top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

УПРАВЛЕНИЕ ПРОГРАММНЫМИ ПРОЕКТАМИ

УРОК №2

ДЕТАЛЬНЕЙ ОБ УПРАВЛЕНИИ ПРОЕКТОМ

СОДЕРЖАНИЕ

1. ПРОЕКТ

Составляющие управления проектом

Управление проектом можно представить, как совокупность действий, которые выполняет руководитель проекта для того, чтобы сделать проект успешным.

Когда говорят об успешности проекта, обычно подразумевают, что успешный проект тот, что достиг целей, уложился в первично утвержденные сроки и бюджет.

В исследовании PMI (американского Института управления проектами) 2018 года приводится следующая статистика успешности проектов:

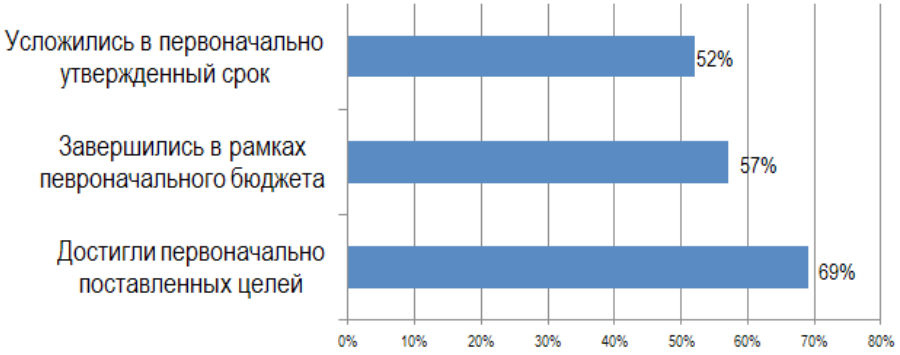


Рисунок 1

Для того чтобы реализовать успешный проект руководителю проекту нужно знать и уметь достаточно много, но в первую очередь, ему нужно понимать, как работает «проектный треугольник».

Параметры проекта

Для того чтобы описать параметры проекта, в управлении проектами используется термин **«проектный треугольник»**, известный также как «треугольник ограничений проекта» или «железный треугольник проекта».

Любой проект должен быть ограничен следующими параметрами:

Рисунок 2

1. **Содержание проекта (функциональность)**

Содержание ИТ-проекта обычно формируется следующим образом:

Цель проекта > Результаты проекта > Требования к результатам проекта.

Так как с формулировкой целей у большинства проектов есть сложности (*ниже будет приведена диаграмма,*

описывающая факторы неудач проектов), то для формулировки цели проекта рекомендуется использовать критерии **SMART**. Цель должна быть:

- **S (Specific)** – конкретной. Иногда расшифровывают S, как «significant» (*существенной*).

Конкретность подразумевает использование такой формулировки, которая позволяет понять, каким должен быть результат (результаты) при достижении цели. При этом в формулировке цели нужно избегать «процессного» описания. Например, цель: «Непрерывно улучшать процесс тестирования» звучит, как процесс, а цель «Улучшить процесс тестирования» звучит, как описание результата.

- **M (Measurable)** – измеримой, должны быть указаны количественные параметры для измерения степени достижения цели.

В описание цели добавляем показатель, с помощью которого можно измерить степень ее достижения и получаем: «Снизить показатель "Доля повторно открытых багов" на X процентов».

- **A (Achievable (Ac), Ambitious (Am)** – с одной стороны, достижимой, и с другой стороны, амбициозной.

Необходимо проверить, хватает ли ресурсов для достижения цели и является ли цель настолько вдохновляющей, чтобы ее достижение стало вызовом.

- **R (Relevant)** – соотносимой с другим целями.

Обычно для проверки на релевантность необходимо в стратегическом документе компании найти стратегическую цель, на достижение которой влияет данная цель.

- **T (Time bound, Trackable)** – определенной во времени и отслеживаемой.

