые она предоставляет).

В качестве одного из результатов системного подхода к рассмотрению организации может являться моделирование ее связей с другими элементами внешней среды.



Подобная схема достаточно наглядно иллюстрирует взаимодействие компании с бизнес-системой и позволяет удостовериться, что ничего не ускользнуло из зоны внимания, и все ресурсы, ограничения, инструменты и результаты деятельности учитываются при дальнейшем структурном рассмотрении активностей предприятия. Такой анализ упрощается при моделировании и построении графических описаний, которые могут применяться как внутри организации, так и аналитиками консалтинговых / подрядных организаций для ознакомления с общей картиной бизнес-процессов.

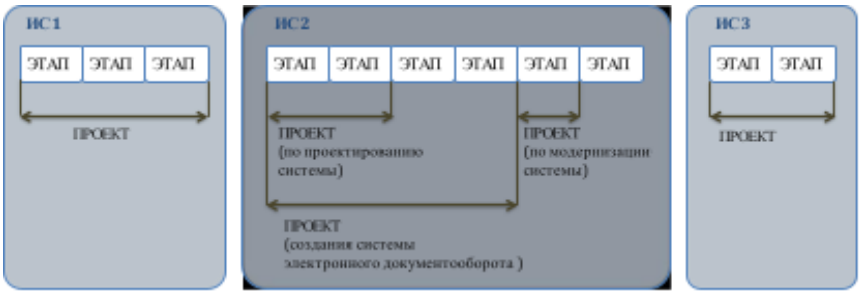
«**Структурный подход** представляет собой прохождение нескольких этапов детализации системы «сверху вниз», описывая таким образом все структурные единицы (подсистемы) компании».

Умение перемещаться по уровням абстракции и видеть иерархию, структуру для выделения общих свойств и зависимостей крайне важно для того, чтобы воспринимать систему как упорядоченный набор функций, взаимодействующий с окружающей средой. Имея высокоуровневое представление о структуре и связях, а также целях системы, можно переходить уже к дальнейшей детализации – а значит, структурному подходу.

Структура системы формируется в иерархическом виде.

Так, объект любого уровня может рассматриваться в качестве системы. Именно поэтому представляется возможным в рамках создания системы рассматривать реализацию отдельных элементов (например, модулей) в рамках этапов жизненного цикла с точки зрения отдельных проектов, а создание единого решения (информационной системы) от концепции до ввода в эксплуатацию – как целую программу проектов. Планирование подобной программы, как правило, осуществляется «сверху вниз», в силу необходимости распределения временных, денежных и прочих ресурсов. Критическим фактором успеха является логическая связь этапов / проектов (определяемая выбранной моделью жизненного цикла). На схеме ниже приведен пример для каскадной модели.

Однако далее в тексте пособия термин «**проект**» используется и в другом значении – как ограниченная рамками договора с исполнителем деятельность, так как на протяжении ЖЦ ИС компания взаимодействует со множеством контрагентов.



Таким образом, совместные с подрядчиками / вендорами активности тоже могут рассматриваться как полноценные проекты, только уже не ограниченные рамками одного этапа ЖЦ, а охватывая либо несколько фаз, либо с реализацией нескольких проектов в течение одной фазы. Тогда, как правило, заключаются отдельные договора на разные проведение информационного обследования и формирование требований в рамках этапа анализа и постановки задачи.

3 Особенности управления проектами по внедрению КИС

Управление жизненным циклом информационной системы – это управление в условиях постоянно изменяющихся требований со стороны бизнеса, в условиях развивающейся функциональности системы, в условиях постоянного развития новых информационных технологий.

Работа КИС касается деятельности различных служб и должностных лиц предприятия. Однако отношения между ними могут быть более чем непростыми. Информатизация всегда вызывает слухи о сокращении штатов и

повышении норм по выработке. Более того, многие специалисты стремятся сохранить свой объем деятельности от постороннего вмешательства, а потому возможны попытки прямого (или скрытого) саботажа нововведений на предприятии. В редких случаях может потребоваться привлечение психологов для работы с коллективом.

К тому же, к сожалению, наблюдается тенденция к «внедрению ради внедрения», когда важен даже не результат, а сам факт работы с ПО определенного масштаба или определенного производителя. В таком случае «заниженная планка» качества / возможностей может устраивать заинтересованные стороны, но важно донести выбранный подход до всех ключевых участников проекта и грамотно подойти к решению вопроса о целях проекта, наличии ограничений, возможностях экономии времени / ресурсов, снижении объема проекта.

Невозможно не отметить, что создание КИС вызывает ряд сложных организационных проблем. Для создание КИС ИТ-службе выгоднее всего привлечь какую-либо стороннюю организацию. Однако ИТ-службе в любом случае придется отвечать за их работу. Попытка обойтись своими силами чревата срывом сроков, а также острым дефицитом ресурсов при обилии текущих задач.

Зачастую профессиональные консультанты, нанятые «на стороне», могут беспристрастно и объективно оценить используемые на предприятии информационные технологии. Консультанты могут разграничить свою сферу деятельности и сферу деятельности ИТ-службы компании-нанимателя. В штате предприятия, тем не менее, должны находиться сотрудники, способные квалифицированно понять и проанализировать действия нанятых консультантов, поэтому предприятию не обойтись без собственных специалистов. ИТ-службы должны контролировать и координировать работы по созданию КИС, начиная с самого первого этапа планирования и обследования – так как только они являются представителями интересов бизнес-заказчика на предприятии и могут обеспечить непредвзятость расчетов финансовых и временных ресурсов для проекта.

Процедура стратегического планирования должна стать основанием построения КИС на предприятии. Следом за этим необходимо определить основные бизнес-процессы предприятия и выделить информационные структуры, обеспечивающие их. Только в этом случае созданная информационная система сможет стать базой для работы и развития предприятия.

Следует учитывать и изменяющиеся условия бизнеса, иначе КИС может устареть прежде, чем будет завершена. Безусловно, управлять глобальным проектом вроде создания КИС следует с точки зрения финансового менеджмента, однако традиционное планирование становится неадекватным в условиях современного динамичного рынка. Однако полный отказ от планирования также рискует привести к серьезным проблемам. Более того, создавать КИС по частям можно только при наличии единого системного проекта, иначе потери могут превысить ожидаемую полезность. Именно поэтому в настоящее время все больше компаний в сотрудничестве с консалтинговыми компаниями (такими, как Accenture, IBM и BearingPoint) организуют обновление или же создание с нуля стратегии развития ИС. Как правило, подобные проекты достаточно стандартны в плане фаз и включают в себя три основные стадии:

I. Анализ текущего состояния ИТ на предприятии (анализ As-Is).

II. Разработка целевого плана ИТ (формирование целевой модели To-Be).

III. Разработка плана перехода от текущего состояния к целевому, еще называемого «дорожной картой» (roadmap).

ИТ-стратегия – план управления ИТ стратегического уровня, определяющий основные направления развития информационных технологий для оптимального их использования и достижения поставленных компанией бизнес-целей.

Причины, по которым компании принимают решение разрабатывать ИТ-стратегию, могут быть самыми разными.

ПРИМЕР

– Изменение стратегии бизнеса требует пересмотра ИТ стратегии.

– Изменения организационной структуры (новое руководство ИТ-департамента, слияние нескольких компаний и их ИТ-функций).

– Предыдущая концепция развития ИТ требует пересмотра в силу своей неэффективности, превышения запланированных затрат, нарушения графика реализации.

– Необходимо сформировать механизм взаимодействия бизнеса и ИТ в соответствии со стратегическими задачами компании.

– Необходимо обосновать эффективность долгосрочных ИТ-инвестиций.

Соответственно, определенную роль играют и внутренние факторы (операционный менеджмент, оргструктура), и внешние (например, M&A). В зависимости от целей формирования стратегии определяется, в течение какого периода она будет использоваться (кратко-, средне-, долгосрочный).

В итоге подобных проектов по формированию стратегии могут создаваться следующие результирующие документы:

- Целевые мероприятия и портфель проектов ИТ (в том числе планирование инвестиций в ИТ, паспорта проектов ИТ, балансировка портфеля ИТ, оценка бюджетов проектов ИТ).
- Система управления проектной деятельностью (в том числе организационно-методологические принципы и стандарты управления проектами).
- Каталог услуг ИТ.
- Модель сорсинга ИТ.
- Организационная структура ИТ.
- Модель ИТ-процессов.

А в ходе непосредственно разработки также создаются как целевое видение: ландшафт бизнес-процессов, приложений, данных и ИТ-инфраструктуры, которые могут использоваться при дальнейшем построении систем на предприятии.

Другим значимым типом проектов, способствующих эффективному построению ИС на предприятии, является ИТ-аудит.

ИТ-аудит – системный процесс получения, оценки и предоставлению руководству компании объективных данных о текущем состоянии ИТ-инфраструктуры, информационных систем, бизнес-процессов организации, действиях и событиях, происходящих в сфере ИТ, устанавливающий уровень их соответствия определенным критериям и предоставляющий результаты заказчику.

Его результаты могут использоваться для планирования дальнейшей деятельности компании в сфере ИТ и ее оптимизации. Решения могут затем приниматься с учетом того, насколько эффективно ИТ-подразделения используют имеющиеся ресурсы для решения задач компании; насколько ИТдеятельность соответствует предъявляющимся требованиям как со стороны организации, так и со стороны внешних контрагентов и регуляторов; насколько надежно организована работа ИТ-систем и обеспечение безопасности данных.

В ходе ИТ-аудита руководству компании предоставляются объективные данные о текущем состоянии ИТ-инфраструктуры, информационных систем, бизнес-процессов организации, действиях и событиях, происходящих в сфере ИТ. В частности, может проводиться аудит систем в процессе разработки, в ходе которого определяется соответствие процессов проектирования и разработки систем утвержденным регламентам и стандартам уровня организации / отрасли / мировых практик. Оценка эффективности управления проектом, анализа выгод и рисков, сбора требований, анализа запросов на изменение (в том числе при использовании данных CMDB), процедур отката изменений и управления исходным кодом, процессов разработки системы контролей.

В процессе внедрения также важен аудит инфраструктуры ИТ, проводящий оценку размещения, состава и состояния инфраструктурных активов организации, в том числе инженерных систем (бесперебойное электропитание, кондиционирование), сетевого обеспечения (LAN), вычислительных мощностей (программно-аппаратные платформы, хранение данных и резервное копирование).

Внедрение на предприятии информационной системы требует значительных (и не только финансовых) ресурсов. Эффективное внедрение КИС может стать только результатов командной работы, причем в команду должны входить представители разработчика и заказчика. Необходима сторонняя критика людей аналитического и синтетического складов ума. Подробнее подход к формированию проектной команды будет рассмотрен в соответствующем разделе «Управление человеческими ресурсами».

Но не следует также забывать, что далеко не последнюю (и даже главную) роль в процессе внедрения информационной системы играет пользователь. Пусть он не участвует в активностях по внедрению на регулярной основе, но именно для него создается система, именно он будет ее использовать в своей деятельности. А значит, именно на пользователей должен быть направлен фокус в проекте, именно их требования необходимо учитывать для достижения максимального эффекта от внедрения системы и получении ожидаемых результатов при ее эксплуатации.

3.1. Создание ИС в соответствии с методологиями и стандартами

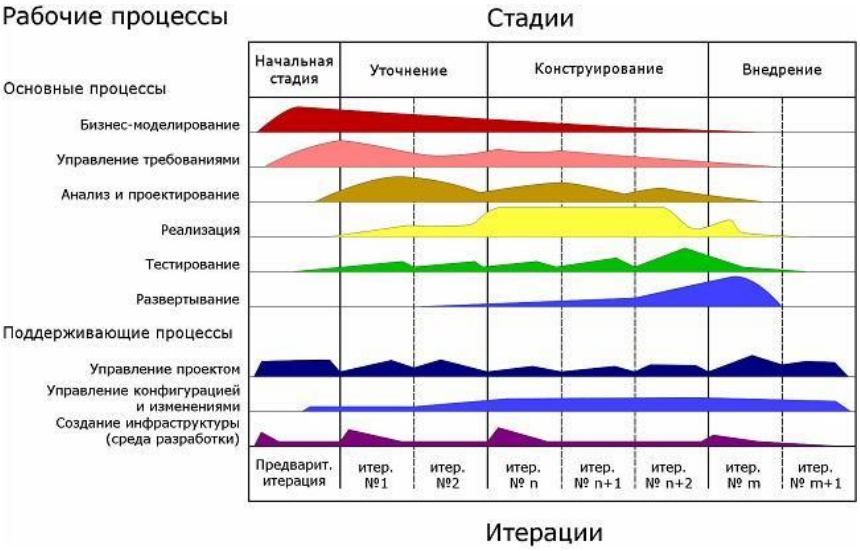
3.1.1. Корпоративные методологии

3.1.1.1. IBM (Rational Unified Process, RUP)

Один из наиболее ориентированных на пользователя подходов, RUP (Rational Unified Process) предполагает итеративную разработку, большое внимание к архитектуре, к потенциальным рискам, и главное – к управлению на основе пользовательских «юзкейсов» (use-cases).

«**Юзкейсы** определяют соглашение между стейкхолдерами системы относительно ее поведения. Юзкейс описывает поведение системы при различных условиях в ответ на запрос со стороны одного из стейкхолдеров» (Коберн).

По сути юзкейсы описывают варианты использования системы и ее действия по достижению поставленных целей. Стейкхолдерами могут быть: заказчики (определяющие конечную цель), менеджеры (ответственные за оперативное управление), аналитики (документирующие требования), разработчики (понимающие реальные потребности заказчика), а также специалисты по тестированию, техническому описанию и разработке интерфейсов пользователя, понимающие поведение системы. Для каждого из них определяются необходимые действия и варианты поведения, но во избежание «тяжеловесности» методологии она предполагает возможность отказаться от определенных работ и задач, в которых нет необходимости для конкретного проекта. Все эти работы и задачи сформированы в рамках четырех основных этапов и сочетания шести основных и трех вспомогательных рабочих процессов.



Важно, что RUP, представляя собой в конечном счете набор решений оптимизации разработки ПО, автоматизируем. В частности, компания Rational Software (в 2003-м году ставшая подразделением IBM), описав возможные роли участников проектов, все возможные виды проектных документов для каждого из них, как и формы распределения ответственности, создала программное решение для поддержки всего процесса разработки. И его использование по сути гарантирует достижение третьей стадии зрелости модели CMM, что для многих процессов многих организаций является достаточным уровнем развития. Данный инструмент, IBM Rational Method Composer, позволяет создавать, конфигурировать и публиковать определенные процессы разработки.

И наконец, RUP содержит шесть основных «лучших практик», описанных в RUP и направленных на повышение продуктивности и снижение вероятности появления ошибок (снижение рисков):

– **Итеративная разработка.**

Несмотря на рациональность единовременного сбора всех требований и создания на их основе системы, значительным преимуществом итеративной разработки является минимизация стоимости каждой из ее фаз за счет снижения вероятности появления ошибок.

– **Управление требованиями.**

– **Компонентный подход.**

Разбиение масштабных проектов на несколько этапов / областей неизбежно, и важно обеспечивать качество каждого из компонентов перед его интеграцией в итоговую систему. При этом RUP предлагает повторное использование кода через объектно-ориентированное программирование как эффективный элемент компонентного подхода.

– **Визуальное моделирование.**

Использование диаграмм, в частности, UML, позволяет обеспечить большую прозрачность процесса как для представления бизнес-заказчику по результатам проектирования, так и для обсуждения внутри проектной команды и последующей реализации на стадии настройки / внедрения решения. – **Поддержание уровня качества.**

Фокус на тестировании, поддерживать который становится все сложнее по мере развития проекта, всегда должен оставаться одним из главных элементов обеспечения качества.

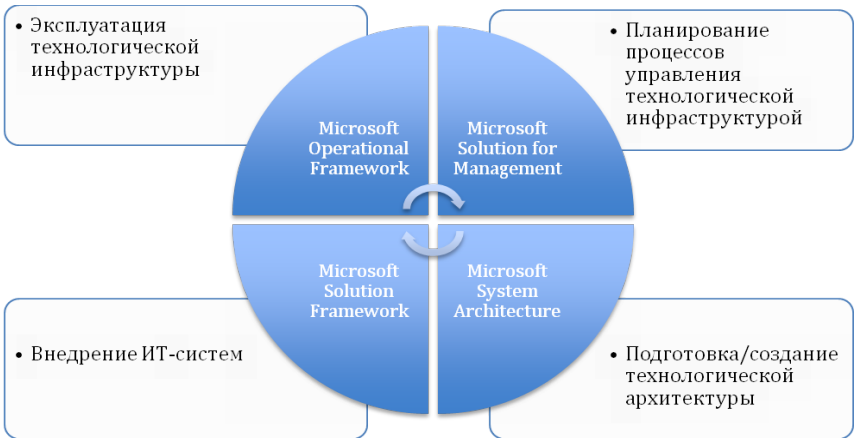
– **Мониторинг изменений.**

Актуальный в условиях разработки продукта несколькими командами, в особенности географически распределенными, мониторинг сделанных изменений позволяет обеспечить плавную интеграцию и консолидацию отдельных элементов системы.

Таким образом, широкий диапазон возможностей и инструментов, предлагаемый методологией, является общей моделью «конструктора», необходимые элементы которого можно выбрать – и таким образом, значительно снизить стоимость / сроки внедрения. Однако крайне важно принимать во внимание, что подобные решения должны приниматься исключительно профессионалами, уже имеющими значительный опыт подобных проектов для манипулирования со степенью формализма разработки и грамотного применения итерационных возможностей.

3.1.1.2. Microsoft (Microsoft Solution Framework, MSF)

Компанией Microsoft, будучи мировым гигантом в сфере разработки программного обеспечения, подготовлены и применяются несколько методик для покрытия не только ЖЦ информационных систем, но и технологической инфраструктуры, их поддерживающей.



Однако в контексте рассмотрения ЖЦ ПО нас интересует именно методология разработки: как являющийся одним из основных аспект управления и взаимодействия участников процесса, так и другие области знаний (управление рисками, планирование). В целом охватываемые MSF дисциплины описаны в пяти частях (т.н. «белых книгах»), однако интересно, что командами консультантов Microsoft применяется на практике не этот ресурс, а методика MSF for Agile Software Development, являясь прикладным вариантом MSF и разумеется, отражая общий методологический подход к итеративной разработке.

Если обратиться непосредственно к процессу разработки / внедрения, то его характеризуют:

– Итеративность.

– Формирование в качестве результата **ИТ-решения**.

ИТ-решение – скоординированная поставка набора элементов (таких как программно-технические средства, документация, обучение, сопровождение и внешние коммуникации), необходимых для удовлетворения некоторой бизнес потребности конкретного заказчика.

Именно ИТ-решения (а не просто программные продукты, поставляемые в виде дистрибутивов) являются основным продуктом и результатом процесса разработки и внедрения по MSF (несмотря на то, что аналогичная философия лежит в основе многих методологий, все равно термин «программный продукт» является очень распространенным).

– Базируемость на вехах (основных и вспомогательных) и артефактах как результатах их достижения.

ПРИМЕР

На схеме концепции MSF основными вехами являются – «Концепция проекта утверждена».

– «Разработка завершена».

В то же время, промежуточными вехами между ними могут быть:

– «Верификация технологий осуществлена».

– *«Базовая версия функциональной спецификации создана».*

– *«Базовая версия сводного плана создана».*

«Базовая версия сводного календарного графика проекта создана».

– *Среды разработки и тестирования развернуты».*

В целом, важным преимуществом MSF является ее практическая направленность и простота. Для MSF очень важно взаимодействие внутри проектной команды, и несмотря на то, что минимальное число ее участников официально в методологии ограничено всего тремя, выделяются шесть основных ролей внутри команды, и связь между участником и ролью имеет тип «многие-ко-многим». Среди шести упомянутых кластеров:

– управление программой (архитектурным решением);

– разработка (программной и технической архитектуры);

– тестирование (планирование, разработка, отчетность);

– управление релизами;

– управление требованиями заказчика (в части интерфейса решения, а также, конечно, обучения и технической поддержки);

– управление продуктом (по сути, управление требованиями бизнесзаказчика и бизнес-приоритетами).