Добавляем в формулировку цели ограничение по времени и получаем: «Снизить показатель "Доля повторно открытых багов" на X процентов за первые полгода с момента внедрения автоматизированных тестов».

Считается, что впервые термин **SMART** ввел в обиход известный гуру менеджмента **Питер Друкер** в 1954 году в своей книге «**The Practice of Management**». Не известно, подбирал ли Друкер аббревиатуру так, чтобы она соответствовала английскому слову «smart» (в переводе – *умный*) или это получилось случайно, но на сегодняшний день в научной литературе по менеджменту – это единственный общепризнанный подход к постановке целей (не считая подходов из НЛП).

Так как после формулировки цели руководитель проекта и заказчик обычно формулируют результаты проекта, а затем рассчитывают срок и бюджет проекта, то часть критериев SMART может быть добавлена только после расчета плановых сроков и бюджета проекта.

Например, цель ИТ-проекта может быть сформулирована так: «Разработать программный продукт, с помощью которого компания сможет увеличить среднюю прибыльность сделок с клиентами на 5% в течение первого года с момента ввода в промышленную эксплуатацию».

В такой формулировке цели соблюдаются два из пяти критериев: **S** (Specific) и **M** (Measurable).

Для проверки того, является ли цель амбициозной и при этом достижимой, руководителю проекта надо создать модель расписания проекта и проверить каким количеством ресурсов и за какой срок можно реализовать проект. После прогноза сроков и бюджета проекта цель будет соответствовать критериям **A** (Achievable (Ac), Ambitious (Am) и критерию **T** (Time bound).

Для проверки критерия **R** (Relevant) руководителю проекта нужно убедиться, что данная цель является одной из стратегических целей компании и присутствует в стратегическом документе компании, либо позволяет достичь одну из стратегических целей.

Результатами данного ИТ-проекта станут:

- работающий программный продукт;
- документация на продукт (для разработчиков);
- материалы обучающих программ для пользователей;
- обученные работе в программном продукте и по новым процессам сотрудники компании;
- данные о средней прибыльности сделок с клиентами за период, равный одному году с момента ввода программного продукта в промышленную эксплуатацию.

Требования к программному продукту будут описаны в виде списка требуемой функциональности от программного продукта и оформлены в документе Software Requirements Specification.

Для проекта разработки программного продукта содержание будет ограничено требованиями, описанными в Software Requirements Specification, которые в конечном итоге трансформируются в WBS-проект и в список задач

проекта. Как только все требования будут реализованы, проект можно завершать и результаты сдавать заказчику проекта.

2. **Бюджет проекта (стоимость проекта)** – денежная сумма, которую заказчик проекта готов потратить на достижение целей проекта и получение ожидаемых результатов.

Плановый бюджет ИТ-проекта обычно формируется из нескольких статей затрат. Статьи затрат ИТ-проекта могут быть следующими:

- затраты на оплату труда команды проекта;
- налоги с заработной платы;
- закупка «железа» (компьютеров, серверов, планшетов и т. д.);
- приобретение лицензионного программного обеспечения (СУБД, операционные системы и т. д.);
- резерв на риски проекта;
- плановая прибыль проекта.

Самой большой статьей затрат для команды исполнителя ИТ-проекта, как правило, является статья «Затраты на оплату труда».

3. **Срок реализации проекта** – длительность проекта во времени, которая рассчитывается как количество дней, начиная с момента официального старта проекта и до даты официального финиша проекта.
4. **Качество** – степень выполнения требований к проекту. Изменение любой из сторон треугольника приводит к изменению уровня качества.

Как работает «проектный треугольник»?

- Чтобы сократить сроки проекта можно привлечь больше ресурсов (это увеличит стоимость проекта) или отказаться от части требований (уменьшение содержания проекта).
- В случае отклонения от плановых сроков проекта можно было бы предложить исполнителям работать сверхурочно по более высоким ставкам, однако это приведет к увеличению бюджета проекта.
- Чтобы сделать продукт проекта более привлекательным для потребителей можно добавить в него ранее не запланированные функции, однако это приведет к необходимости изменить крайний срок проекта или привлечь дополнительных сотрудников, чтобы остаться в рамках сроков (но это приведет к увеличению бюджета).

Надо понимать, что универсального определения термина «качество» для проектного треугольника не существует. Для каждого проекта «качество» определяется заказчиком проекта. Для одного заказчика «качество» означает «уложиться в бюджет», а для другого важнее в плановый срок вывести новый продукт на рынок. Руководителю проекта важно на старте проекта узнать у заказчика проекта, как он трактует термин «качество» и что для него важнее: сделать проект в срок, уложиться в бюджет или реализовать все запланированные продукты (результаты) и их функциональность.

Заказчику проекта следует выбрать одну из стратегий, изображенных на рисунке 3.

Рисунок 3

На месте заказчика большинство из нас хочет сделать проект *Быстро-Дешево-Качественно*, однако, доля таких проектов крайне мала, потому эту стратегию называют утопичной.

Для определения ограничений по проекту предлагаем следующую последовательность действий:

1. Определить вместе с заказчиком цель проекта. Проверить формулировку цели на соответствие критериям SMART.
2. Обсудить с заказчиком и зафиксировать планируемые результаты проекта.
3. Извлечь (выполнить сбор) и проанализировать требования к результатам проекта.

4. Исходя из требований к результатам проекта, осуществить проектирование WBS-проекта.
5. Спрогнозировать объемы работ по всем задачам из WBS и по проекту в целом.
6. Получив прогнозы по объемам работ задач проекта, спрогнозировать срок реализации проекта (например, используя метод критического пути).
7. Рассчитать бюджет на проект.
8. Оформить договоренности о содержании, сроках и бюджете проекта в договоре или Уставе проекта.

Факторы, влияющие на стоимость проекта

Стоимость ИТ-проекта во многом зависит от объема работ, который необходимо проделать, чтобы реализовать запланированное содержание проекта, и ставок человеко-часа участников проекта.

Например, если объем работ по проекту оценивается командой проекта в 1000 чел.-часов, из которых 350 часов – это работа программистов, 250 часов – работа тестировщиков, 200 часов – задачи бизнес-аналитиков, 100 часов – работа руководителя проекта и 100 часов – работа технического писателя, то, зная ставки сотрудников, можно рассчитать затраты на их заработную плату в проекте. Статья «Затраты на заработную плату команды проекта» является самой большой статьей затрат в бюджетах большинства ИТ-проектов.

Принципы оценки стоимости проекта

Для оценки стоимости проекта руководителю ИТ-проекта нужно оценить объемы работ по проекту.

В разработке программного обеспечения есть много подходов для оценки объемов работ по проекту, условно их можно разделить на две группы:

1. **Традиционные подходы к оценке.**
2. **Подходы к оценке, используемые в Agile.**

К **традиционным подходам** относятся:

- модель COCOMO 2;
- Functional Point Analysis (FPA);
- метод оценки по трем точкам (PERT).

Модель COCOMO 2

Оценка трудоемкости проекта основана на анализе результатов исследования 63-х проектов разработки программного обеспечения, который выполнил **Барри Боэм** (*Barry Boehm*) и опубликовал в 1981 году.

Через 10 лет появилась новая версия **COCOMO II**, учитывающая появившиеся с того времени новые жизненные циклы проектов разработки ПО и использующая больший набор данных.

Объем работ по проекту при использовании COCOMO II зависит от такого показателя как количество оценочных тысяч строк кода (*KLOC – kilo lines of code*) и от поправочных коэффициентов, выведенных Б. Боэмом на основании анализа 63-х реализованных проектов.

Есть три уровня оценки, используемых в COCOMO II:

1. **Базовый уровень оценки** – не учитывает влияние на объем проекта ряда факторов, которые сложно учесть на ранних стадиях разработки, например, опыт команды в разработке ПО.
2. **Средний уровень оценки** – размер проекта зависит от размера программного продукта, оценки характеристик продукта, характеристик проекта, характеристик участников проекта и характеристик аппаратного обеспечения.
3. **Детальный уровень оценки** – учитывает влияние отдельных фаз проекта на его трудоемкость и, соответственно, стоимость.