Именно на этих областях концентрируется методология, каждый из кластеров отвечает за достижение определенной цели, однако за итоговый результат отвечает вся команда.

Для более крупных проектов с большими командами внедрения могут дополнительно создаваться группы направлений и функциональные группы, причем MSF даже предлагает таблицу совместимости ролей: какие из них допустимо, нежелательно или совсем нельзя совмещать. Допустимо совмещать, например:

– Управление продуктом, Тестирование, Управление требованиями заказчика.

– Управление релизами и Тестирование.

Однако возложение ответственности по Управлению проектом и Управлению продуктом на одного человека может привести к конфликту интересов, так как менеджер проекта должен контролировать основные процессы в области стоимости, сроков, взаимодействия команды, рисков и т.д., а не определять приоритеты бизнеса. Есть и роль, которую вообще не рекомендуется совмещать с другими – роль Разработчика, что сделано исходя из предположения, что его активности являются наиболее критичными в проекте, и любое наделение разработчиков дополнительными обязанностями приведет к срыву план-графика проекта.

Данное распределение становится более понятным, если рассмотреть значение каждой роли и основные активности:

Таким образом, в отличие от многих других корпоративных методологий, определенные в MSF этапы / вехи, состав проектной группы, ролевая модель и другие элементы подходят не только для решений Microsoft. А значит, MSF представляет собой более гибкий и универсальный подход для внедрения других систем / программных продуктов.

3.1.1.3. On Target

Методология внедрения решений On Target была разработана компанией Navision для внедрения своих программных продуктов. После приобретения Navision и вхождения ее в состав Microsoft было принято решение доработать On Target, к тому моменту содержавшую шаблоны описаний бизнес-процессов, документации, организационных структур ИТ и квалификационных требований к специалистам.

Рассмотрим основные этапы внедрения в соответствии с On Target: – Подготовка проекта.

Формирование проектной команды и подготовка документации.

– Анализ.

Формирование функциональных требований к системе по результатам анализа бизнес-процессов. Обучение ключевых пользователей, разработчиков и системных администраторов со стороны компании заказчика. – Дизайн.

Разработка и согласование технического задания и эскизного проекта.

Финализация плана проекта и бюджета.

– Разработка и тестирование.

Написание программного кода, разработка интерфейсов, настройка и тестирование.

– Развертывание.

Установка системы на стендовом сервере и рабочих местах заказчика, настройка прав доступа, миграция данных, подготовка системной документации и инструкций пользователей, их обучение.

– Опытная эксплуатация.

Приемка в ОПЭ, окончательная верификация данных и процедур после миграции, опытная эксплуатация, передача в промышленную эксплуатацию.

В силу того, что к моменту приобретения Navision у Microsoft уже применялись свои проверенные корпоративные методологии MSF и MOF, в дальнейшем On Target была дополнена и к моменту выведения на рынок Microsoft Dynamics превратилась в результате доработок в MS Dynamics Sure Step / Microsoft Business Solutions Partner Methodology.

3.1.1.4. Microsoft Dynamics Sure Step и Microsoft Business Solutions Partner Methodology

Sure Step была разработана для партнерской сети Misrosoft с целью обеспечения единого подхода к внедрению решений класса MS Dynamics (почему и получила название MS Business Solutions Partner Methodology, в частности используемое для программного решения Modeling Tool). Она основывается на лучших практиках проектов внедрения и для соответствия различным сценариям предлагает несколько компонентов:

– **Этапы.**

С предшествующей методологии On Target отличия незначительные: **Подготовка проекта** заменена на **Диагностику,** а **Опытная эксплуатация** на **Начальное сопровождение.** Соответственно, охватывается уже не только внедрение, но и поддержка заказчика по различным каналам коммуникации, периодические обновления, а также планирование новых проектов.

– **Процессы.**

ПРИМЕР

На стадии дизайна схема процессов выглядит следующим образом:

– **Предложения.**

Sure Step предоставляет возможность не обязательно проходить все этапы модели методологии, но и выбирать необходимые в соответствии со спецификой бизнеса и задачами проекта, что и отражается в компоненте Предложения.

ПРИМЕР

Диагностика, подробный анализ; внедрение (полное либо быстрое); оптимизация.

– **Отчетные материалы.**

Методология предоставляет информацию по входным и итоговым документам каждого этапа, а в отношении всего проекта в качестве стандартных конечных результатов она предполагает создание следующих документов:

– оценка инфраструктуры;

– план проекта;

– функциональные требования;

– план и контрольный список реализации.

Отчетные материалы являются прямыми результатами соответствующих процессов этапов. – **Межэтапные (постоянные) процессы.**

Как уже было сказано, в зависимости от сценария методология внедрения может меняться и для ее поддержки существует группа постоянных процессов в виде взаимосвязанных операций.

– анализ бизнес-процесса;

– конфигурация;

– перенос данных; – инфраструктура;

– инсталляция;

– интеграция; – тестирование; – обучение.

Выделяются те операции, которые относятся к конкретным процессам и охватывают несколько этапов.

– **Процедуры управления проектами.**

А в свою очередь, для охвата всех процессов управления проектами выделены и такие процедуры, как Управление временем и затратами, коммуникациями, качеством, снабжением и прочие стандартные области проектного менеджмента. – **Роли консультантов и клиентов.**

Sure Step определяет основные роли специалистов со стороны заказчика внедрения и подрядчиков и охватывает рекомендации для каждой из них.

Со стороны заказчика	Со стороны исполнителя
– Руководитель, принимающий решения – Руководитель проекта – Руководитель отдела ИТ – Ключевой пользователь – Конечный пользователь	– Руководитель проекта – Менеджер по контактам – Архитектор решений – Консультант по приложениям – Консультант по разработке – Технический консультант

Сочетание одним специалистом нескольких ролей, в особенности в проектах небольшого масштаба соответствует методологии в полной мере.

Таким образом, MS Dynamics Sure Step представляет собой комплексную методологию внедрения, оптимальную для программных продуктов Dynamics CRM, AX, NAV за счет подробных шаблонов и схем процессов, а также средств редактирования для адаптации методологии к специфике предприятия и отрасли. Методология определяет необходимые шаги каждого этапа внедрения продуктов MS Dynamics как в виде иерархической структуры работ, так и в виде графических схем процессов MS Visio.

3.1.1.5. SAP (Accelerated SAP)

Призванная снижать риски, стоимость и время внедрения программных продуктов, методология ASAP основывается на собственном опыте SAP в области разработки технических решений и бизнес-консалтинга, со значительным фокусом на уже проверенных методиках управления проектами и инфраструктуре PMBoK. Так, пять ключевых этапов маршрутной карты SAP достаточно четко соотносятся с этапами проектного менеджмента PMBoK [53, сл. 6].

В целом же корпоративная методология Accelerated SAP (ASAP) предлагает комплексный инструментарий, сочетающий технологическую и управленческую части в виде трех основных компонентов:

– Синхронизация бизнес-приоритетов клиента (в зависимости от индустрии) с конкретными решениями SAP через отраслевые карты бизнес-ценностей (value maps) инструмента Solution Explorer.

– Карты управления проектами (маршрутные карты) с основными этапами и активностями по ним.

– Проектирование, настройка, тестирование и эксплуатация решения при помощи менеджера решений SAP Solution Manager.

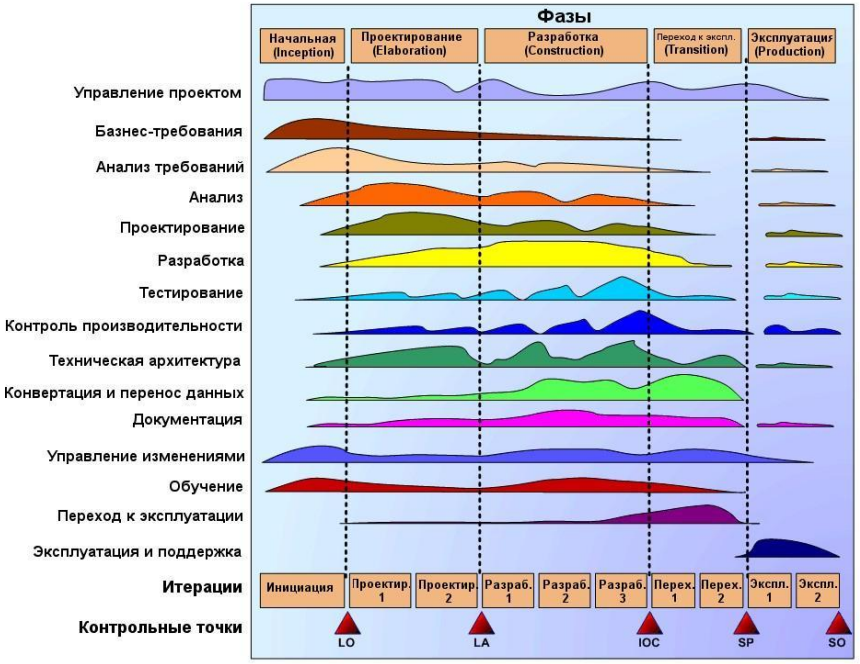
Таким образом, за счет выстроенной методологии управления проектами внедрения своих программных продуктов, сочетающейся с программным инструментом для ее практического применения в виде Solution Explorer, SAP способствует структурированию большого количества задач, которые необходимо исполнять на каждом этапе.

Среди других преимуществ данной методологии, уменьшается время между инсталляцией и запуском системы в продуктив в режиме реального времени, а также создает общую модель управления проектом, которая может впоследствии применяться и для других внедряемых компаний продуктов / обновлений системы.

3.1.1.6. Oracle (Oracle Unified Method, OUM)

Oracle Unified Method (OUM) – корпоративная методология поддержки ЖЦ информационных технологий на предприятии, содержащая общее руководство по настройке и типовую структуру работ, а также набор шаблонов, комплекты материалов по конкретным продуктам Oracle и необходимые программные инструменты. Изначально основой методологии Oracle стала модель унифицированного процесса разработки (Unified Process), которая была адаптирована также IBM для формирования собственной корпоративной методологии Rational Unified Process.

OUM охватывает три основных аспекта: управление программами и проектами, поддержку ИТ-архитектуры и подход к реализации проектов. Выделяя аналогичные ASAP пять стадий, Oracle изначально предполагает возможность итераций внутри каждой из них (две итерации проектирования, три итерации разработки) с наглядным представлением, на каком именно из 15ти процессов разработки следует фокусироваться в каждый конкретный момент [52].



Вполне резонно методология от Oracle предполагает регулярные активности процессов управления изменениями и документацией, множество раундов тестирования и контроля производительности, отдельное внимание технической архитектуре, обучение (в особенности на начальной стадии и в ходе разработки при подготовки к эксплуатации). А релизы поддерживают также сервис-ориентированную архитектуру, управление учетными записями, идеологию Entreprise 2.0 и другие актуальные проектные технологии, что способствует популярности данного метода разработки ПО среди корпоративных клиентов.

3.1.1.7. Oracle / PeopleSoft One Methodology

Разработанная компанией PeopleSoft, вошедшей в состав Oracle данная методология предназначена для описания внедрения ИС американской компании JD Edwards (в свою очередь, тоже поглощенной PeopleSoft). Первоначально разрабатывая финансовое ПО под аппаратное обеспечение IBM, JD Edwards создала методологию, в равной степени применимую к разным категориям систем: управления ресурсами, цепочками поставок, взаимоотношениями с клиентами, интеллектуального анализа данных и сотрудничества в Интернет.

Среди некоторых целей One Methodology – создание единой системы из иерархии целей проекта, его ограничений и планируемых результатов, формирование требований к команде внедрения, определение политики в области рисков. Содержание этапов данной методологии значительно отличается от рассмотренных ранее[1].

Этап	Описание работ

[1] По материалам [54].

Границы внедрения (основные задачи и цели в рамках проекта)	– Определение функциональных целей – Разработка технологической архитектуры – Конвертация данных – Интерфейсы с внешними программами
Модель (проект системы)	– Общий обзор и планирование – Моделирование бизнес-процессов – Анализ недостающей функциональности – Планирование доработок ПО
Конфигурирование (реализация пилотного проекта и инсталляция / развертывание системы)	– Обучение проектной группы – Прогонка по системе – Ввод исходных данных – Конфигурирование ПО – Разработка пользовательской документации – Формирование прав доступа – Интеграция
Ввод в эксплуатацию (старт ОПЭ)	– Тестирование рабочей конфигурации – Обучение конечных пользователей – Настройка производительности системы – Запуск системы в ОПЭ
Развитие (оптимизация системы)	– Оценка работоспособности недостающей функциональности – Оптимизация бизнес-процессов – Передача системы

Однако в результате последовательного поглощения JD Edwards ⇨ PeopleSoft ⇨ Oracle и значительной направленности JD Edwards на узкую линию программных продуктов, данная методология потеряла свою актуальность и уступила место другим методологиям, в частности, разработанной Oracle AIM в составе Oracle Unified Method.

3.1.2. Индустриальные стандарты и методологии

3.1.2.1. Agile

В переводе с английского Agile означает «гибкий» – именно поэтому такое название получила методология разработки, которая не содержит четких инструкций и «лучших практик», однако представляет собой сборник основных принципов и ценностей в предметной области разработки ПО. К тому же, сам процесс гибкой разработки является адаптивным по отношению к постоянно изменяющимся условиям, что достигается разработкой за короткие итерации, после каждой из которых происходит пересмотр требований и в случае необходимости – изменение практик коммуникаций и работы команды. Тем временем, есть еще несколько не менее важных и требующих рассмотрения идей Agile:

– Приоритет взаимодействия людей над процессами и традиционными инструментами управления.
– Приоритет получения работающего продукта над исчерпывающей всеобъемлющей документацией.
– Приоритет сотрудничества с потребителями (заказчиком) над формальными вопросами контрактов.
– Приоритет быстрого реагирования на изменения над неотступным следованием плану.

Исходя из этих ценностей, среди команд, использующих Agile-методологию, почти не встречается одинаковых практик документирования или координации, так как «гибкая методология» предполагает самоорганизацию, самостоятельное определение объема, содержания и ограничений каждого элемента управления разработкой. Со времени своего появления в 1990-х Agile-практики получили свое распространение во множестве компаний, в то время как первоначально они позиционировались для управления ИТ-проектами.

В качестве одного из важнейших элементов Agile-подхода к разработке выступают пользовательские истории (user stories). Это описанные одним или несколькими предложениями основные аспекты функциональности системы, необходимые для выполнения пользователем своей работы. В достаточно сжатом виде, они должны отвечать на вопросы: «Кто?», «Что?» и «Почему?» и могут создаваться как менеджером проекта по результатам обследования как один из форматов фиксирования требований, так и разработчиками для выражения нефункциональных требований (например, к безопасности, производительности, качеству решения).

ПРИМЕР

Отдельные требования, представленные в виде user stories:

Как ответственный за проведение рассылок, я должен вести мониторинг адресатов по разным категориям.

Как ответственный за проведение рассылок, я бы хотел давать права администратора моим подчиненным в случае необходимости.

Как ответственный за проведение рассылок, я хочу видеть знать, сколько процентов пользователей проходят по ссылке в письме.

Как ответственный за проведение рассылок, я должен иметь возможность выделить адресатам, не открывшим письмо в течение двух недель.

Как ответственный за проведение рассылок, я бы хотел создавать их план в перспективе на 2 недели вперед.

Как ответственный за проведение рассылок, мне необходим удобный формат визуального представления и выгрузки статистических данных.

Как ответственный за проведение рассылок, я должен знать источник получения контактных данных каждого адресата.

Как ответственный за проведение рассылок, я должен получать недублированные данные, то есть одному пользователю может соответствовать только один текущий электронный адрес. ***Итоговая user story на «техническом» языке:***

Как ответственному за проведение рассылок, пользователю необходимо обеспечить следующую функциональность:

- *Иметь конфигурацию статусов каждого адресата: адрес электронной почты, реакция на каждую из проведенных рассылок (красная иконка для открывших письмо и не прореагировавших на него, желтая – для неоткрытых сообщений, зеленая – для прочитавших и выполнивших требовавшееся в рассылке действие).*
- *Иметь несколько уровней прав доступа к рассылкам (на просмотр статистики, на выгрузку e-mail адресов, на отправку рассылок, ...).*
- *Создать персональные ссылки в отправляемых сообщениях, через которые будет осуществлен подсчет числа пользователей, пришедших на страничку портала по результатам рассылки.*
- *...*

В то же время, agile почти не ограничивает команду проекта: так, если необходим больший уровень детализации технического задания / спецификаций / другой документации – можно его написать, если необходимы дополнительные приемочные критерии – можно их составить, если нужны прототипы – можно их сформировать или в виде макетов (мокапов, mockups), или в виде блок-схем, или в виде реальной модели прототипа. Уровень детализации и формализации всегда зависит от специфики системы и проектной команды, от их компетенций и опыта реализации подобных проектов.

3.1.2.2. SCRUM

В переводе с английского языка «SCRUM» означает «схватка». Впервые подобная методология была введена в употребление профессорами японского Hitotsubashi University в их статье «Новая игра в производстве продукта» («The New Product Development Game»), написанной в 1986 году. В ней разработка продукта была сравнена со схваткой вокруг мяча в регби – приемом, называемым «скрам». Тем не менее, авторами концепции считаются разработчики из США, Кен Швабер и Джефф Сазерленд, впервые определившие и описавшие основополагающие принципы SCRUM – «гибкой» методологии управления проектами (чаще всего применяющейся в области разработки ПО).

По своей сути SCRUM представляет собой набор принципов разработки, которые позволяет представлять конечному пользователю (и заказчику) действующий продукт / прототип, обладающий новыми функциями и возможностями с наивысшим приоритетом. Работа в этом случае проводится в четко фиксированные недлительные (в среднем от 1 до 6 недель, чаще от 2 до 4) итерации (спринты). Ожидаемые результаты спринта определяются командой под руководством SCRUM-мастера в самом его начале и НЕ могут изменяться до завершения. К ним под руководством SCRUM-мастера (по сути руководителя проекта) и product-мастера (заказчика, владельца проекта) создается список работ на период спринта и на весь проект (sprint-backlog и product-backlog соответственно).

С одной стороны, в силу того, что релизы выпускаются часто, минимизируется вероятность ошибок за счет постоянной обратной связи с потребителями при демонстрации новой функциональности, а также ежедневных (!) встреч команды проекта. С другой стороны, увеличиваются временные (и денежные) затраты на проведение презентаций, а также на исправление выявленных проблем и осуществляемый ретроспективный анализ.

Однако при более глубоком рассмотрении SCRUM имеет очень важные основы: методология предполагает циклический и очень активный процесс, минимизирующий неопределенность требований за счет коротких итераций и предоставляя возможность детального прототипирования. Долгие, нестандартные, динамические проекты, при расплывчатом представлении о конечном ожидаемом результате и постоянной смене приоритетов не являются проблемой для SCRUM, и именно поэтому он часто применяется на первых этапах, затем в целях экономии средств / ресурсов переходя к использованию другой методологии. Соответственно, SCRUM может применяться практически любым типом организаций – от стартапов с их высокой неопределенностью до госзаказчиков с частым изменением требований и частой необходимостью демонстрации текущей версии для отчета о результатах.

При принятии решения о переходе на SCRUM необходимо принимать во внимание возможное сопротивление со стороны некоторых специалистов, не привыкших к подобным подходам или предпочитающими индивидуальную работу, так как именно в этой методологии крайне важна сплоченность команды, ее уровень ответственности и кросс-функциональность для ежедневного поиска оптимальных путей совершенствования продукта. Также высока зависимость успеха SCRUM-проектов от мастерства и компетенций SCRUM-мастера, который должен обладать достаточным авторитетом и умением управлять командой для успешной организации выполнения работ и контроля проекта на всех этапах.