Functional Point Analysis (FPA)

Для использования FPA руководителю проекта необходимо иметь функциональные требования к программному продукту.

Все функциональные требования подразделяются на пять категорий: выходы, запросы, входы, внутренние файлы и внешние интерфейсы. Каждой функции присваивается оценочное число функциональных баллов в зависимости от ее сложности. Оценка трудоемкости функции рассчитывается по следующей формуле:

$$FP = UAF \times VAF$$

где:

FP – функциональная точка;

UAF – нескорректированная функциональная точка;
VAF – поправочный коэффициент.

Оценка всего проекта получается суммированием оценок всех функциональных точек.

Метод трех точек (PERT – Program/Project Evaluation and Review Technique)

Для оценки трудоемкости одной задачи проекта используется три оценки (см. рис. 4):

O – оптимистическая оценка трудоемкости задачи (если все пойдет хорошо);

P – пессимистическая оценка трудоемкости задачи (если все пойдет плохо и случатся все возможные проблемы);

M – наиболее вероятная оценка.

Формула, которая используется для расчета PERT-оценки:

$$E = (O + 4*M + P) / 6$$

Оптимистическую оценка трудоемкости можно получить у исполнителя задачи, которого руководитель проекта считает оптимистом: исполнитель чаще всего не укладывается в плановое время выполнения задач.

Пессимистическую оценку делает потенциальный исполнитель задачи, который чаще всего дает прогнозную оценку выше, чем его коллеги

И наиболее вероятную оценку трудозатрат можно получить, изучив статистику по фактическим трудозатратам для аналогичных задач, в ранее выполненных проектах.

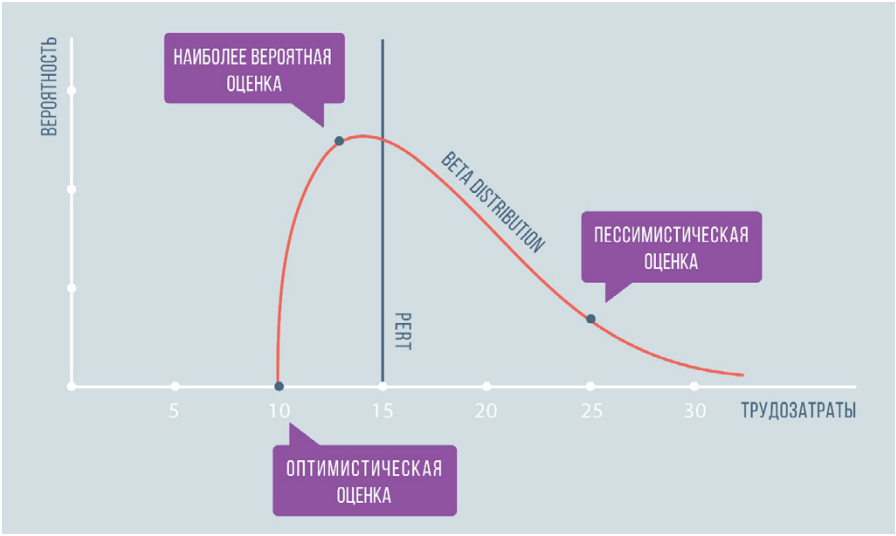


Рисунок 4

Рассчитав оценки PERT для всех задач проекта, руководитель проекта может получить прогнозную оценку трудоемкости всего проекта.

Для этого нужно просуммировать полученные по формуле PERT оценки трудоемкостей по всем задачам проекта и добавить резерв на непредвиденные обстоятельства. Этот резерв можно спрогнозировать с помощью расчета **среднеквадратического отклонения** по проекту: подсчитать среднеквадратическое отклонение (СКО) по каждой задаче проекта, полученное значение возвести в квадрат, после чего извлечь корень квадратный из суммы квадратов СКО по всем задачам.