3.1.2.3. RAD

Методология быстрой разработки приложений[2] RAD (rapid application development) предполагает наличие трех основных элементов: небольшой команды до 10 человек, непродолжительный период внедрения в 2-6 месяцев и итеративный цикл, в основе которого спиральная модель жизненного цикла ИС. Основу разработки RAD составляет использование инструментальных средств для каждой фазы жизненного цикла системы – анализа требований, проектирования, генерации кода и разработки интерфейса. Часто используются CASE-средства и инструменты прототипирования интерфейса, которые позволяют продемонстрировать саму концепцию планируемого результата уже в самом начале проекта. Именно поэтому итерационный принцип настолько соответствует методологии и он удобен при нечетко определенных и изменяющихся требованиях к итоговой системе. Итоговый результат, работоспособная система, актуальная на текущий момент (не обязательно в перспективе нескольких лет!) может быть получен более оперативно именно при помощи RAD.

Однако есть и некоторые отличия с традиционной спиральной моделью. В силу этого же итеративного подход, как и другие подобные методологии, RAD не может применяться для разработки сложных систем. (например, от которых зависит безопасность людей, таких как системы управления судами или самолетами), так как первые версии не будут полностью работоспособны. Не следует удивляться также и предложению подрядчика использовать принципы RAD на первых этапах, затем переписывая часть кода на «традиционном» языке программирования – таком, как C# или C++.

Преимуществом RAD среди методологий разработки является совместная работа с заказчиком / бизнес-пользователями, которые активно участвуют в прототипировании, написании сценариев тестирования и его проведении. Другой особенностью является совместное принятие решений команды, и подобный децентрализованный стиль управления может привести к периодической смене курса проекта или возможному излишне длительному принятию решения из-за поиска консенсуса.

3.1.2.4. XP

Основная особенность[3] методологии экстремального программирования (**eXtreme Programming**) состоит в ее эффективности в условиях неопределенных или нечетких требований. Популярность данного подхода увеличивалась по мере роста числа разработчиков, недовольных традиционным подходом со множеством формальных требований и постоянно готовивших документацию, собиравшими информацию о показателях проекта. Напротив, новая методология предлагала простой дизайн, переработку (рефакторинг) кода для контроля затрат, постоянное присутствие заказчика, разработку через тесты и другие аспекты, выгодно отличавшие ее.

Обозначим другие ключевые аспекты данной методологии.

Для работы в подобных условиях необходима **постоянная связь с заказчиком**, которая обеспечивается полноценным участием в работе проектной команды квалифицированного, находящегося в курсе проекта представителя заказчика.

При разработке в большинстве случаев выбираются наиболее простые методы, исходя из принципа того, что легче в будущем вносить дополнения в базовую версию, чем перестраивать усложненную (хотя, разумеется, бывают исключения, когда изначально учет многих факторов в системе может помочь избежать в дальнейшем многих сложностей). **Принцип простоты** также важен и в интерфейсном решении, когда более интуитивный пользовательский дизайн интерфейса обладает исключительно необходимыми (но не излишними!) функциями. Однако предварительное детальное проектирование интерфейса согласно исходной версии методологии не осуществляется, он создается лишь по мере работы команды в течение проекта.

Коллективная работа по написанию кода / внесению изменений в настройки. Этот принцип относится к двум аспектам. В первую очередь, это «**коллективное владение кодом**», когда любой участник команды разработчиков может внести изменения благодаря единым правилам оформления когда и использованию стандартов для ПО. С

[2] На основе материалов портала edu.dvgups.ru.

[3] На основе материалов портала edu.dvgups.ru.

другой стороны, применяется метод «**парного программирования**», в соответствии с которым двое разработчиков используют один компьютер, чередуя написание кода и доработку настроек, реализацию требований и прочие вопросы.

4. Управление стоимостью проекта

4.1 . Методы оценки трудоемкости

Методы, основанные на экспертных оценках

Проведение экспертизы по методу Дельфи производится по схеме, представленной на рис.

Подходы, основанные на экспертных оценках, применяются при отсутствии дискретных эмпирических данных. Они используют опыт и знания экспертов-практиков в различных областях. Оценки, получаемые при этом, представляют собой синтез известных результатов прошлых проектов, в которых принимал участие эксперт.

Очевидными недостатками таких методов являются зависимость оценки от мнения эксперта и тот факт, что, как правило, не существует способов проверить правильность этого мнения до тех пор, когда уже сложно что-либо исправить в том случае, если мнение эксперта окажется ошибочным. Известно, что годы практической работы не всегда переходят в высокое качество профессиональных знаний. Даже признанные эксперты иногда делают неверные догадки и предположения. На основе экспертных оценок были разработаны два метода, допускающие возможность ошибки экспертов: метод Дельфи и метод декомпозиции работ.

Метод Дельфи

Метод Дельфи был разработан в корпорации «Рэнд» в конце 1940-х гг. и использовался первоначально для прогнозирования будущих событий (отсюда метод и получил свое название

по сходству с предсказаниями Дельфийского оракула в Древней Греции). Позднее метод использовался для принятия решений по спорным вопросам. На предварительном этапе участники дискуссии должны без обсуждения с другими ответить на ряд вопросов, относительно их мнения по спорному вопросу. Затем ответы обобщаются, табулируются и возвращаются каждому участнику дискуссии для проведения второго этапа, на котором участникам снова предстоит дать свою оценку спорного вопроса, но на этот раз, располагая мнениями других участников, полученными на первом этапе. Второй этап завершается сужением и выделением круга мнений, отражающих некоторую общую оценку проблемы. Изначально в методе Дельфи коллективное обсуждение не использовалось; обсуждение между этапами метода было впервые применено в обобщенном методе Дельфи. Метод достаточно эффективен в том случае, если необходимо сделать заключение по некоторой проблеме, а доступная информация состоит больше из «мнений экспертов», чем из строго определенных эмпирических данных

Метод Wideband Delphi является вариантом классического метода принятия управленческих решений Delphi. Применяется на этапах постановки задач и оценки различных способов ее решения. Цель метода - получение согласованной информации в процессе анонимного обмена мнениями между экспертами. Это инструмент, с помощью которого можно получить независимое мнение всех участников группы экспертов по конкретному вопросу посредством последовательного объединения идей, мнений и предложений и прийти к согласию. Метод основан на многократных анонимных опросах внутри группы.
План действий
Сформировать рабочую группу для сбора и обобщения мнений экспертов.
Сформировать экспертную группу из специалистов, владеющих вопросами по обсуждаемой теме.
Подготовить анкету, указав в ней поставленную проблему, уточняющие вопросы. Формулировки должны быть четкими и однозначно трактуемыми, предполагать однозначные ответы.
Провести опрос экспертов в соответствии с методикой, предполагающей при необходимости повторение процедуры. Полученные ответы служат основой для формулирования вопросов для следующего этапа.
Обобщить экспертные заключения и выдать рекомендации по поставленной проблеме.
При проведении оценки привлекается несколько экспертов, на первоначальном этапе независимо вырабатываются оценки и анонимно рассматриваются. Такой метод позволяет легко выявить неточные или необоснованные прогнозы, сформулировать основные требования и спецификации проекта. В результате формируется общий документ с числовыми оценками размера проекта, трудоемкости, минимальных аппаратных средств, стоимости.

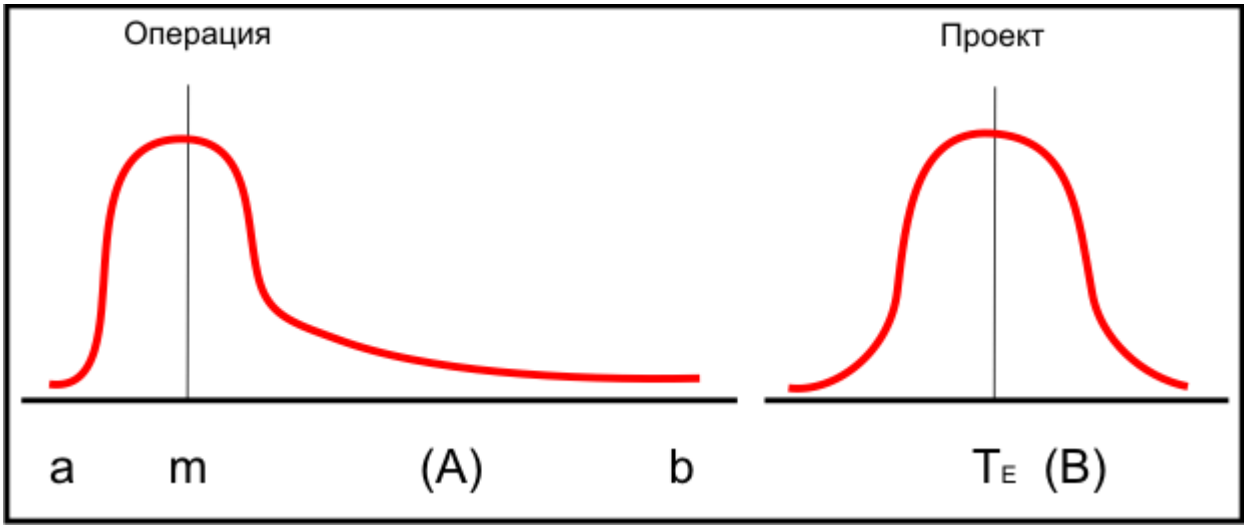
Считается, что метод Delphi лучше использовать, если к работе привлекаются эксперты, компетентные не по всей проблеме, а по ее отдельным вопросам.
Достоинства:

- Способствует независимости мышления членов группы.
- Обеспечивает спокойное и объективное изучение проблем.
- Недостатки:
- Чрезмерная субъективность оценок.
- Требует достаточно много времени и организационных усилий.

PERT - метод оценки и проверки программ

В 1958 г. Особый отдел Военно-морского флота и консалтинговая фирма Booze, Allen and Hamilton создали *PERT* (метод оценки и проверки программ) с целью разработки графика для

более чем 3300 подрядчиков, работающих над проектом подводной лодки Поларис, для решения проблемы неопределенности в расчетах времени выполнения работ.
PERT почти полностью совпадает с методом критического пути (CPM), за исключением того, что *PERT* считает, что продолжительность каждой операции имеет пределы, которые исходят из статистического распределения.
PERT использует 3 оценки расчета времени для каждой операции:
· оптимистическое (наилучшее);
· средний показатель;
· пессимистическое (наихудшее).
Разработчики *PERT* для выражения продолжительности операции решили избрать аппроксимацию бета-распределения.
На(А) представлено бета-распределение для продолжительности операции, отклоняющееся вправо, и оно представляет собой работу, которая имеет тенденцию отставать от графика.



Операция и плотность распределения проекта

Распределение продолжительности проекта показано в симметрии на рис. 5-2 (В).
Распределение проекта представляет собой сумму средневзвешенных показателей операций на *критическом пути*.
Средневзвешенное время операции рассчитывается по следующей формуле:

$$t_e = \frac{a + 4m + b}{6} \quad (5.1)$$

где t_e - средневзвешенное время операции;
a - оптимистическое время операции (1 шанс из 100, что при нормальных условиях операция будет закончена раньше срока);
b - пессимистическое время операции (1 шанс из 100, что при нормальных условиях операция будет закончена позже срока);
m - наиболее вероятное время операции.
Среднее (детерминистическое) значение накладывают на сеть проекта, как и при использовании CPM, и затем рассчитывают раннее, позднее, резервное и время завершения проектных работ, как они указаны в CPM.
Отклонения в оценках времени операции определяются при помощи следующих уравнений.
Уравнение 5.2 представляет стандартное отклонение для операции.

$$\sigma_{t_e} = \frac{b - a}{6} \quad (5.2)$$

$$\sigma_{T_E} = \sqrt{\sum \sigma_{t_e}^2}$$ (5.3)

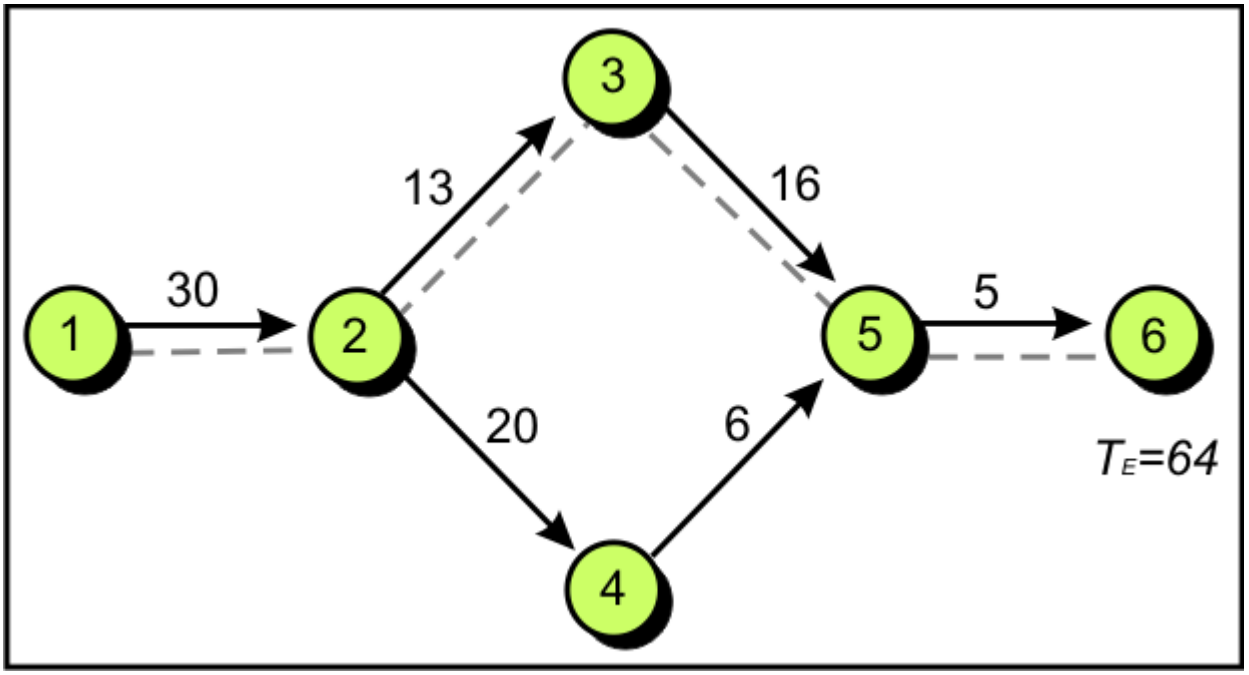
Уравнение 5.3 представляет стандартное отклонение для проекта.
Эта сумма включает в себя только виды операций на критическом или проверенном пути.
Средняя продолжительность проекта (T_E) - это сумма всех средних показателей времени, отведенных на выполнение операций по критическому пути (сумма от t_e), и она следует *нормальному распределению*.
Зная среднюю продолжительность проекта и дисперсии (среднего отклонения) операций, можно с помощью статистических таблиц рассчитать выполнение проекта (или сегмента проекта) к конкретному времени.

Уравнение 5.4 используется для расчета величины Z, приводимой в статистических таблицах (Z - количество *стандартных отклонений* от *средней величины*):

$$Z = \frac{T_S - T_E}{\sum \sigma_{t_e}^2}$$ (5.4)

где T_E - продолжительность критического пути;
T_S - продолжительность работы по графику;
Z - вероятность (выполнения графика), определенная по статистической табл. 5.6.
Гипотетический пример использования метода PERT

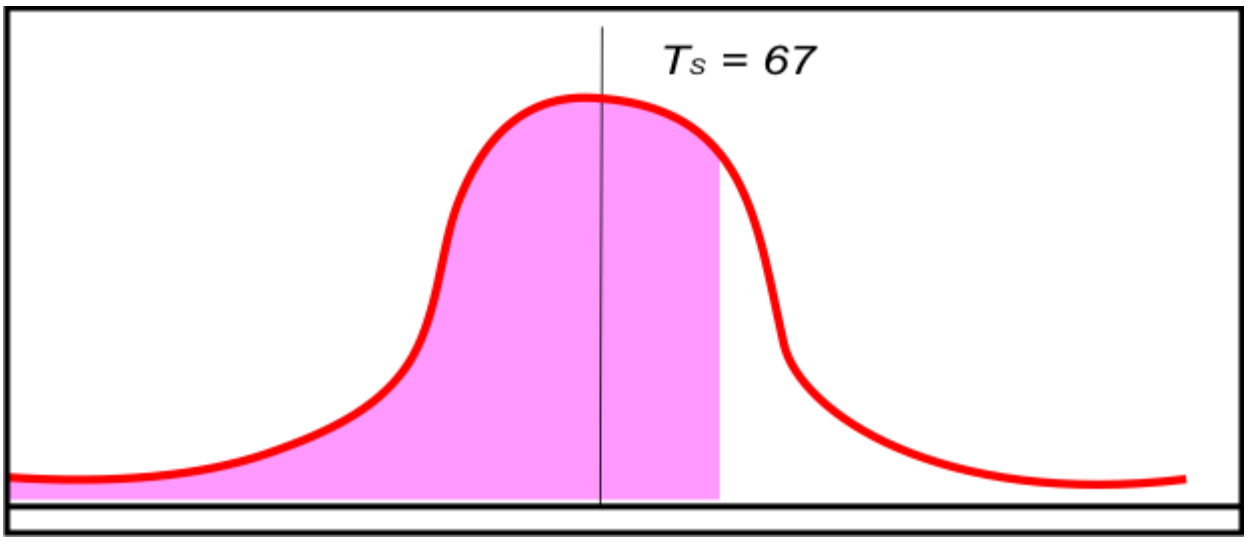
Операция a m b T_E квадрат среднего отклонения
Сеть проекта представлена на рис. 5.3.



Гипотетическая сеть

Прогнозируемый срок работы (T_E) представлен 64 единицами времени;
Критический путь - 1, 2, 3, 5, 6.
Имея эту информацию и используя стандартные *статистические методы*, можно легко рассчитать вероятность выполнения проекта к конкретному времени.

Например, какова вероятность завершения работы над проектом до указанного в графике времени (T_s) из 67?
Обычная кривая проекта будет такой как на рис



Возможная продолжительность проекта

Используя формулу для значения Z, можно рассчитать вероятность следующим образом:

$$Z = \frac{T_S - T_E}{\sum \sigma_{t_e}^2} = \frac{67 - 64}{\sqrt{25 + 9 + 1 + 1}} = +0.50$$
$$P = 0.69$$

По данным табл. 5.6 значение Z + 0,5 дает вероятность 0,69, что означает 69%-ную вероятность завершения работы над проектом на 67-ю единицу времени или ранее.
Таблица 5.6.

Величина Z	Вероятность	Величина Z	Вероятность
-2,0	0,02	+2,0	0,98
-1,5	0,07	+ 1,5	0,93
-1,0	0,16	+1,0	0,84
-0,7	0,24	+0,7	0,76
-0,5	0,31	+0,5	0,69
-0.3	0,38	+0,3	0,62
-0,1	0,36	+0,1	0,54

Вероятность выполнения проекта к периоду времени 60 рассчитывается следующим образом:

$$Z = \frac{60 - 64}{\sqrt{25 + 9 + 1 + 1}} = \frac{-4}{\sqrt{36}} = -0.67$$
$$P = 0.26$$

По табл. 5.6 значение Z - 0,67 дает вероятность 0,26, что означает около 26% вероятности завершения работы над проектом на 60-ю единицу времени или ранее.

Аналогичный способ расчетов можно использовать для любого пути или участка пути в сети.
PERT-моделирование
Для этой методики необходима компьютерная программа, моделирующая отпущенные на проект время, затраты и/или наличие ресурсов с использованием метода Monte Carlo Technique.

Решения по сохранению или переадресации рисков принимаются с помощью информации, полученной в результате моделирования времени, затрат и ресурсов.
PERT и моделирование *PERT* применяются в проектах чрезвычайной важности, которым присуща большая степень неопределенности и где в достаточной степени точно можно рассчитать время на выполнение операций.

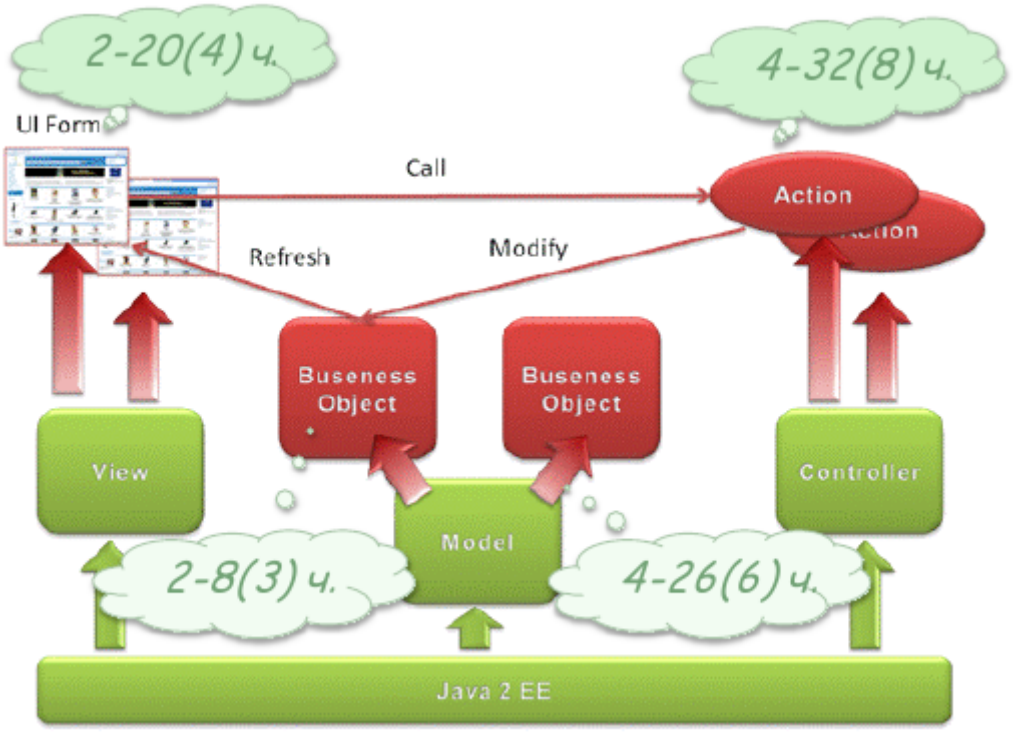
Прагматичный подход. Метод PERT

Использование собственного опыта или опыта коллег, полученного в похожих проектах, это наиболее прагматичный подход, который позволяет получить достаточно реалистичные оценки трудоемкости и срока реализации программного проекта, быстро и без больших затрат.
Инженерный метод оценки трудоемкости проекта PERT (Program / Project Evaluation and Review Technique) был разработан в 1958 году в ходе проекта по созданию баллистических ракет морского базирования «Поларис». Входом для данного метода оценки служит список элементарных пакетов работ. Для инженерного подхода нет необходимости точно знать закон распределения нашей оценки трудоемкости каждого такого элементарного пакета. Диапазон неопределенности достаточно охарактеризовать тремя оценками:

- *Mi* — наиболее вероятная оценка трудозатрат.
- *Oi* — минимально возможные трудозатраты на реализацию пакета работ. Ни один риск не реализовался. Быстрее точно не сделаем. Вероятность такого, что мы уложимся в эти затраты, равна 0.
- *Pi* — пессимистическая оценка трудозатрат. Все риски реализовались.

Оценку средней трудоемкости по каждому элементарному пакету можно определить по формуле:
Ei = (Pi + 4Mi + Oi)/6.
Для расчета среднеквадратичного отклонения используется формула:
СКОi = (Pi - Oi)/6.
Если наши оценки трудоемкости элементарных пакетов работ статистически независимы, а не испорчены, например, необоснованным оптимизмом то, согласно центральной предельной теореме теории вероятностей суммарная трудоемкость проекта может быть рассчитана по формуле:
E = ∑ Ei
А среднеквадратичное отклонение для оценки суммарной трудоемкости будет составлять:
$$СКО = \sqrt{\sum СКО_i^2}$$
Тогда для оценки суммарной трудоемкости проекта, которую мы не превысим с вероятностью 95%, можно применить формулу:
$E_{95\%} = E + 2 * СКО.$
Это значит, что вероятность того, что проект превысит данную оценку трудоемкости составляет всего 5%. А это уже вполне приемлемая оценка, под которой может расписаться профессиональный менеджер.
Список элементарных пакетов работ, который используется при оценке трудоемкости, как правило, берется из нижнего уровня ИСР проекта. Но может быть использован и накопленный опыт аналогичных разработок. Проиллюстрирую данный подход на примере реального проекта. В Ассоциации CBOSS задачей проекта, который нам с коллегами посчастливилось реализовывать, была разработка на основе стандартов J2EE общесистемного ПО для перевода рабочих мест CBOSS на новую трехзвенную архитектуру. Был разработан набор стандартных компонентов и сервисов, из которых как из конструктора можно эффективно и

качественно собирать прикладные подсистемы. Высокоуровневая архитектура реализовывала стандартный паттерн MVC (Рисунок 36), каждый из компонентов которого имел «точки расширения» для прикладной разработки, которые на рисунке выделены красным светом.



Высокоуровневая архитектура J2EE фреймворка для разработки приложений.

Такими точками расширения являлись:

- Пользовательский экран (UI Form), который собирался из готовых визуальных компонентов.
- Обработчики (Action), которые обрабатывали на сервере приложений события от активных визуальных компонентов, входящих в состав экрана.
- Объекты (Business Obj), которые моделировали прикладную область, и к которым обращались обработчики событий.

Так вот, хотя все разрабатываемые рабочие места различались по функциональности и сложности, накопленная статистика фактических трудозатрат на разработку прикладных систем позволяла нам оценивать проекты разработки нового приложения достаточно быстро и с высокой достоверностью.

Согласно этой статистике, разработка и отладка требовала у программиста:

- для одного экрана — от 2 до 20 часов (наиболее вероятно — 4 часа);
- для одного обработчика событий — от 4 до 32 часов (наиболее вероятно — 8 часов);
- для нового бизнес-объекта — от 2 до 8 часов (наиболее вероятно — 3 часа);
- для добавления нового бизнес-метода — от 2 до 26 часов (наиболее вероятно — 6 часов).

Весь проект прикладной разработки измерялся в «попугаях»:

- K_{UI} — количество пользовательских экранов.
- K_{Act} — количество обработчиков событий.

- K_{BO} — количество новых бизнес-объектов.
- K_{BM} — количество новых или модифицируемых бизнес-методов.

Если новое разрабатываемое приложение содержит 20 пользовательских экранов, 60 обработчиков событий, 16 новых бизнес-объекта и 40 новых бизнес-методов, которые необходимо добавить, как в новые, так и в уже существующие бизнес-объекты, тогда, согласно нашей статистике,

*E_{UI} = (2 + 4*4 + 20) / 6 = 6.7 чел.*час., E_{Act} = (4 + 4*8 + 32) / 6 = 11.3 чел.*час., E_{BO} = (2 +4*3 + 8) / 6 = 3.7 чел.*час., E_{BM} = (2 + 4*6 + 26) / 6 = 8.7 чел.*час.,*

*$СКО_{UI}$ = (20 - 2) / 6 = 3 чел.*час $СКО_{Act}$ = (32 - 4) / 6 = 4.7 чел.*час $СКО_{BO}$ = (8 - 2) / 6 = 1 чел.*час $СКО_{BM}$ = (26 - 2) / 6 = 4 чел.*час*

Для средней трудоемкости работ по кодированию в проекте может быть получена следующая оценка:

*E = 20 * 6.7 + 60 * 11.3 + 16 * 3.7 + 40 * 8.7 ≈ 1220 чел.*час.*

$$СКО = \sqrt{20*3^2 + 60*4.7^2 + 16*1^2 + 40*4^2} =$$

$$= \sqrt{180 + 1325 + 16 + 640} \approx 46 \text{ чел.*час.}$$

Тогда для оценки суммарной трудоемкости проекта, которую мы не превысим с вероятностью 95%, получим

*$E_{95\%}$ = 1220 + 2 *46 ≈ 1300 чел.*час.*

Хотя относительная погрешность в оценке трудоемкости каждой такой элементарной работы составляла десятки процентов, для нашего проекта, в котором было таких «попугаев» было 136, относительная погрешность оценки суммарной трудоемкости, сделанной по методу PERT, составила, приблизительно, лишь 4%.

Даже если у нас очень размытые оценки трудоемкости каждой из элементарных работ, но они независимы, то ошибки мы делаем как в меньшую, так и большую стороны. Поэтому при фактической реализации проекта эти ошибки будут компенсироваться, что позволяет нам оценить общие трудозатраты по проекту существенно точнее, чем трудозатраты на каждую элементарную работу. Но это утверждение будет справедливо только в том случае, если наша ИСР содержит *все* необходимые работы, которые должны быть выполнены для получения *всех* продуктов проекта.

Полученную оценку трудоемкости кодирования необходимо умножить на четыре, поскольку помним (см. Лекция 3. Инициация проекта), что кодирование составляет только 25% общих трудозатрат проекта. Поэтому суммарная трудоемкость нашего проекта составит, приблизительно, 5200 чел.*час.

Как мы уже говорили ранее, если сотрудник на 100% назначен на проект, это, как правило, не означает, что он все 40 часов в неделю будет тратить на проектные работы. Тратить он будет 60–80% своего рабочего времени. Поэтому, в месяц сотрудник будет работать по проекту, примерно, 165 * 0.8 = 132 чел.*час/мес. Следовательно, трудоемкость проекта в человеко-месяцах составит, приблизительно 5200 / 132 ≈ 40.

Тогда согласно формуле Б.Боэма (Рисунок 15) оптимальная продолжительность проекта составит:

$T = 2.5 * (40)^{1/3} = 8.5$ месяцев,

а средняя численность команды — 5 человек.

Помним, что потребление ресурсов в проекте неравномерно (Рисунок 13), поэтому начинать проект должны 1–3 человека, а на стадии реализации начальная численность команды может быть увеличена в несколько раз.

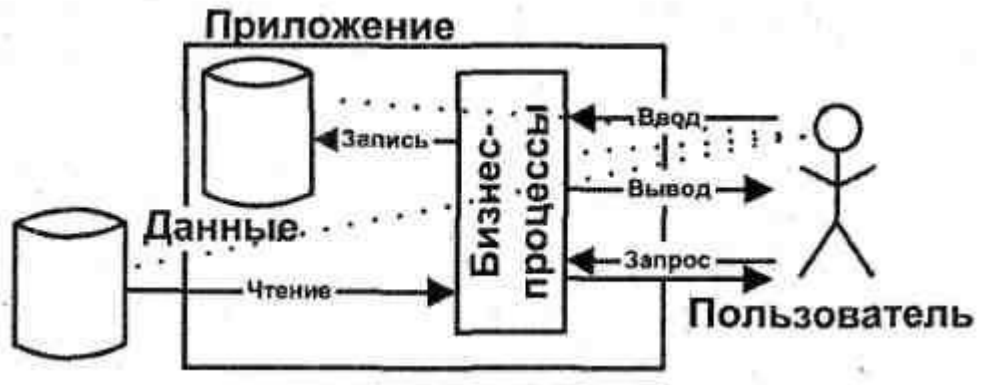
Если же собственный опыт аналогичных проектов отсутствует, а коллеги-эксперты недоступны, то нам не остается ничего другого, как использовать формальные методики, основанные на

обобщенном отраслевом опыте. Среди них наибольшее распространение получили два подхода:

- FPA IFPUG — метод функциональных точек,
- метод COCOMO II, Constructive Cost Model.

Методы оценки трудоемкости: Метод Function Points.

Определение числа функциональных точек является методом количественной оценки ПО, применяемым для измерения функциональных характеристик процессов его разработки и сопровождения независимо от технологии, использованной для его реализации.
Подсчет функциональных точек, помимо средства для объективной оценки ресурсов, необходимых для разработки и сопровождения ПО, применяется также в качестве средства для определения сложности приобретаемого продукта с целью принятия решения о покупке или собственной разработке.
Метод разработан на основе опыта реализации множества проектов создания ПО и поддерживается международной организацией IFPUG (International Function Point User Group).
Рассматриваемый в данном разделе сокращенный вариант методики оценки трудоемкости разработки ПО основан на материалах IFPUG и компании SPR (Software Productivity Research), которая является одним из лидеров в области методов и средств оценки характеристик ПО.
Согласно данной методике трудоемкость вычисляется на основе функциональности разрабатываемой системы, которая, в свою очередь, определяется на основе выявления ***функциональных типов*** — логических групп взаимосвязанных данных, используемых и поддерживаемых приложением, а также элементарных процессов, связанных с вводом и выводом информации (рис. 6.2).



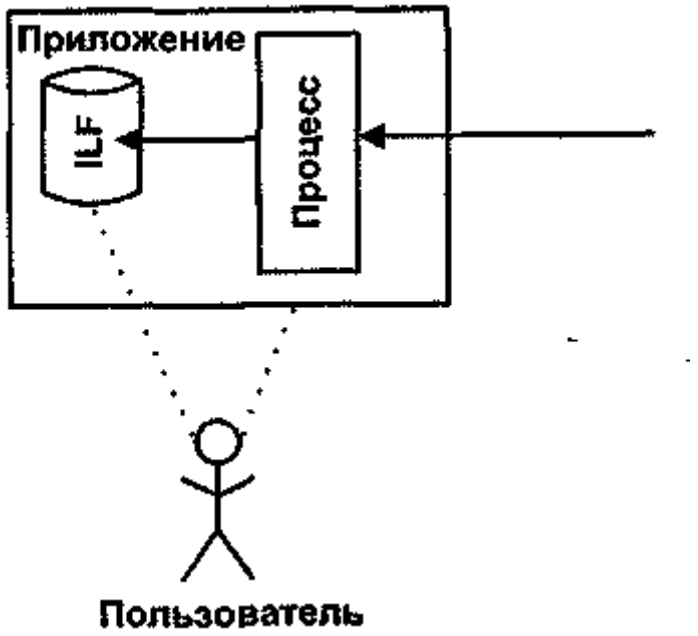
Выявление функциональных типов

Порядок расчета трудоемкости разработки ПО:
· определение количества и сложности функциональных типов приложения;
· определение количества связанных с каждым функциональным типом элементарных данных (DET), элементарных записей (RET) и файлов типа ссылок (FTR);
· определение сложности (в зависимости от количества DET, RET и FTR);
· подсчет количества функциональных точек приложения;
· подсчет количества функциональных точек с учетом общих характеристик системы (рис.6.3);
· оценка трудоемкости разработки (с использованием различных статистических данных).

Определение количества функциональных точек

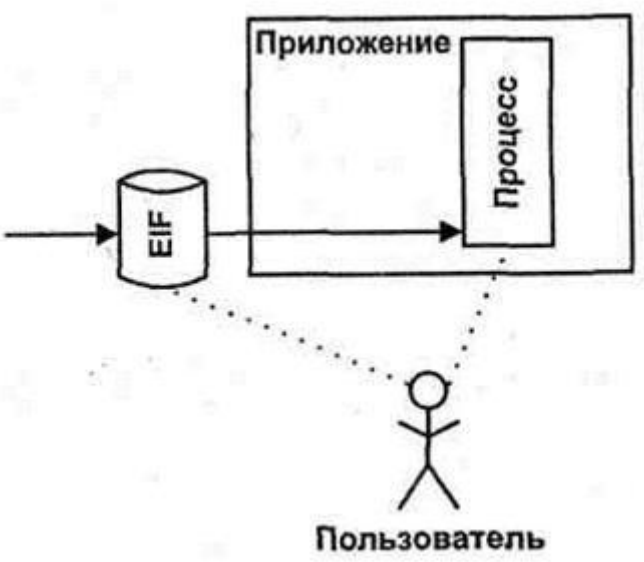
В состав функциональных типов (function type) включаются следующие элементы приложений разрабатываемой системы.

1. ***Внутренний логический файл (internal logical file, ILF)*** — идентифицируемая совокупность логически взаимосвязанных записей данных, поддерживаемая внутри приложения посредством элементарного процесса (рис. 6.4).



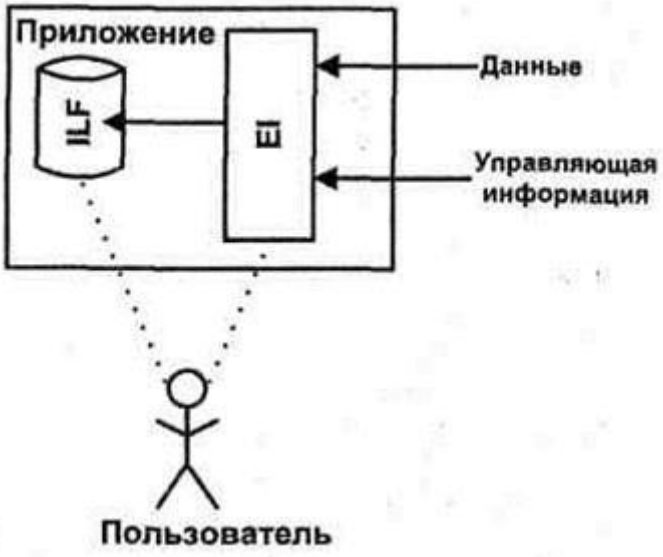
Внутренний логический файл

2. ***Внешний интерфейсный файл (external interface file, EIF)*** — идентифицируемая совокупность логически взаимосвязанных записей данных, передаваемых другому приложению или получаемых от него и поддерживаемых вне данного приложения (рис. 6.5).



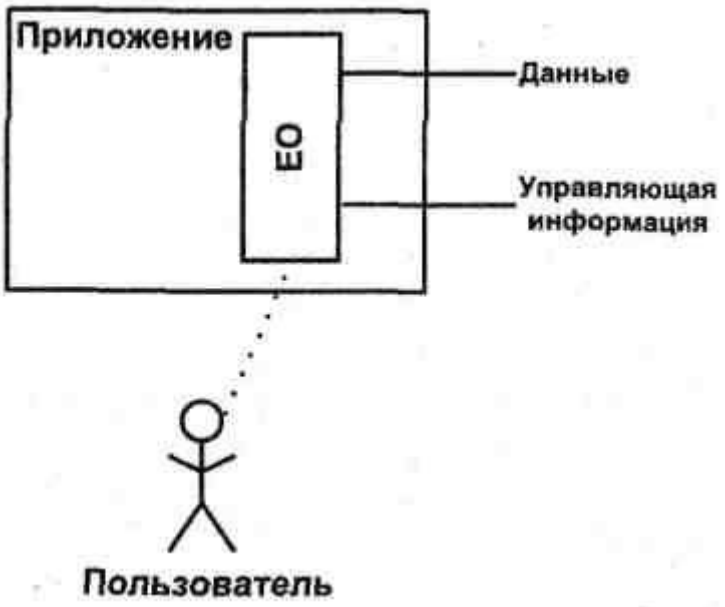
Внешний интерфейсный файл

3. ***Входной элемент приложения (external input, EI)*** — элементарный процесс, связанный с обработкой входной информации приложения - входного документа или экранной формы. Обрабатываемые данные могут соответствовать одному или более ILF (рис. 6.6).