В случае если руководитель проекта хочет получить прогноз трудоемкости проекта с точностью до 95%, нужно использовать следующие формулы (где *i* – номер задачи) (см. рис. 5).

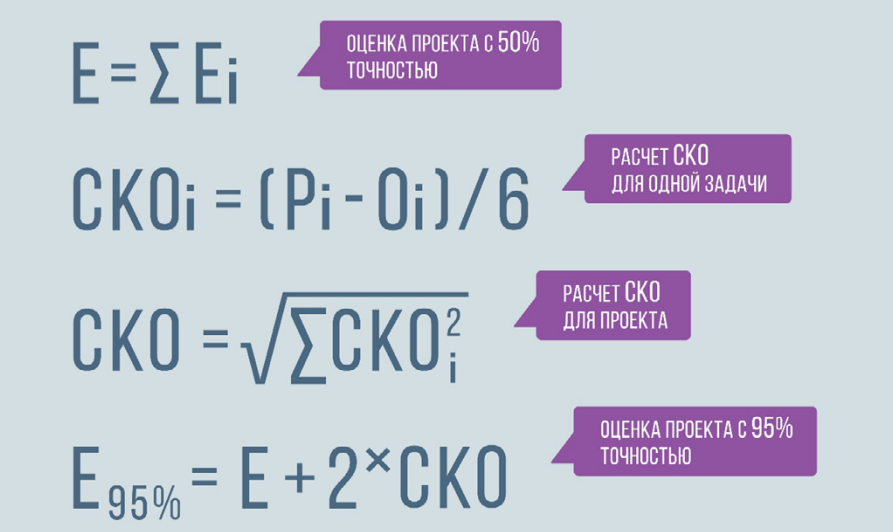


Рисунок 5

Подходы к оценке объемов работ, используемые в Agile

Покерное планирование

Покерное планирование представляет из себя модификацию метода Дельфи. В методе Дельфи оценка основана на мнении группы экспертов, при этом очень важно избавиться от так называемого эффекта «привязки». **Д. Канеман** так описывает эффект «привязки»: «*Эффект привязки проявляется, когда перед оценкой неизвестного значения испытуемые сталкиваются с произвольным числом. Этот эксперимент дает одни из самых надежных и стабильных результатов в экспериментальной психологии: оценки не отдаляются от рассмотренного числа, отсюда и образ привязки к определенной точке*».

Для того чтобы получить групповую независимую оценку, никто из экспертов не должен называть свою оценку, пока все не сделают выбор, и только после этого оглашаются результаты.

Как проходит покерное планирование?

Участники оценки (потенциальные исполнителя задач) и модератор совещания собираются в одном помещении. У каждого участника, выполняющего оценку трудозатрат задач проекта, есть колода карт, например, такая:

Рисунок 6

Значения на картах данной колоды соответствуют ряду Фибоначчи, в котором для определения следующего значения ряда используется правило: сумма двух предыдущих чисел должна давать следующее число. Есть несколько карт, которые используются скорее как «раз-

влекательные»: например, карта со знаком «?», эксперт вытягивает ее в случае, если до конца не понимает требования к результатам задачи, или карта с изображением «чашки», используемая в том случае, если эксперт устал от оценки и жаждет перерыва.

Есть несколько карт, цифры на которых не соответствуют ряду Фибоначчи: 20, 40 и 100. Если участник оценки вытягивает такие карты, то тем самым он демонстрирует свое мнение о том, что задача слишком масштабная. В этом случае, модератору митинга рекомендуется декомпозировать задачу на более мелкие подзадачи и оценивать их.

Процедура использования карт с оценками следующая. Модератор зачитывает пожелание (или формулировку задачи), оценщики задают вопросы относительно требований к пожеланию, и заказчик проекта отвечает на их вопросы.

После того, как все ответы получены, модератор просит каждого участника сделать оценку трудозатрат. Продумав оценку, эксперт должен вытянуть карту с цифрой, соответствующей этой оценке и положить ее рубашкой вверх, чтобы никто не видел его оценку. То есть, если эксперт оценивает реализацию задачи в 8 часов, он вытянет карту с цифрой 8 и положит ее рубашкой вверх.