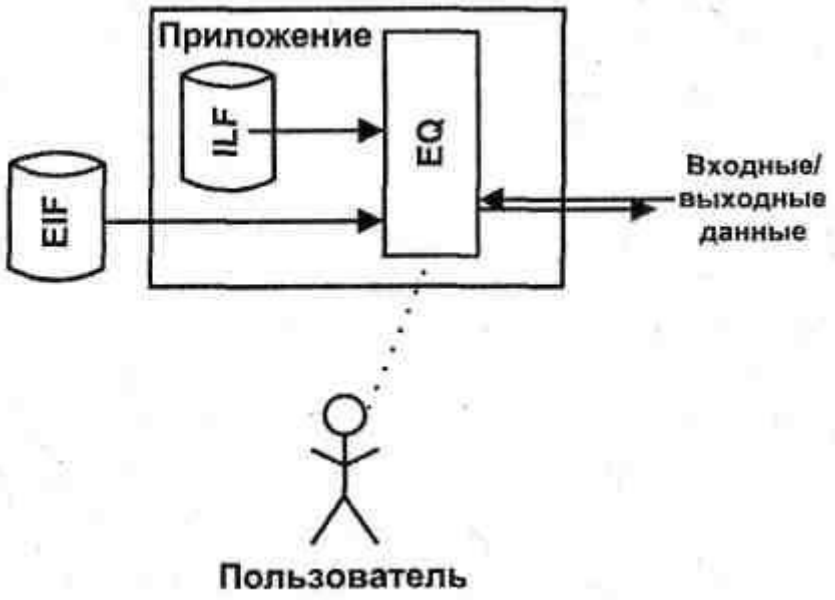
Входной элемент приложения

4. ***Выходной элемент приложения (external output, EO)*** — элементарный процесс, связанный с обработкой выходной информации приложения — выходного отчета, документа, экранной формы (рис. 6.7).



Выходной элемент приложения

5. ***Внешний запрос (external query, EQ)*** — элементарный процесс, состоящий из комбинации «запрос/ответ», не связанный с вычислением производных данных или обновлением ILF (базы данных) (рис. 6.8).



Внешний запрос

ФУНКЦИОНАЛЬНЫХ ТИПОВ ПО ДАННЫМ

Количество функциональных типов по данным (внутренних логических файлов и внешних интерфейсных файлов) определяется на основе диаграмм «сущность-связь» (для структурного подхода) и диаграмм классов (для объектно-ориентированного подхода). В последнем случае в расчете участвуют только устойчивые (persistent) классы, или классы-сущности.
Устойчивый класс соответствует ILF (если его объекты обязательно создаются внутри самого приложения) или EIF (если его объекты не создаются внутри самого приложения, а получаются в результате запросов к базе данных).

Примечание. Если операции класса являются операциями-запросами, то это характеризует его принадлежность к EIF.
Для каждого выявленного функционального типа (ILF и EIF) определяется его сложность (низкая, средняя или высокая). Она зависит от количества связанных с этим функциональным типом элементарных данных (data element types, DET) и элементарных записей (record element types, RET), которые в свою очередь определяются следующим образом:
· DET — уникальный идентифицируемый нерекурсивный элемент данных (включая внешние ключи), входящий в ILF или EIF;
· RET — идентифицируемая подгруппа элементов данных, входящая в ILF или EIF На диаграммах «сущность-связь» такая подгруппа обычно представляется в виде сущности-подтипа в связи «супертип-подтип».
Один DET соответствует отдельному атрибуту или связи класса. Количество DET не зависит от количества объектов класса или количества связанных объектов. Если данный класс связан с некоторым другим классом, который обладает явно заданным идентификатором, состоящим более чем из одного атрибута, то для каждого такого атрибута определяется один отдельный DET (а не один DET на всю связь). Производные атрибуты могут игнорироваться.
Повторяющиеся атрибуты одинакового формата рассматриваются как один DET
Одна RET на диаграмме устойчивых классов соответствует либо абстрактному классу в связи обобщения (generalization), либо классу — «части целого» в композиции, либо классу с рекурсивной связью «родитель-потомок» (агрегацией).
Зависимость сложности функциональных типов от количества DET и RET определяется следующей таблицей (табл. 6.1).

Сложность ILF и EIF

Количество RET	Количество DET		
1-19	20-50	51 +	
	Низкая	Низкая	Средняя
2-5	Низкая	Средняя	Высокая
6 +	Средняя	Высокая	Высокая

оличество транзакционных функциональных типов (входных элементов приложения, выходных элементов приложения и внешних запросов) определяется на основе выявления входных и выходных документов, экранных форм, отчетов, а также по диаграммам классов (в расчете участвуют граничные классы).

Далее для каждого выявленного функционального типа (EI, EO или EQ) определяется его сложность (низкая, средняя или высокая). Она зависит от количества связанных с этим функциональным типом DET, RET и файлов типа ссылок (file type referenced, FTR) - ILF или EIF, читаемых или модифицируемых функциональным типом.

Правила расчета DET для EI:

· каждое нерекурсивное поле, принадлежащее (поддерживаемое) ILF и обрабатываемое во вводе;
· каждое поле, которое пользователь хотя и не вызывает, но оно через процесс ввода поддерживается в ILF;
· логическое поле, которое физически представляет собой множество полей, но воспринимается пользователем как единый блок информации;
· группа полей, которые появляются в ILF более одного раза, но в связи с особенностями алгоритма их использования воспринимаются как один DET;
· группа полей, которые фиксируют ошибки в процессе обработки или подтверждают, что обработка закончилась успешно;
· действие, которое может быть выполнено во вводе.

Правила расчета DET для EO:

· каждое распознаваемое пользователем нерекурсивное поле, участвующее в процессе вывода;

· поле, которое физически отображается в виде нескольких полей его составляющих, но используемое как единый информационный элемент;

· каждый тип метки и каждое значение числового эквивалента при графическом выводе;

· текстовая информация, которая может содержать одно слово, предложение или фразу;

· литералы не могут считаться элементами данных;

· переменные, определяющие номера страниц или генерируемые системой логотипы не являются элементами данных.

Таблица 6.2

Сложность EI

Количество FTR	Количество DET		
1-4	5-15	16 +	
0-1	Низкая	Низкая	Средняя
	Низкая	Средняя	Высокая
3 +	Средняя	Высокая	Высокая

Таблица 6.3

Сложность EO

Количество FTR	Количество DET		
1-5	6-19	20 +	
0-1	Низкая	Низкая	Средняя
2-3	Низкая	Средняя	Высокая
4 +	Средняя	Высокая	Высокая

Сложность EQ определяется как максимальная из сложностей EI и EO, связанных с данным запросом.

ФУНКЦИОНАЛЬНЫЕ ТОЧКИ

Общая классификация сущностей

СУЩНОСТЬ	Общее число параметров сущности				
	1-3	4-6	7-16	17-35	36 и более
	2	5	7	10	13

Сущности – функции, процедуры, таблицы БД,…

Общая классификация функций ввода

ВВОД Общее число сущностей, для которых вводимые параметры являются входными	Общее число вводимых параметров				
	1-2	3-4	5-11	12-19	20 и более
1	1	1	3	3	4
2	1	3	4	4	6
3-4	3	4	4	4	6
5-7	3	4	4	6	10
8 и более	4	6	6	10	10

Общая классификация функций вывода

ВЫВОД Общее число сущностей, которые задействованы в процессе подготовки выводимых параметров	Общее число выводимых параметров				
	1-4	5-10	11-18	19-26	27 и более
1	2	2	4	4	5
2-3	2	4	5	5	7
4-5	4	5	5	5	7
6-9	4	5	5	7	11
10 и более	5	7	7	11	11

Общая классификация интерфейсов

ИНТЕРФЕЙСЫ	Общее число параметров интерфейса				
	1-4	5-9	10-30	31-60	Более 60
	2	5	7	10	13

Общая классификация запросов

Запросы Общее число сущностей, которые задействованы в процессе обработки запроса	Общее число параметров запроса				
	1	2-3	4-10	11-19	20 и более
1	1	1	3	3	4
2-3	1	3	4	4	6
4-5	3	4	4	4	6
6-7	3	4	4	6	10
8 и более	4	6	6	10	10

Общая классификация функций Алгоритм

АЛГОРИТМ число элементарных блоков обработки данных	Общее число обрабатываемых параметров				
	1-3	4-6	7-12	13-19	30 и более
1- 4	1	1	3	3	4
5-7	1	3	4	4	6
8-10	3	4	4	4	6
11-15	3	4	4	6	10
16 и более	4	6	6	10	10

Определение суммарной оценки в *fp*

$$FP_{\Sigma} = FP_I + FP_O + FP_Q + FP_A + FP_E + FP_F$$

$$FP_X = \sum_i fp_i$$

- FP_I - трудоемкость функций Ввод,
- FP_O - трудоемкость функций Вывод
- FP_Q - трудоемкость функций Запрос
- FP_A - трудоемкость функций Алгоритм
- FP_E - трудоемкость функций Сущность
- FP_F - трудоемкость функций Интерфейс.

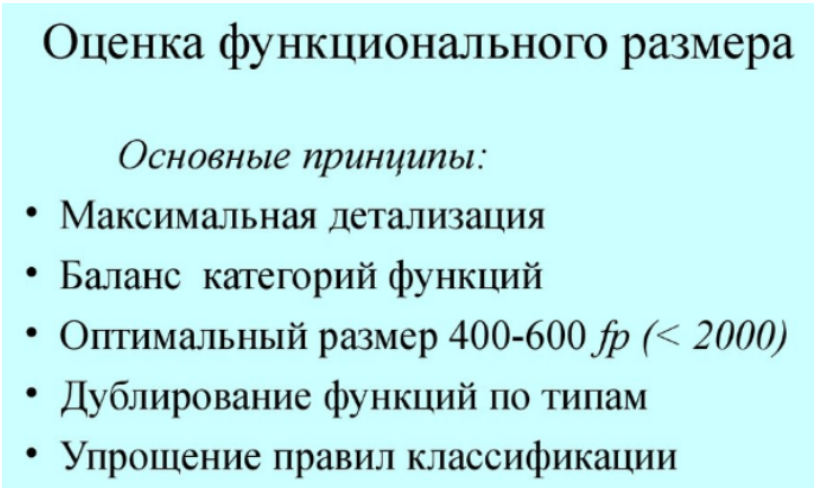
Оценка функционального размера

ДОСТОИНСТВА

- Детализация программы
- Понятна заказчику и разработчику
- Слабо зависит от инструмента разработки
- Оценка производительности труда

НЕДОСТАТКИ

- Трудность в детализации функций
- Точная оценка требует большой статистики
- Плохо оценивает большие проекты



Международные стандарты на основе Function Point.

ISO Standards

- FiSMA: ISO/IEC 29881:2010 Information technology – Systems and software engineering – FiSMA 1.1 functional size measurement method.
- IFPUG: ISO/IEC 20926:2009 Software and systems engineering – Software measurement – IFPUG functional size measurement method.
- Mark-II: ISO/IEC 20968:2002 Software engineering – MI II Function Point Analysis – Counting Practices Manual
- Nesma: ISO/IEC 24570:2018 Software engineering – Nesma functional size measurement method version 2.3 – Definitions and counting guidelines for the application of Function Point Analysis
- COSMIC: ISO/IEC 19761:2011 Software engineering. A functional size measurement method.
- OMG: ISO/IEC 19515:2019 Information technology — Object Management Group Automated Function Points (AFP), 1.0

The first five standards are implementations of the over-arching standard for Functional Size Measurement ISO/IEC 14143.[2] The OMG Automated Function Point (AFP) specification, led by the Consortium for IT Software Quality, provides a standard for automating the Function Point counting according to the guidelines of the International Function Point User Group (IFPUG) However, the current implementations of this standard have a limitation in being able to distinguish External Output (EO) from External Inquiries (EQ) out of the box, without some upfront configuration.[3]

Лабораторные работы

Лабораторная работа №1. Стратегическое планирование информационных систем

Цель работы: Разработка стратегического плана автоматизации компании.

Полученные знания и навыки: В результате выполнения лабораторной работы студенты должны получить знания и навыки по разработке стратегического плана автоматизации компании.

Теоретические основы

Автоматизация - применение технических средств, освобождающих человека частично или полностью от непосредственного участия в процессах получения, преобразования, передачи и использования энергии, материалов и информации. Автоматизация управления направлена на использование компьютеров и других технических средств обработки и передачи информации в управлении производством, экономикой.

Стратегический план автоматизации компании содержит основные принципы и условия, с соблюдением которых должно осуществляться принятие решений на каком-либо отрезке времени, и результаты, которые должны быть достигнуты при соблюдении этих условий.

Стратегия автоматизации должна соответствовать приоритетам и задачам бизнеса компании и включать пути достижения этого соответствия. Поэтому стратегия автоматизации основывается на стратегии бизнеса компании и представляет собой план, согласованный по срокам и целям со стратегией компании с учетом ограничений.

Стратегия автоматизации должна содержать: цели автоматизации; способ автоматизации; ограничения; требования к информационной системе; способ приобретения информационной системы (ИС).

Цели автоматизации соответствуют целям бизнеса компании и включают области деятельности компании и последовательность, в которой они будут автоматизированы.

Способами автоматизации являются хаотичная, по участкам, по направлениям, полная и комплексная автоматизация, которые имеют свои преимущества и недостатки.

Ограничениями, которые необходимо учитывать при выборе стратегии автоматизации компании, являются финансовые, временные, трудовые и технические.

Финансовые ограничения определяются величиной инвестиций, которые компания способна сделать в развитие автоматизации.

Временные ограничения могут быть связаны со сменой технологий основного производства, стратегией бизнеса компании (временные ограничения в стратегии компании), государственным регулированием экономики.

Трудовыми ограничениями может быть отношение персонала к автоматизации, привычка работать по стандартизированным процедурам и исполнительская дисциплина; особенности рынка труда (безработица, недостаток квалифицированных специалистов и т. п.).

Технические ограничения связаны с реальными возможностями компании (например, отсутствие помещений для размещения компьютеров, ограничения по использованию определенного вида оборудования и т. п.).

При выборе стратегии автоматизации существенную роль играет состояние информационных технологий. Существуют следующие способы приобретения ИС: покупка готовой ИС; разработка ИС (самостоятельно или с помощью специализированной фирмы-разработчика ИС), если необходимой ИС нет на рынке; покупка ядра ИС и его доработка под потребности компании; аутсорсинг ИС.

При выборе ИС основным критерием ее оценки должен быть критерий удовлетворения потребностей бизнеса компании. Потребности бизнеса формулируются в терминах бизнеса, например снижение себестоимости продукции и издержек; сокращение трудозатрат; рост объемов продаж; укрепление и расширение своих позиций на рынке; сокращение длительности основных производственных циклов; улучшение контроля над выполняемыми операциями; изучение и максимальное удовлетворение потребностей клиентов и т.д.

При выборе ИС потребности бизнеса преобразуются в технические и экономические требования к информационной системе: функциональные возможности; совокупная стоимость владения; перспективы развития, поддержки и интеграции; технические характеристики. Функциональные возможности ИС должны соответствовать основным бизнес-процессам, которые существуют или планируются к внедрению в компании.

Перспективы развития и поддержки ИС в основном определяются поставщиком решения и тем комплексом стандартов, который заложен в ИС и составляющие ее компоненты. Возможность интеграции с другими системами определяется совокупностью поддерживаемых информационной системой стандартов.

Устойчивость поставщика ИС и поставщиков отдельных компонентов определяется временем существования их на рынке и долей рынка, которую они занимают. Важным фактором является форма, в которой осуществляется присутствие поставщика ИС на российском рынке: наличие сети сертифицированных центров технической поддержки, авторизованных учебных центров, "горячих линий" для консультаций и т.д.

К техническим характеристикам информационной системы относятся: архитектура системы; масштабируемость; надежность; способность к восстановлению при сбоях оборудования; наличие средств архивирования и резервного копирования данных; средства защиты от преднамеренных и непреднамеренных технических нападений; поддерживаемые интерфейсы для интеграции с внешними системами. Технические характеристики влияют на такие параметры системы, как возможность наращивания при необходимости функциональных возможностей и увеличение числа пользователей ИС.

В качестве критериев выбора стратегии автоматизации выступают различия таких реальных и ожидаемых показателей, как время и затраты на внедрение; экономический эффект

от внедренных систем; влияние системы на условия труда или конкурентоспособность компании.

Задание на лабораторную работу. Разработка стратегического плана автоматизации компании.

Разработка стратегического плана автоматизации компании.

1. Описание ситуации в компании. Составить представление о деятельности компании, выбранной в качестве базовой для выполнения КНИР и магистерской диссертации.

2. Описать характер деятельности компании, сложившуюся в ней проблемную ситуацию в сфере информатизации - цели и задачи бизнеса компании (например: снижение стоимости продукции; увеличение количества или ассортимента; сокращение цикла разработки новых товаров и услуг; переход от производства на склад к производству под конкретного заказчика с учетом индивидуальных требований и т. д.).

3. Описание целей автоматизации. Цели автоматизации должны соответствовать целям бизнеса компании, т.е. функциям, которые необходимо автоматизировать для решения проблемы компании. Последовательность автоматизации выделенных функций. Преимущества, которые даст автоматизация выделенных функций компании.

4. Выбор способа автоматизации компании и обоснование выбора.

4.1. Перечислить возможные способы автоматизации (хаотичная, по участкам, по направлениям, полная, комплексная автоматизация) и описать преимущества и недостатки каждого способа автоматизации.

4.2. Описать существующий в компании способ автоматизации и недостатки данного способа автоматизации для компании.

4.3. Проанализировав преимущества и недостатки всех существующих способов автоматизации, выбрать из них один для данной конкретной компании и обосновать свой выбор (на основании чего выбран способ автоматизации, каковы преимущества способа автоматизации для данной компании).

5. Описание ограничений. Описать ограничения, которые необходимо учитывать при выборе стратегии автоматизации компании.

5.1. Финансовые - определить величину инвестиций, которые компания способна сделать в развитие автоматизации.

5.2. Временные - определить, в какие сроки необходимо осуществить автоматизацию.

5.3. Трудовые - описать возможные ограничения, связанные с влиянием человеческого фактора (отношение персонала компании к автоматизации; новые процедуры работы, которые могут потребоваться после автоматизации; увеличение нагрузки на персонал в первое время работы ИС; необходимость обучения персонала; прием дополнительного персонала после автоматизации; перестановки персонала после автоматизации и т.д.).

5.4. Технические - описать возможные ограничения, связанные с реальными возможностями предприятия (отсутствие помещений для размещения оборудования, ограничения по использованию определенного вида оборудования и т.п.).

6. Анализ требований к ИС.

6.1. Описать функции, которые должна выполнять будущая система (то, что нужно автоматизировать).

6.2. Перечислить основные классы ИС (MRPII, ERP, CRM, OLAP и др.) и кратко охарактеризовать структуру, функциональные возможности, преимущества и недостатки внедрения ИС различных классов.

6.3. Обосновать выбор класса ИС, подходящий для внедрения в данной конкретной компании в соответствии с требованиями к будущей ИС и определенными выше ограничениями.

7. Выбор способа приобретения ИС.

7.1. Описать способы приобретения ИС (самостоятельная разработка, покупка готовой). Оценить каждый способ приобретения ИС, описать его преимущества и недостатки.

7.2. Обосновать способ приобретения ИС для рассматриваемой компании: описать возможности и потребности компании (наличие отдела ИТ, наличие денежных средств, персонала, времени, потребности в функционале, наличие требуемой ИС на рынке и т.д.). Принять решение о способе приобретения ИС.

8. Рассмотрение варианта покупки ИС (если такое решение принято в п.8).

8.1. Выполнить с помощью Интернет обзор ИС, в которых реализована

автоматизация необходимых функций, выявленных в процессе анализа требований к ИС.

8.2. В результате обзора составить список ИС, в которых реализованы необходимые функции (3-5 информационных систем).

8.3. Выделить критерии оценки информационных систем (функциональные возможности; стоимость; перспективы развития, поддержки и интеграции; технические характеристики).

8.4. Описать функциональные возможности каждой ИС.

8.5. Описать соответствие функциональных возможностей каждой ИС бизнес-функциям компании.

8.6. Рассчитать стоимость приобретения каждой ИС.

8.7. Описать перспективы развития, поддержки и интеграции каждой ИС. 8.8. Оценить устойчивость каждого поставщика ИС (т.е. определить время

существования их на рынке; определить долю занимаемого рынка; наличие сети сертифицированных центров технической поддержки; авторизованных учебных центров; "горячих линий" для консультаций и т.д.)

8.9. Оценить преимущества и недостатки каждой ИС, сопоставив полученные данные, и выбрать наиболее подходящую ИС по выделенным критериям.

9. Составить отчет.

Лабораторная работа №2. Планирование проектной деятельности.

Цель работы: получить навыки по первичному планированию.

Полученные знания и навыки: в результате выполнения лабораторной работы студенты должны получить практические навыки по описанию программных систем.

Задание на лабораторную работу. Планирование проектной деятельности.

1) Произвести изучение программных систем по планированию.

2) По предложенной структуре описания произвести описание не менее трех систем. Пункты описания системы:

- назначение и область применения (также необходимо отнести систему к соответствующему классу);
- описание объекта проектирования в виде ER-диаграмм (подробная структура представления объекта с указанием компонент объекта, их структуры и связей);
- описание процесса проектирования (в виде IDEF0, UML-диаграмм или блок-схем);
- особенности некоторых видов обеспечения (необходимо выделить те виды обеспечения, которые имеют особенности, например свой язык проектирования, сложная структура БД, математическое обеспечение и алгоритмы),
- типы и структуры входных/выходных файлов;
- описание достоинств и недостатков системы.

3) Описать тестовый пример с детализированным (на уровне работ и операций по созданию документов и проектных процедур – по ГОСТ) собственным планом работы по выполнению магистерской работы. Необходимо планировать работу нескольких исполнителей в различных реально выполняемых ролях и преподавателей по проверке и выполнению задания.

Приложения:

- системы OpenProj, Corel TimeLine, MS Project, CA Super Project и пр. в интернете.
- Задание на выполнение маг. работы.
- План выполнения (составлен исполнителем в начале семестра).

Лабораторная работа №3. Анализ рисков по методу PERT.

Цель лабораторной работы: овладеть инструментальными средствами анализа рисков по методу PERT, поддерживаемыми Microsoft Office Project .

Теоретические основы

Анализ опасностей, которые могут возникнуть при выполнении составленного плана, — один из самых интересных и сложных этапов планирования проекта. От того, как проведен анализ, зависит, будет ли проект успешно завершен. В этом уроке вы научитесь определять риски с помощью MS Project, описывать их и разрабатывать стратегии их смягчения. Для проведения анализа мы задействуем все имеющиеся в нашем арсенале средства: настраиваемые поля, формулы, стандартные и настраиваемые фильтры, сортировки. Анализ рисков состоит из нескольких этапов. Сначала нужно определить возможные риски. Затем для каждого из них нужно определить стратегию смягчения влияния риска на проект, то есть действия, предпринимаемые для предотвращения риска или в случае осуществления риска для того, чтобы проект был успешно завершен.

Определение рисков.

Часто в процессе определения рисков невозможно детально проанализировать весь план проекта в разумное время (например, если план состоит из нескольких сотен задач). В таких случаях в первую очередь нужно анализировать риски у задач, которые находятся на критическом пути проекта или могут стать критическими. Чтобы определить, какие задачи могут стать критическими, можно воспользоваться оптимистической и пессимистической диаграммами Ганта, полученными в результате анализа методом PERT.

При определении рисков информацию нужно заносить в план проекта. Для этого нужно:

· Переименовать поле «Текст2» в «Описание риска

· «Текст3» в «Вероятность осуществления риска», причем для последнего мы создали список значений: «Высокая», «Средняя» и «Низкая»

· На основании таблицы «Ввод» для задач мы создали таблицу «Ввод» информации о рисках и оставили в ней лишь необходимый набор полей.

· И, наконец, на базе таблицы мы создали два представления: «Риски», в котором эта таблица находится рядом с диаграммой Ганта

· И комбинированное представление «Риски2», в верхней части которого находится представление «Риски», а в нижней – «Форма задач». Теперь можно переходить к определению рисков.

Риски определяются для трех аспектов проекта: «Расписания», «Ресурсов» и «Бюджета». Так выявляются события, осуществление которых может помешать завершить проект в срок или создать нехватку ресурсов или денег в определенный момент его выполнения. Если при

определении риска становится ясно, как уменьшить его, то нужно сразу же вносить соответствующие изменения в план проекта.

Риски в расписании.

Задача, стоящая перед руководителем проекта при анализе рисков расписания, заключается в том, чтобы уменьшить вероятность срыва сроков работ. Срыв сроков работ может произойти в том случае, если длительности задач в плане не будут соответствовать тому времени, которое потребуется ресурсам на их выполнение. Несоответствие запланированных длительностей работ фактическим может произойти в двух случаях: если неточно составлен план проекта и если неожиданно окажется, что та или иная работа требует больше времени, чем ожидалось. Поскольку каждый проект уникален, то обязательно случится так, что какая-то из задач будет длиться дольше запланированного времени, но чем точнее и детальнее план, тем меньше будет таких задач. Ведь при неточном плане несоответствия возникают даже тогда, когда их могло бы и не быть. Поэтому уменьшение рисков в расписании начинается с детализации плана работ. Затем нужно обнаружить задачи, у которых вероятность срыва наиболее велика. Эти задачи можно обнаружить по некоторым формальным критериям, рассматриваемым ниже.

Задачи с предварительными длительностями.

Один из наибольших рисков представляют задачи, в выполнении которых у сотрудников нет опыта. Главная проблема в планировании таких задач заключается в том, что их длительность не известна заранее, поскольку нет опыта в их выполнении. Поэтому обычно при планировании длительность этих задач остается предварительной. Такие задачи можно обнаружить в плане проекта с помощью стандартного фильтра «Задачи с оценкой длительности». В нашем плане таких задач нет, но если бы они обнаружились, то пришлось бы изменить план проекта таким образом, чтобы неожиданное увеличение их длительности не сказалось на сроках окончания проекта или на сроках исполнения важных задач (например, тех, у которых сроки исполнения регламентируются договором). Желательно увеличить планируемую длительность исполнения этих задач до пессимистичной и рассчитывать план с учетом этой длительности задач. Кроме того, можно добавить в план отдельную задачу по освоению нового оборудования или технологии раньше того, как начнется выполнение задачи, где это оборудование или технология будет использоваться.

Слишком короткие задачи.

Часто при планировании проекта длительность задач определяется на основании оценки будущих исполнителей. Например, руководитель проекта просит сотрудника оценить, сколько времени ему потребуется на исполнение определенной задачи, а затем оценка сотрудника заносится в план. Сотрудники же часто дают слишком оптимистичные сроки, что приводит к тому, что запланированные работы не удается выполнить в срок или сотруднику приходится работать сверхурочно.

Другой источник задач со слишком короткими сроками - сами менеджеры, выделяющие на задачу столько, сколько считают нужным (исходя из ограничений по срокам проекта), не советуясь при этом с потенциальными исполнителями.

Чтобы избежать таких случаев, нужно проанализировать все задачи плана проекта длительностью меньше одного дня (кроме вех) и все задачи, у которых при анализе PERT ожидаемая длительность совпадала с оптимистичной. Для этого создадим новый фильтр и настроим. Для этого нужно:

1. В меню «Проект» выберите команду «Фильтр», а затем щелкните «Другие фильтры»;
2. Щелкните «Создать» и введите имя «Слишком короткие задачи»;
3. В столбце «И/Или» установите «Или» для поля «Длительность1»;
4. В столбце «Имя поля» выберете «Длительность» и «Длительность1»;
5. В столбце «Проверка» для поля «Длительность» установите «Меньше или равно» , для поля «Длительность1» установите равно «Равно»;
6. В столбце «Значения» для поля «Длительность» установите «1д», для поля «Длительность1» установите «[Длительность2]»;
7. Аналогичным способом создайте условие для вех, в столбце «И/Или» установите «И», в столбце «Имя поля» выберете «Веха», столбце «Проверка» установите «Равно», в столбце «Значения» установите «Нет»;
8. Нажмите кнопку «Ok», а затем кнопку «Применить».

Фильтр отбирает задачи, у которых длительность меньше либо равна одному дню или значение настраиваемого поля «Длительность» равно значению настраиваемого поля «Длительность2» (эти настраиваемые поля используются при анализе по методу PERT для хранения информации об оптимистической и ожидаемой длительности). Среди задач, отобранных по одному из этих критериев, фильтр отбирает те задачи, у которых значение поля «Веха» равно «Нет», то есть задачи, не являющиеся вехами. В нашем случае коротких задач оказалось 8 («Определение проекта», «Определение тематики и организации контента», «Разработка шаблонов (дизайна)№1», «Разработка шаблонов (дизайна)№2», «Разработка графического контента (кнопки, логотипы)», «Тест функционала», «Внешнее тестирование», «Регистрация в поисковых системах»). После того как короткие задачи отобраны, определим реалистичность отведенного на них времени. Если мы обнаруживаем в плане задачи, имеющие неоправданно короткие сроки, то длительность таких задач нужно дополнительно обсудить с будущими исполнителями. При этом желательно запросить у них все три возможных срока исполнения задачи, чтобы внести их в таблицу для анализа PERT и рассчитать длительность задачи.

Слишком длинные задачи и задачи с большим числом ресурсов.

Мы уже говорили о том, что при составлении плана стоит избегать слишком длинных задач. Как правило, без детализации работ очень сложно точно оценить трудозатраты для таких задач и возможную загрузку ресурсов, поэтому, включая их в план, вы повышаете вероятность того, что он окажется неточным.

Обнаружить в плане задачи с большой длительностью очень просто. Достаточно воспользоваться автофильтром и отфильтровать задачи по столбцу «Длительность», отобрав задачи с длительностью, превышающей, например, 5 пли 10.

Оптимистическая длительность может совпадать с ожидаемой не точно, а с определенным допущением, например, различаться на 1 или 2 часа. Чтобы такие задачи тоже можно было обнаружить, в этом же файле мы создали фильтр «Слишком короткие задачи - 2», в котором можно внести это допущение.

А вот автоматически отобрать задачи с большим числом ресурсов нельзя, поскольку в MS Project нет специального столбца «внутренней» таблицы, в котором было бы указано число ресурсов, назначенных на задачу. Поэтому нам, как обычно, придется воспользоваться настраиваемым полем. Переименуем поле задач «Число2» в «Число ресурсов» и поместим в него формулу «(Len ([Названия ресурсов]))». Функция Len определяет длину текстовой строки, переданной ей в качестве параметра. В нашем случае этой строкой является значение поля «Названия ресурсов». Чем больше ресурсов назначено на задачу, тем длиннее строка и тем больше будет значение поля Число ресурсов.

Этот метод сравнения задач довольно груб, поскольку не гарантирует точного сравнения числа ресурсов. Точно определить число назначенных на задачу ресурсов можно лишь с помощью макроса (функции этого не позволяют).

После завершения настройки поля отсортируем задачи по этому полю. Для этого с помощью команды меню «Проект > Сортировка > Сортировать по» откроем диалоговое окно сортировки и выберем созданное поле в качестве критерия. Сортировать задачи будем по убыванию, чтобы задачи с наибольшим числом ресурсов оказались в верхней части списка, и сбросим флажок «Сохранить структуру», чтобы сортировка осуществлялась в рамках всего проекта, а не в рамках отдельных фаз.

Определив задачи с большими длительностями или большим числом назначенных ресурсов, нужно разбить их на серию более коротких задач или превратить в фазы, поскольку, как правило, в рамках длинной задачи решается несколько коротких. Еще одно подтверждение тому - много назначений на задачу: как правило, над решением одной задачи работает не больше двух человек, а если их назначено больше, то это значит, что задача может быть разделена на несколько составляющих.

Список всех предшественниц задачи приведен в поле «Предшественники», причем номера задач-предшественниц разделены точками. И если в этом поле встречается хотя бы одна точка с запятой, значит, у задачи есть как минимум две предшественницы. Поэтому наш фильтр будет отбирать те задачи, у которых в поле «Предшественники» содержится точка с запятой.

В результате работы фильтра важно не только обнаружить задачи с несколькими предшественницами, но и понять, как эта задача связана с другими задачами в плане проекта. Поэтому созданный фильтр удобнее всего применять в режиме подсветки, чтобы задачи с несколькими зависимостями лишь подсвечивались среди всех. После того как задачи с несколькими зависимостями обнаружены, нужно определить, как можно уменьшить риск их задержки. Уменьшить риск можно, увеличив длительности одной или нескольких задач-предшественниц за счет более раннего их начала (если это возможно). Кроме того, можно увеличить запланированную длительность задачи, если ограничения по длительности проекта позволяют это сделать. Иногда одна из двух задач начинается намного позже другой, и тогда она создает временной резерв другим. Создавать такие резервы можно, когда дата начала одной из задач-предшественниц связана с другой задачей или имеет ограничение, а у другой задачи

такого ограничения нет. Если перенести задачу, дату начала которой ничто не ограничивает, на более ранний срок, то это создаст ей временной резерв.

В нашей случае нет задач, имеющих более одного предшественника, поэтому при применении фильтра мне не увидим результата.

Задачи с внешними зависимостями.

Иногда задачи зависят от внешних по отношению к проекту событий, не использующих проектные ресурсы и не поддающихся планированию. Например, если организация выполняет два взаимосвязанных проекта, то в качестве предшественника задачи может выступать задача из другого проекта. Определить такие задачи с помощью фильтра можно лишь в том случае, если в качестве предшественников выступают задачи, хранящиеся в других файлах проектов. В таком случае для обнаружения этих задач нужно настроить фильтр, созданный нами для определения задач с несколькими предшественницами, заменив символ «;» на «\». Бывает и так, что у задачи нет предшественниц в других файлах проектов, но, тем не менее, внешние зависимости у нее есть. Обычно такие задачи может определить лишь менеджер при анализе плана вручную. Чтобы эти задачи можно было определить на формальной основе, при создании списка задач можно добавить настраиваемое поле типа «Флаг» и изменять его значение для задач с внешними зависимостями.

Ресурсные риски.

Цель анализа ресурсных рисков заключается в том, чтобы определить ресурсы и назначения, увеличивающие вероятность срыва проекта. Например, рискованно привлечение недавно принятого на работу сотрудника, поскольку у нас нет опыта работы с ним, и мы не знаем, сможет ли он справиться с поставленными задачами. Другой риск — использование одного сотрудника в слишком многих задачах, поскольку проект становится зависимым от одного сотрудника, и если он станет недоступным, то проект может провалиться.

Использование неопытных сотрудников.

Часто случается так, что для проектных работ привлекаются сотрудники, недавно вступившие в организацию. Поскольку еще нет опыта использования этих сотрудников в проектах, это представляет определенный риск. Нужно определить задачи, где задействованы эти сотрудники, и описать риск их использования. При разработке стратегии смягчения рисков эти риски нужно будет проанализировать и определить, как их уменьшить. Чтобы выделить сотрудников без опыта работы, настроим столбец «Флаг2», назвав его «Опыт есть», и определим отображение красного индикатора для тех случаев, когда значением поля является «Нет», и зеленого - когда значением является «Да». Добавим настроенное поле в представление «Лист ресурсов» и установим в нем значение «Нет» для тех ресурсов, у которых нет опыта работы. В нашем случае в проекте задействованы только два ресурса без опыта: Семенов и Сергеева. Теперь разделим окно, отобразим в нижней части представление «Использование задач» и откроем таблицу «Ввод информации о рисках». Для того чтобы в ней отобразились только те задачи, в которых задействованы неопытные сотрудники, выделим этих сотрудников в списке в верхнем представлении, щелкнув на их фамилиях при нажатой клавише «Ctrl».

Ресурсы с большим объемом работ.

Иногда загрузка между участниками проекта распределяется неравномерно, и некоторые из членов команды делают больший объем работы, чем другие. Если не проконтролировать распределение работы, то может оказаться, что некоторые сотрудники отвечают за исполнение слишком большого числа задач. Слишком высокая ответственность отдельных сотрудников опасна тем, что в случае болезни такого «ключевого» сотрудника или недоступности его по другой причине выполнить все задачи в срок будет невозможно. Определить ресурсы с большим числом назначений можно с помощью представления «Использование ресурсов». Откроем в этом представлении таблицу «Трудозатраты» и отберем для отображения только человеческие ресурсы, воспользовавшись фильтром «Ресурсы - трудовые». Затем отсортируем ресурсы по убыванию по колонке «Трудозатраты». Теперь участники проекта с наибольшей загрузкой отображаются в начале списка. Для того чтобы просмотреть, какое место в плане проекта занимают назначения наиболее занятых сотрудников, разделим окно и в нижнем представлении отобразим диаграмму Ганта. Теперь при выборе ресурса в верхнем представлении в нижнем отображаются все его назначения, как в таблице, так и на диаграмме. Критические задачи выделены красным, и чем в большем числе критических задач задействован ресурс, тем выше опасность срыва сроков проекта, если этот ресурс вдруг перестанет быть доступным. Поскольку в этом случае риск, связанный с задействованностью ресурса, распространяется на все задачи, в которых он участвует, то нет смысла заполнять поля с описанием риска для задач - удобнее создать аналогичные настраиваемые поля для ресурсов и вводить информацию в них. Чтобы внести в план информацию о ресурсных рисках и использовать ее в дальнейшем при разработке стратегии смягчения рисков, изменим настраиваемые поля для ресурсов «Текст2» и «Текст3». Переименуем их в «Описание риска» и «Вероятность осуществления риска». Поскольку во втором поле можно использовать список значений, уже составленный нами в аналогичном поле для задач, импортируем его с помощью кнопки «Импорт настраиваемого поля».

В поле «Текст2» могут вводиться одинаковые риски для разных ресурсов, поэтому настроим список значений таким образом, чтобы при вводе можно было указывать значения, не входящие в список, и они автоматически добавлялись бы в него для дальнейшего. Создадим новую таблицу на базе ресурсной таблицы «Ввод», назовем ее «Ввод информации о рисках ресурсов» и добавим в нее настроенные поля. Теперь откроем ее в верхнем представлении и заполним ее данными для ресурсов, выполняющих большой объем работы. Наиболее «рискованными» ресурсами проекта являются Таратухин, Бурков и Васин, задействованные в самом большом числе задач, большинство из которых лежит на критическом пути. Соответственно, в поле «Описание риска» введем «Срыв работ из-за недоступности ресурса», а в поле «Вероятность осуществления» выберем значение «Высокая».

Ресурсы со сверхурочной работой.

Сотрудники, загруженные сверхурочной работой, из-за усталости могут начать работать медленнее, чем обычно. Поэтому при планировании стоит избегать использования сверхурочной загрузки. Если же при составлении плана вам пришлось запланировать сверхурочную работу, то при анализе рисков стоит предусмотреть ее возможные последствия. Для анализа мы будем использовать то же представление, что и в предыдущем примере, но на диаграмме использования ресурсов отобразим детальные данные о превышении нагрузки и сверхурочных. Пролистывая эту диаграмму, можно быстро обнаружить ресурсы со сверхурочной нагрузкой. В нашем примере сверхурочная загрузка есть у Жукова, и поэтому

укажем в описании риска «Срыв работ из-за усталости ресурса». Но поскольку объем сверхурочной работы небольшой, то вероятность осуществления риска оценим как среднюю.

Сотрудники с уникальными навыками и материалы с единственными поставщиками.

Проект может оказаться под угрозой срыва, если неожиданно станет недоступен сотрудник, обладающий особыми знаниями или навыками, поскольку только он может выполнить определенные задачи проекта. Кроме того, риск провала проекта из-за несвоевременной поставки материалов повышается, если материалы могут быть получены только от одного поставщика, поскольку в этом случае выполнение проекта становится зависимым от качества его работы. Чтобы определить такие ресурсы и внести в план информацию о рисках, связанных с их использованием, откроем представление «Лист ресурсов» и отобразим в нем таблицу «Ввод информации о рисках ресурсов». Затем нужно определить риски с уникальными знаниями и ввести в таблицу описание рисков и вероятность их осуществления. Поскольку для выполнения проекта необходимо специализированное ПО, то при его поломке работа над проектом может остановиться, отразим это в столбце «Описание риска» и укажем «Вероятность осуществления риска» - «Низкая». Среди сотрудников только Терехов обладает уникальными знаниями, и его отсутствие может сказаться на сроках исполнения работ. Поэтому и для него мы укажем соответствующий риск, оценив степень вероятности его осуществления как среднюю. В нашем проекте задействовано не так много ресурсов, и поэтому просмотреть весь список и внести информацию о рисках можно довольно быстро. Если же проект, в котором вы оцениваете ресурсные риски, содержит большое число ресурсов, то при их анализе стоит воспользоваться стандартными фильтрами «Ресурсы — материальные» и «Ресурсы — трудовые», с помощью которых можно отобрать для анализа только сотрудников или только материалы.

Бюджетные риски.

В результате осуществления рисков возможно увеличение объема работы по проекту, что приведет к росту затрат на него. Риск увеличения бюджета проекта стоит рассматривать тогда, когда проект имеет ограниченные бюджетные рамки. Например, в нашем проекте задействованы в основном штатные сотрудники организации, регулярно получающие зарплату, и бюджет проекта не имеет большого значения. Бывают и другие случаи: например, проект может выполняться на заказ, и заказчик может выделять на выполнение работ определенную сумму, которую нельзя превысить. В тех случаях, когда затраты на проект ограничены, важно предусмотреть риск увеличения бюджета в результате тех или иных обстоятельств. Для оценки возможного увеличения бюджета можно применять различные методики. Мы продемонстрируем здесь оценку возможного изменения стоимости проекта на основании данных, полученных в ходе анализа PERT. Наш анализ исходит из предположения, что при увеличении длительности задачи объем работ всех назначенных ресурсов и, соответственно, цена возрастают пропорционально. Например, если задача длится 2 дня и стоит $100, то при увеличении длительности до 4 дней стоимость возрастет до $200. Этот метод оценки не очень точен, но он и не претендует на точность. Ведь при планировании рисков сложно предсказать, как именно будут задействованы ресурсы при увеличении длительности назначения. Задача анализа - определить возможный бюджет проекта при неблагоприятном развитии событий и задачи, цена которых сильно увеличится при осуществлении рисков.

Переименуем таблицу «Ввод PERT» в «Бюджетные риски». Затем на основе фильтра «Вехи» создадим фильтр «Не вехи», изменив условие в исходном фильтре на противоположное. После его применения на плане не будут отображаться задачи с нулевой длительностью.

При анализе PERT программа автоматически помещает значения оптимистической, ожидаемой и пессимистической длительности в поля «Длительность1-3». Если разделить длительность каждого из типов на длительность, внесенную в план проекта (поле «Длительность»), то в результате мы получим коэффициент, который можно использовать для расчета стоимости. Например, если длительность задачи в плане составляет 2 дня, а пессимистическая длительность составляет 4 дня, то коэффициент будет равняться 2. Соответственно, пессимистическая стоимость задачи будет равняться стоимости, умноженной на этот коэффициент, и в случае неблагоприятного развития событий будет в два раза больше запланированной. Настроим три поля типа «Затраты» для расчета стоимости каждого из типов по этой формуле «[Длительность1(2,3)]/[Длительность]*[Затраты]»Рис.. Если затраты на проект не были введены вами ранее – необходимо сделать это сейчас.

Видно, что в случае неблагоприятного развития событий стоимость проекта может увеличиться более чем на $15 000 (вычитаем из пессимистической стоимости планируемую стоимость), что составляет лишь 15,5% от общей стоимости проекта. Но у отдельных задач или фаз отклонение цены может быть значительным, и нужно проанализировать план, чтобы понять, у каких задач в случае осуществления риска стоимость может существенно измениться. Для этого рассчитаем для каждой задачи процент отклонения пессимистической стоимости от запланированной. Переименуем поле «Число3» в «Разница стоимости» и введем в него формулу «(([Затраты5]-[Затраты])/[Затраты]*100)». Сначала определяется разница между пессимистической ценой и запланированной, для чего из поля «Затраты5», где хранится пессимистическая стоимость, рассчитанная в предыдущем примере, вычитается планируемая стоимость, хранящаяся в поле «Затраты». Затем мы определяем, какой процент от запланированной стоимости составляет полученная разность. Для этого полученное в результате вычитания число делится на запланированную стоимость и результат умножается на 100. Чтобы полученный результат было легче обрабатывать, настроим отображение индикаторов для поля. Те задачи, у которых отклонение при неблагоприятном развитии событий составит более 50%, пометим красным индикатором. Задачи с отклонением больше 25% пометим желтым, а с отклонением больше или равным 10% - зеленым. Задачи с отклонением менее 10% пометим флажком. Установим флажок «Показывать значения данных во всплывающих подсказках», и тогда значение поля будет отображаться при наведении курсора на индикатор. Определение количественных характеристик отклонений (чтобы решить, какое отклонение считать слишком высоким, а какое приемлемым) зависит от принятых в организации стандартов. В нашем случае будем считать отклонение менее 10% приемлемым, а более 50% — слишком высоким и нуждающимся в коррекции. Задачи плана, помеченные красным индикатором, нуждаются в коррекции: нужно или уменьшить пессимистическую оценку стоимости для них, или увеличить планируемую стоимость. Определение количественных характеристик отклонений (чтобы решить, какое отклонение считать слишком высоким, а какое приемлемым) зависит от принятых в организации стандартов. В нашем случае будем считать отклонение менее 10% приемлемым, а более 50% — слишком высоким и нуждающимся в коррекции. Задачи плана, помеченные красным индикатором, нуждаются в

коррекции: нужно или уменьшить пессимистическую оценку стоимости для них, или увеличить планируемую стоимость. После завершения коррекции нужно определить пессимистическую стоимость проекта, согласовать ее с руководством и учитывать при планировании финансирования проекта. Если события будут развиваться по неблагоприятному сценарию, организация должна быть готова к выплате необходимого проекту бюджета. Пессимистическая стоимость проекта указана в строке суммарной задачи проекта.

Разработка стратегии смягчения рисков.

После того как мы выявили проектные риски, нужно определить меры, смягчающие их влияние на проект. Это можно сделать двумя путями: разработать план их сдерживания или план реакции на них. План сдерживания рисков состоит из работ, которые включаются в план проекта и, будучи выполненными, существенно снижают вероятность осуществления риска. План реакции на риски определяется в плане проекта, но не оформляется в виде задач до осуществления риска. Если риск осуществляется, нужные задачи добавляются в план проекта. Определяя стратегию смягчения рисков, следует всегда сравнивать затраты на предотвращение риска с затратами, которые будут понесены, если риск осуществится. Например, если в случае осуществления риска бюджет возрастет на $100, то стоимость работ по сдерживанию не должна превышать этой цифры. Когда важнее сроки проекта, следует сравнивать длительность плана в случае осуществления риска с длительностью плана, учитывающей задачи на его смягчение.

План реакции на риски.

Многие риски часто имеют очень низкую или неизвестную вероятность осуществления. Кроме того, для некоторых рисков нельзя определить момент их наступления. Например, есть риск, связанный с использованием Терехова, поскольку тот обладает уникальными знаниями, и все четыре задачи, где он задействован, не могут быть выполнены без его участия. Но точно определить момент наступления риска нельзя, поскольку он не связан с календарем проекта. В подобных случаях нужно разработать план реакции на риск, который будет применен в тот момент, когда риск осуществится.

План реакции на риски хранится в плане проекта в виде текстовой информации, связанной с определенными задачами или ресурсами. Для хранения информации о реакции на ресурсные риски настроим ресурсное поле «Текст4», переименовав его в «План реакции на риски». Даже после того, как план проекта проанализирован, многие риски выявлены и разработана стратегия смягчения их влияния на проект, все равно сохраняется вероятность, что в ходе выполнения проекта может произойти нечто непредвиденное. Иными словами, вполне возможно, что какие-то риски не были выявлены либо их существование нельзя предположить на нынешнем этапе планирования проекта. Поэтому в план нужно заложить временной и финансовый буфер, позволяющий отреагировать на возникающие риски и снизить вероятность увеличения длительности проекта. Финансовый буфер можно создать простым увеличением стоимости проекта на коэффициент, который принято использовать в вашей организации в таких случаях. Например, если бюджет проекта составляет $100 000, а пессимистический бюджет — $120 000, то с учетом буфера бюджет проекта может равняться $130 000.

Формирование временного буфера.

В хороший план проекта должна быть заложена определенная степень устойчивости к возникающим рискам. Так как риски приводят к задержкам в исполнении работ, то

устойчивость к рискам подразумевает в первую очередь возможность начать исполнение некоторых задач позже даты, указанной в плане, и при этом закончить проект в срок. Если у задачи можно перенести дату начала на более поздний срок или увеличить длительность, значит, она не является критической. Поэтому чем меньше в плане проекта критических задач, тем больше он подготовлен к возникающим рискам. Если план состоит только из критических задач, то он вряд ли будет выполнен в срок, поскольку в таком плане любая задержка приводит к смещению даты окончания проекта. В зависимости от стандартов планирования, принятых в организации, в плане проекта должен быть определенный процент некритических задач. Для анализа существующего в плане временного резерва удобно воспользоваться представлением «Диаграмма Ганта» и таблицей «Календарный план», в которой отображается информация о существующем временном запасе. Для того чтобы эта же информация отображалась и на диаграмме, настроим ее с помощью мастера «Мастер диаграмм Ганта». На первом шаге мастера (определение типа информации для отображения на диаграмме) выберем переключатель «Настроить диаграмму Ганта». На следующем шаге выберем переключатель «Да» для отображения информации о критических и обычных задачах разными способами. После этого пропустим все диалоговые окна с настройками цветов отрезков и дойдем до пятого, в котором определяются типы дополнительных отрезков, отображаемых па диаграмме. В этом диалоговом окне выберем переключатель «Общий временной резерв». Данные о существующем у задач резерве будут отображаться в виде тонких отрезков. На образце в области предварительного просмотра видно, что временной резерв может быть только у обычных задач (они более темные), поскольку у критических его не бывает. Теперь самые важные настройки завершены и можно нажать кнопку «Готово» прямо в этом диалоговом окне. Представление настроено, и можно начать работу с временным буфером. Таблица «Календарный план» содержит несколько колонок, с помощью которых можно определить степень устойчивости к рискам как расписания проекта в целом, так и его отдельных задач. В колонке «Общий временной резерв» содержится информация о времени, на которое исполнение задачи можно отложить, чтобы длительность проекта не изменилась. Колонка «Свободный временной резерв» содержит информацию о времени, на которое можно отложить исполнение задачи, чтобы не задерживать последующие задачи. А в колонках «Позднее начало» и «Позднее окончание» содержатся самые поздние даты, когда можно начать и окончить задачу, чтобы не изменить дату окончания проекта. Поле свободного временного резерва или общего резерва обычно содержит значение от нуля и больше. Если общий временной резерв задачи равен нулю,то она является критической (Если не изменены стандартные настройки). Однако при расчете временного резерва учитываются крайние сроки задачи и ограничения, поэтому если окончание задачи запланировано позже крайнего срока, то ее временной резерв будет отрицательным. Это значит, что ее не только нельзя отложить, а наоборот, надо ускорить. Если хотя бы у одной задачи проекта временной резерв меньше нуля, то временной резерв всего проекта (суммарной задачи проекта) также будет меньше нуля. На диаграмме информация об общем временном резерве задачи отображается с помощью тонких отрезков. MS Project рассчитывает общий и свободный временной резерв задачи, исходя из ее ограничений и положения в плане проекта. В нашем примере, исходя из положения задачи Проверка состояния статей в плане проекта, временной резерв составил больше 30 дней, хотя на самом деле эта задача должна быть выполнена за несколько дней до начала задачи. После того как вы просмотрите файл проекта и убедитесь, что временной резерв у каждой задачи соответствует действительности, нужно попытаться найти в проекте несбалансированности. Например, может

оказаться, что у одной из фаз слишком большой резерв, а у другой его нет или он вовсе отрицательный. В таком случае стоит перенести часть задач из фаз с маленьким резервом в те, где он значительно больше. В плане не должно быть задач или фаз с отрицательным резервом, потому что наличие таких задач свидетельствует об ошибках в плане проекта. Отрицательный временной резерв может образоваться, если задача заканчивается после крайнего срока или если нарушены даты ограничений у соседних с ней задач. Чтобы быстро найти задачи с отрицательным резервом, можно отсортировать таблицу по убыванию по полю «Общий временной резерв». Если задачи с ограничениями имеют предшественниц, заканчивающихся слишком поздно для того, чтобы ограничение было удовлетворено, у последующих задач образуется отрицательный резерв. Чтобы задачи с ограничением и с отрицательным резервом помещались в расписании в соответствии со связями, а не с датами ограничений, в диалоговом окне «Параметры» на вкладке «Планирование» нужно сбросить флажок «Для задач всегда соблюдаются заданные для них даты». Добавить резерв на задачи критического пути можно, увеличив их длительность или вставив задачи-буферы. Тогда при выполнении проекта длительность буферов нужно будет уменьшать, и после завершения проекта их длительность будет равна нулю. Если резерв задач можно организовать с помощью таблицы, то временной резерв проекта можно определить с дополнительных индикаторов. Например, можно запланировать закончить проект раньше реально нужного срока. Или же, как мы сделали, добавить крайний срок на последнюю задачу плана. В таком случае время между окончанием задачи и ее крайним сроком и будет временным резервом проекта.

Анализ распределения трудозатрат.

Когда план проекта готов и в него заложены буферы и временной резерв, следует проанализировать распределение трудозатрат в проекте. Эта информация часто оказывается полезной: например, можно заметить, что в определенные периоды в проекте наступает перерыв, который можно заполнить работами. Кроме того, руководитель проекта сможет оценить, в какие периоды его ожидает более интенсивная работа, а в какие нагрузка будет спадать. Анализ распределения трудозатрат выполняется в MS Project специальным мастером, вызываемым с помощью кнопки «Анализ повременных данных в Excel», расположенной на панели инструментов «Анализ». После щелчка на кнопке «Анализ повременных данных в Excel» появляется окно мастера анализа данных в Excel. На первом шаге нужно выбрать, анализировать ли весь проект или только выбранные задачи (если мастер запускается, когда открыто представление для просмотра ресурсов, то нужно выбрать, все ресурсы анализируются или только выбранные). На втором шаге выбираются поля «внутренней» таблицы, которые будут проанализированы. Чтобы выбрать поле для анализа, нужно выделить его курсором в списке полей (слева) и нажать кнопку «Добавить». Удаление поля из списка анализируемых осуществляется с помощью кнопки «Удалить». В нашем примере для анализа выбрано поле «Трудозатраты». Выбрав поля для анализа, нужно определить временной диапазон, в рамках которого будет осуществляться анализ. Этот диапазон определяется на третьем шаге мастера, и по умолчанию поля заполнены датами начала и окончания проекта. Под полями для выбора границ временного диапазона в раскрывающемся списке «Единицы» выбираются единицы измерения времени, используемые при анализе. Можно выбрать любую из единиц измерения шкалы времени MS Project.

На следующем шаге мастера нужно определить, будет ли в Excel строиться график по выбранным данным, и на последнем шаге — нажать кнопку «Экспорт данных», чтобы процесс

импорта начался. После этого в Excel будет создан файл, в котором на одном листе будут помещены данные, аналогично тому, как они представлены на диаграмме использования задач или ресурсов, а на втором листе будет создан график по этим данным.

Чем меньше выбранные единицы измерения, тем более неровным будет график распределения трудозатрат во время исполнения проекта. Иногда это полезно, а иногда не нужно. Например, для анализа распределения трудозатрат пи длительности всего проекта в качестве единицы измерения стоит выбрать педелю или месяц, и тогда график примет необходимую «обтекаемость» (понедельные и помесячные графики распределения работ по проекту хранятся в файлах wrk2.xls и wrk3.xls). А для сравнения загрузки ресурсов наиболее подходящим может оказаться именно ежедневный график. Выберете три трудовых ресурса и постройте по ним график.

Анализ загрузки ресурсов в Excel помогает определить, насколько равномерно она распределена. Такой анализ можно провести и в MS Project в представлении «График ресурсов», но в нем не так удобно сравнивать загрузку, просматривая ее для нескольких ресурсов сразу, потому что у графика нет объемного вида. В нашем примере видно, что почти весь февраль ресурсы простаивают, и их можно занять дополнительной работой в этом или другом проекте.

Чтобы получить график, отображенный на рисунке, мы изменили файл, полученный в результате автоматического экспорта данных в Excel. При автоматическом экспорте па графике отображаются суммарные данные о проценте загрузки для всех выбранных ресурсов. Мы удалили ее с графика и добавили на него отдельные ряды с данными каждого из ресурсов.

Задание на лабораторную работу. Анализ рисков по методу PERT.

1. Ввести в модель проекта данные об оптимистической, ожидаемой и пессимистической длительности проектных работ.

2. Выполнить анализ рисков по методу PERT для своего магистерского проекта.

3. Разработать предложения по повышению устойчивости плана к рискам в предположении, что дефицит ресурсов можно возместить их привлечением.

4. Оформить отчет.

Для выполнения лабораторной работы необходимо изучить следующие возможности и средства программы Microsoft Office Project:

· ввод данных об оптимистической, ожидаемой и пессимистической продолжительности работ;

· расчет средневзвешенной продолжительности работ с учетом сценарных продолжительностей;

· сопоставление планов, соответствующих реализации каждого сценария.

Лабораторная работа №4. Оценка и контроль проекта.

Цель лабораторной работы:

Научиться оценивать состояние проекта и контролировать его выполнение

Задание на лабораторную работу. Оценка и контроль проекта.

1. Оценка стоимости и затрат

Произвести расчет стоимости проекта на основе функционально-ориентированных метрик

Произвести расчет стоимости проекта на основе COCOMO II

Произвести расчет стоимости проекта на основе сетевого графика:

Определить стоимость часа/дня каждого вида специалистов.

Определить стоимость часа/дня по каждому виду работ.

Определить стоимость всего проекта и график финансирования по месяцам.

Произвести расчет стоимости и сроков проекта с учетом рисков.

Внести изменения в предыдущие расчеты.

2. Контроль состояния проекта

Рассчитать по состоянию на текущую дату по своему проекту:

- Показатели PV, EV, AC.
- CV, SV, CPI, SPI.
- Произвести оценку эффективности проекта
- BAC, ETC, EAC, VAC

3. Оформить отчет.

Практические занятия

Практическое занятие 1. Планирование состава работ.

План.

1. Диаграмма Ганта.
2. Шкала времени. Рабочее время.
3. Задачи. Фазы. Вехи.
4. Длительности задач.
5. Связи. Запаздывания и опережения.
6. Ограничения. Крайние сроки.
7. Повторяющиеся задачи. Суммарная задача. Задачи типа «гамак».
8. Самостоятельная работа.

Конспект.

Диаграмма Ганта (Gannt chart), ленточная диаграмма или график Ганта названа по имени своего разработчика Генри Л. Ганта (Henry L. Gantt). Гант изучал менеджмент на примере постройки кораблей во время первой мировой войны.

Диаграмма Ганта – популярное средство визуализации плана проекта. Представляет собой график, где на горизонтальной оси располагается шкала времени, а на вертикальной список задач. График представляет собой множество горизонтальных отрезков, соответствующих выполняемым задачам. Длина отрезка пропорциональна длительности задачи.

Шкала времени соответствует выбранному масштабу (Вид|Масштаб, Формат|Шкала времени) и рабочему времени календаря проекта. Настройка рабочего времени (Сервис|Изменить рабочее время) позволяет задать рабочие и нерабочие дни недели, продолжительность рабочего дня (Рабочие недели|[по умолчанию]|Подробности), а также исключения из общих правил (Исключения|[имя исключения]|Подробности): сокращенные рабочие дни, праздничные, переносы рабочих дней.

Список задач определяет состав работ проекта. Для удобства восприятия список может быть структурирован (дерево задач). Фаза (этап) – задача, содержащая вложенные задачи (не листовой узел). Обычно определение состава работ начинают с наиболее крупных фаз (скелетный план работ), затем определяют их состав и т.д. (проектирование сверху вниз). Управление положением задачи в ярусах дерева (Проект|Структура |на уровень выше, на уровень ниже). На практике используют не более 3-6 уровней вложенности (во избежание «микроменеджмента»). Правило «1.5–2%» – минимальная длительность задачи должна составлять 1.5–2% от длительности всего проекта. Правило «80 часов» – задачи длительнее двух недель рекомендуется разбить на подзадачи.

Веха (завершающая задача) – контрольная точка проекта, где достигаются ключевые промежуточные результаты проекта. Планируется как задача нулевой длительности.

Длительности обычных задач задаются вручную (Проект|Сведения о задаче|Длительность), например, 4ч (в часах), 3,5д (в днях) в соответствии с настройками условных обозначений (Сервис|Параметры|Правка). Специальный символ «а» обозначает астрономические (в отличие от рабочих) единицы времени, например, 16ач – шестнадцать астрономических часов. Значение длительности с вопросительным знаком означает приблизительную оценку, которая требует уточнения. К определению длительности полезно привлекать исполнителей. Длительности фаз вычисляются автоматически на основе длительностей вложенных задач.

В начале 1990-х диаграмма Ганта была дополнена линиями связей между задачами. Связь между задачами определяет, каким образом время начала или завершения одной задачи влияет на время начала или завершения другой. Задачи, влияющие на другие, называются предшествующими. Задачи, зависящие от других, – последующими. У каждой задачи может произвольное количество предшествующих (Проект|Заметки задачи|Предшественники). В колонке «Тип» задается тип связи:

- ОН – Окончание-начало (FS – Finish-to-start). Последующая задача не может начаться, пока не завершилась предшествующая.
- НН – Начало-начало (SS – Start-to-start). Последующая задача не может начаться, пока не началась предшествующая.
- ОО – Окончание-окончание (FF – Finish-to-finish). Последующая задача не может завершиться, пока не завершена предшествующая (но могут выполняться одновременно).
- НО – Начало-окончание (SF – Start-to-finish). Последующая задача не может завершиться до начала предшествующей.

Связи также можно задать путем перетаскивания мышью предшествующей задачи на последующую. В соответствии с заданными связями происходит автоматическая корректировка плана работ при изменении сроков одной из задач.

В случае если начало или окончание последующей задачи не должно быть ранее начала или окончания предшествующей плюс некоторый интервал времени, то говорят о наличии запаздывания (столбец «Запаздывание»). Если задача может начаться или окончиться раньше на некоторое время, чем начало или окончание предшествующей задачи, то говорят о наличии опережения. Опережение задается как отрицательной величины запаздывание. Величина запаздывания или опережения может задаваться в единицах времени или в процентах от выполнения предшествующей задачи.

Ограничения определяют способ привязки задач к датам (Проект|Сведения о задаче|Дополнительно|Тип ограничения):

- КМР – Как можно раньше (ASAP – As soon as possible). Поле «Дата ограничения» не используется (значение «НД»). КМР – тип по умолчанию при планировании от даты начала проекта (Проект|Сведения о проекте|Планирование от).
- КМП – Как можно позже (ALAP – As late as possible). Поле «Дата ограничения» не используется. КМП – тип по умолчанию при планировании от даты окончания проекта.
- ОНП – Окончание не позднее (FNLT – Finish no later than) заданной даты (поле «Дата ограничения»).
- ННП – Начало не позднее (SNLT – Start no later than) заданной даты.
- ОНР – Окончание не ранее (FNET – Finish no earlier than) заданной даты.
- ННР – Начало не ранее (SNET – Start no earlier than) заданной даты.
- ФН – Фиксированное начало (MSO – Must start on) с заданной даты.
- ФО – Фиксированное окончание (MFO – Must finish on) на заданную дату.

По возможности рекомендуется использовать более гибкие ограничения (КМР, КМП), что дает больше возможностей при перерасчете плана в случае изменений. Также рекомендуется вводить ограничения до планирования ресурсов, чтобы назначить достаточное количество исполнителей для выполнения работы в срок.

Крайний срок – это предельная дата выполнения задачи. В отличие от ограничения не влияет на расчет плана, а лишь предупреждает (значком в поле «Индикаторы») о срыве срока.

Для создания регулярно выполняющихся задач удобно воспользоваться специальным инструментом (Вставка|Повторяющаяся задача). В диалоговом окне можно определить частоту, длительность возникающей задачи, пределы повторений, календарь. Повторяющаяся задача создается как фаза, где повторения – это вложенные задачи, параметры которых можно скорректировать, если они не укладываются в общее правило.

Для определения общей длительности проекта и прочих итоговых величин не требуется вводить корневую фазу, поскольку таковая создается автоматически – это суммарная задача проекта (Сервис| Параметры| Вид| Параметры структуры для проекта. Показывать| Суммарную задачу проекта).

Иногда возникает необходимость жестко связать параметр одной задачи с параметром другой задачи. Например, для задач типа «гамак», которые должны длиться на протяжении всей фазы (бухгалтерская поддержка, охрана объекта). В таких задачах дата начала должна совпадать с датой начала фазы, а время окончания – с датой окончания фазы. Осуществить это можно с помощью связки одних ячеек таблицы задач со значениями других ячеек (через механизмы OLE). Для этого нужно скопировать значение ячейки (Правка|Копировать ячейку) и вставить в зависимую ячейку (Правка|Специальная вставка). В диалоговом окне выбрать «Связать» как «Текстовые данные». В результате все изменения в исходной ячейке будут отображаться в зависимую ячейку. Конечно, с помощью данного механизма можно решать и более изощренные задачи.

Задание для самостоятельной работы.

Выбрать вариант учебного проекта. В рамках предполагаемого времени проведения проекта определить календарь рабочего времени, учитывая праздничные дни РФ, сокращенное рабочее время предпраздничных дней и переносы рабочих дней. Определить состав работ: фазы, задачи, вехи; приблизительные длительности задач, типы связей, ограничения. В ходе планирования следует учитывать, что дата начала проекта известна, но лишь приблизительно и впоследствии может быть изменена. Определить приблизительные сроки выполнения всего проекта.

Варианты

Практическое занятие 2. Планирование ресурсов.

План

1. Типы ресурсов. Бюджетные ресурсы.
2. Единицы измерения. Рабочее время. Доступность.
3. Назначения. Типы задач.
4. Профиль загрузки. Ограниченная длительность назначения.
5. Перерывы в выполнении работы.
6. Самостоятельная работа.

Конспект

Планирование начинается с определения состава ресурсов (Вид|Лист ресурсов). Каждый из ресурсов необходимо отнести к одному из трех типов: трудовой, материальный и затраты. Трудовой тип может относиться как к людям так оборудованию, которые назначаются задаче, но не расходуются в ходе ее выполнения. Для расходных ресурсов выбирают материальный тип. Затраты позволяют учитывать расходы, не зависящие от трудозатрат и длительности задачи. Типом определяется мера использования ресурса. Для трудовых – это время, для материальных – количественные величины (поле «единицы измерения материалов»). Независимо от типа ресурс может быть бюджетным или внебюджетным (управляется с помощью флага Проект|Сведения о ресурсе|Общие|Бюджет).

По умолчанию считается, что все сотрудники или оборудование работает в соответствии с календарем проекта. Если же отдельные ресурсы имеют собственный календарь, то он может быть определен в Проект|Сведения о ресурсе|Общие|Изменить рабочее время. В течение рабочего времени сотрудник или оборудование могут не быть доступными на 100%, например, в случае если они заняты в нескольких проектах. Доступность сотрудника задается (Проект|Сведения о ресурсе|Общие|Доступность ресурса) в процентах на множестве непересекающихся временных интервалов. При этом значение НД (NA) в поле «доступен с» имеет смысл минус бесконечности, а в поле «доступен до» – плюс бесконечности. Итоговая доступность ресурса определяется и рабочим временем и доступностью в процентах. Поле «Максимальная загрузка» в списке ресурсов соответствует первой строке в таблице доступности.

Назначение – это выбор ресурсов, необходимых для выполнения задачи. Для создания назначения нужно добавить строку в Проект|Сведения о задаче|Ресурсы. При этом указывается название назначаемого ресурса и единицы. В случае трудового ресурса «единицы» означают процент занятости. При превышении доступности в списке ресурсов появляется индикатор о необходимости выравнивания загрузки. Для материальных ресурсов вводится либо фиксированная величина объема выделяемых на задачу ресурсов, соответствующая единицам измерения данного ресурса, либо переменная (например, 3 кг/д – 3 кг в день).

Тип задачи и флаг фиксированного объема работ определяют, как изменение одного из свойств задачи – длительности, трудозатрат или объема работ, – повлияет на два других (Проект|Сведения о задаче|Дополнительно).

Тип задачи	Фиксированный объем работ	Изменение	
		длительности	ресурсов
фиксированная длительность	вкл	пересчет объема работ	пересчет загрузки ресурсов
	выкл	пересчет объема работ	пересчет объема работ

фиксированный объем ресурсов	вкл	пересчет объема работ	пересчет длительности
	выкл	пересчет объема работ	пересчет объема работ
фиксированные трудозатраты	вкл	пересчет загрузки ресурсов	пересчет длительности

Профиль загрузки определяет распределение трудозатрат во времени (Проект|Сведениея о назначении|Общие в представлении «Использование ресурсов»). По умолчанию они распределяются равномерно (плоский). Для длительных задач, возможно, будут более удобны другие варианты загрузки: загрузка в конце, загрузка вначале, двойной пик, поздний пик, колокол, черепаха (колокол с большей дисперсией). Так же в представлении «использование задач» можно вручную указать распределение трудозатрат по дням.

Если ресурс назначается не на все время выполнения задачи, то это можно отразить в полях «Начало» и «Окончание». При этом будет произведен пересчет трудозатрат для задач с фиксированной длительностью и объемом ресурсов и пересчет длительности при фиксированных трудозатратах.

В случае, если ресурс должен выбыть из исполнения задачи на некоторый срок, это можно сделать либо в представлении «использование задач», указав вручную профиль загрузки по дням, либо, если на время отсутствия ресурса, задача не может выполняться в принципе, то с помощью специальной команды (Правка| Прервать задачу).

Задание для самостоятельной работы.

Определить список необходимых для выполнения проекта трудовых и материальных ресурсов. Определить рабочее время для сотрудников, работающих по особому графику, учесть отбытие сотрудников в отпуска и командировки. Выполнить назначения ресурсов, определить типы задач.

Практическое занятие 3. Планирование стоимости.

План

1. Стоимость ресурсов, назначений, задач.
2. Начисление затрат. Порядок оплаты работ.
3. Планирование доходов, выплат заработной платы, штрафов.
4. Бюджет.
5. Самостоятельная работа.

Конспект.

Методики оценки проектов: по аналогии (если проект аналогичен ряду других), по параметрам (накопленный опыт оценки однотипных проектов формализуется в виде формулы, зависящей от ряда параметров проекта), сверху вниз (исходя из фиксированных затрат на фазу или проект в целом, при ограниченном бюджете), снизу вверх (определяются стоимости отдельных задач, которые суммируются). MS Project во многом ориентирован на методику оценки стоимости проекта снизу вверх.

Стоимость каждой задачи складывается из стоимости назначенных ресурсов и фиксированной стоимости задачи.

Стоимость ресурсов определяется из значений стандартной ставки (R), ставки сверхурочных (O) и затрат на использование (U) (Проект| Сведения о ресурсе| Затраты). Для трудовых ресурсов ставки задаются в формате «число/единица времени». Для материальных стандартная ставка задается в формате «число» (понимается как «число/единица ресурса»), а понятие сверхурочной ставки не определено. Если требуется учесть изменение ставок во времени, то нужно задать более одной строки ставок, где поле «дата действия» означает дату начала действия ставок в данной строке. Ресурс может иметь различные ставки (нормы затрат) при выполнении различного рода задач или с учетом особых условий выполнения задач (работа на выезде). В MS Project предусмотрено до 5 различных норм затрат для ресурса (вкладки A, B, C, D, E).

Стоимость назначения вычисляется автоматически: R*(кол-во потраченных рабочих часов)+O*(сверхурочные часы)+U*(использование ресурса) или для материальных: R*(кол-во единиц)+U. Значения R, U берутся из выбранной таблицы норм затрат (Проект| Сведения о назначении| Общие| Таблица норм затрат), соответствующие времени использования ресурса, если R, U зависят от времени. Сверхурочные часы задаются вручную в столбце «сверхурочные трудозатраты» в строке назначения.

Стоимость задачи складывается из стоимостей назначений и фиксированной стоимости задачи, не связанной с использованием проектных ресурсов (Вид| Таблица| Затраты| Фиксированные затраты).

Для планирования динамики освоения бюджета следует определить не только стоимость работ, но и порядок оплаты: предоплата, оплата по факту завершения или оплата по мере выполнения. Способ оплаты задается и для ресурсов (Проект| Сведения о ресурсе| Затраты| Начисление затрат), и для фиксированных затрат (Вид| Таблица| Затраты| Начисление фиксированных затрат). Необязательно порядок оплаты должен совпадать с договорным. Например, если по договору оплата производится по окончании работ, но неизвестно, когда они завершатся, то имеет смысл выбрать порядок оплаты в начале работ, чтобы средства были запланированы, и можно было расплатиться в любой момент.

Если труд сотрудников оплачивается не по сдельной схеме, а в форме оклада, то это можно учесть следующим образом. В отдельной таблице норм затрат для каждого сотрудника

зафиксировать зарплату в поле «затраты на использование». Далее на определенный день запланировать задачу «выдача зарплаты» и назначить на нее всех сотрудников, получающих зарплату.

Доходы в проектах в простейших случаях могут быть спланированы как веха с отрицательными фиксированными затратами. В случае если доходы являются штрафами за просроченное выполнение работ и величина их зависит от длительности задержки сдачи работ, то в плане это можно учесть с помощью дополнительно назначенного на задачу ресурса («штрафного ресурса»). Штрафной ресурс – это обычный трудовой ресурс с нулевой ставкой до договорной даты окончания работ и ненулевой ставкой после этой даты. Знак ставки определяет направление выплат штрафа.

В случае заранее определенного бюджета удобно внести в проект его размер для анализа соответствия плана и бюджету. Это особенно актуально в тех случаях, когда бюджет разделен по статьям и временным интервалам. Для отражения информации о бюджете нужно создать ресурс типа «затраты» (или несколько – по числу статей бюджета) и назначить на суммарную задачу проекта. После этого в представлении «использование задач» выберем отображение строк «бюджетная стоимость» и «затраты» (Формат| Подробности). Далее, выбрав удобный масштаб времени, вводим в ячейки параметры бюджета (можно вводить даже в те ячейки, которые за пределами дат начала и окончания проекта). Меняя масштаб времени, можно получить сводку о величине бюджета и расходах в разрезе месяцев, кварталов и пр.

Задание для самостоятельной работы.

Определить ставки трудовых ресурсов и порядок оплаты работ, определить стоимость материальных ресурсов, учесть штрафы за выполнение работ не в срок. Учесть оклады штатных сотрудников и премии по окончании работ. Определить итоговую стоимость проекта. Ввести информацию о бюджете, сравнить с оценочными данными.

Практическое занятие 4. Анализ и оптимизация.

План

1. Выравнивание загрузки ресурсов.
2. Анализ критического пути.
3. Анализ и оптимизация стоимости.
4. Самостоятельная работа.

Конспект.

Превышение доступности ресурса – это назначение работы ресурсу, для выполнения которой требуется больше времени, чем у него есть (например, назначены две задачи, выполняемые одновременно). Такие ресурсы будут выделены красным цветом, а также будут помечены специальным индикатором. Возможные способы устранения перегрузок: сократить задачи, назначить других исполнителей (Сервис| Назначить ресурсы), избавиться от пересечения задач, учесть работу сверх нормы как сверхурочную (столбец «сверхурочные трудозатраты» в строке назначения).

Диалоговое окно (Сервис| Выравнивание загрузки ресурсов) в MS Project иммет следующие настройки:

- Выполнять автоматически/вручную. В первом случае выравнивание происходит непосредственно при назначении ресурса. Во втором случае – при нажатии кнопки «Выровнять».
- Поиск превышений доступности (по дням, по часам, по минутам…). Шаг группировки при вычислении суммарной загрузки ресурса.
- Очистка данных предыдущего выравнивания перед новым выравниванием. MS Project отделяет исходный проект и полученный в результате выравнивания. Всегда можно отменить изменения (Сервис| Выравнивание загрузки ресурсов| Очистить выранивание).
- Диапазон выравнивания (во всем проекте или на заданном временном интервале).
- Порядок выравнивания. Только по идентификаторам – в первую очередь будут изменяться параметры задач, расположенные ниже в списке задач (с наибольшими идентификаторами). Стандартный – в первую очередь задачи с наибольшим временным резервом, более поздней датой, меньшим приоритетом (задачи с приоритетом 1000 при выравнивании не откладываются и не прерываются). По приоритетам, стандартный – задачи упорядочиваются по приоритету, затем анализ стандартным способом.
- Выравнивание только в пределах имеющегося резерва. Полагает, можно ли при выравнивании изменять дату окончания проекта.
- При выравнивании допускается коррекция отдельных назначений для задачи (в противном случае меняются только свойства задачи в целом).
- При выравнивании допускается прерывание оставшихся трудозатрат (разрешение прерывать задачи).
- Выравнивание загрузки предложенных ресурсов (разрешение использования как подтвержденных, так и предложенных ресурсов).

Для сравнения исходного плана и плана после выравнивания можно воспользоваться представлением «Leveling Gantt».

Анализ плана проекта методом критического пути позволяет найти те задачи, изменение длительности которых приведет к изменению сроков проекта. Критический путь – это множество задач, определяющие дату окончания проекта. MS Project может относить задачи к критическим не только, если ее временной резерв меньше, либо равен нулю, но и близок к нулю

(Сервис| Параметры| Расчеты| Считать критическими задачи, имеющие резерв не более … дней). Для отображения критического пути можно воспользоваться мастером диаграмм Ганта (Формат| Мастер диаграмм Ганта). На втором шаге нужно выбрать «Критический путь». После этого критический путь будет выделен красным. Изменение длительностей задач, может исключить их из критического пути, может сделать критическими другие задачи.
Помимо общей стоимости проекта («Общие затраты» для суммарной задачи) при анализе структуры затрат обычно рассматриваются:

- Распределение затрат по фазам проекта. Чтобы видеть процент затрат на фазу, удобно ввести дополнительное поле в таблицу (Вставка| Столбец), которое будет вычисляться по формуле (Сервис| Настройка| Поля| Формула).
- Распределение затрат по типам работ. Создается новое поле (Вставка| Столбец). Задается множество возможных значений (Сервис| Настройка| Поля| Подстановка). Для каждой задачи определяется ее тип. Создается новое поле затрат (с помощью формулы) с суммированием для суммарных строк задач и групп (Сервис| Настройка| Поля| Сведение). Далее задачи группируются по типу (Проект| Группировка| Другие группы| Создать| Группировать по).
- Соотношение между затратами на сверхурочные трудозатраты и обычные. Аналогично распределению затрат по фазам можно создать новое поле, вычисляющее по формуле отношение [Затраты на сверхурочные]*100/[Затраты]. Чтобы избавиться от делений на ноль, можно воспользоваться условным оператором: IIF([Затраты]<>0; [Затраты на сверхурочные]*100/[Затраты];0).
- Распределение затрат на ресурсы различных типов.

В основном, уменьшить стоимость проекта можно только привлекая дешевые ресурсы или частично или полностью отказавшись от некоторых работ за счет качества итогового продукта или сроков его получения. Оптимизируя сроки выполнения работ, можно также сократить выплаты неустоек.

Задание для самостоятельной работы.

Выровнять загрузку тех сотрудников, кто не согласился работать сверхурочно (здесь и далее определяется вариантом). Учесть в стоимости проекта сверхурочную занятость сотрудников. Изменить план проекта в связи с переносами сроков окончания проекта. Оптимизировать стоимость проекта в связи с уменьшением бюджета.