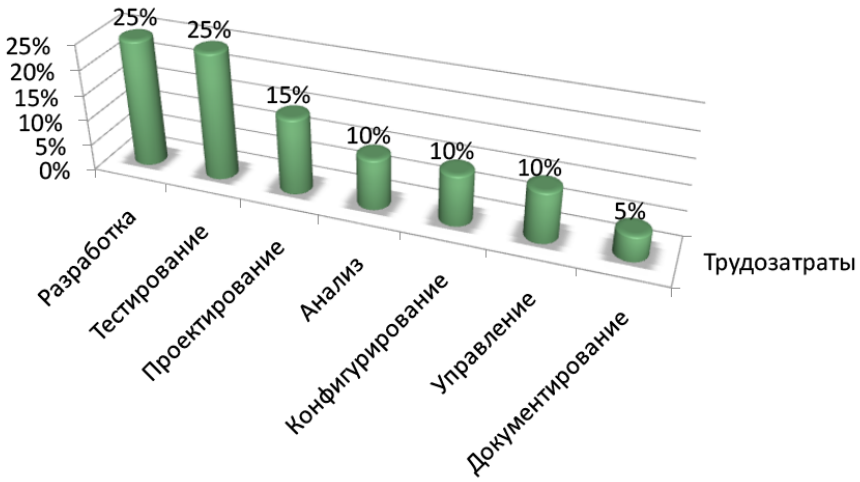
После того, как эксперты вытащили карты, они переворачиваются и модератор объявляет результаты. В итоге группа получает диапазон оценок, который может быть очень широким. Для приближения оценок экспертов модератор просит объяснить свою точку зрения того эксперта, у которого получилась самая низкая оценка. Эксперт объясняет свою точку зрения и часто оказывается,

что он неверно понял требования. После этого модератор просит высказаться эксперта, сделавшего самую высокую оценку, и часто оказывается, что он выбрал не самый оптимальный подход к реализации. Выслушав мнения двух экспертов, как правило, требования становятся более понятными, и в ходе коллективного обсуждения группа выбирает единый подход к реализации. Модератор предлагает участникам оценки проголосовать второй раз. После второго тура голосования диапазон оценок экспертов уменьшается. Можно провести и третий тур голосования, однако, второго тура обычно достаточно чтобы, усреднить оценки экспертов и получить одну оценку трудоемкости реализации данной задачи.

В конечном итоге, все задачи проекта (или итерации проекта) получают свои оценки. Просуммировав эти оценки, руководитель проекта получает оценку трудоемкости всего проекта (или итерации).

Как правило, команда делает оценку только тех задач, которые связаны с разработкой (программированием). В индустрии разработки программного обеспечения есть диаграмма (см. рис. 7), которая позволяет сделать общую оценку работ по проекту, связанному с разработкой ПО.

Из этой диаграммы видно, что в общем объеме работ проекта по разработке ПО кодирование занимает примерно 25% трудозатрат. Значит, для того чтобы правильно рассчитать бюджет проекта, нужно прогнозные трудозатраты на разработку необходимого функционала ПО умножить на 4, что в итоге позволит учесть оставшиеся виды работ.



© Лекции по управлению программными проектами, Сергей Архипенков

Рисунок 7

После того как мы сделали оценку объемов работ, остается умножить оценочные часы разработчиков, тестировщиков, бизнес-аналитиков и руководителя проекта на их ставки человека часа, и мы получим основную статью затрат в бюджете проекта – заработная плата участников проекта. Далее руководителю проекта нужно рассчитать плановое значение по другим статьям затрат:

- налоги с заработной платы – как правило, доля налогов от заработной платы – это известная цифра, которую остается умножить на полученное значение по статье «Заработная плата»;
- закупка «железа» (компьютеров, серверов, планшетов и т. д.) – эту статью бюджета необходимо согласовать с заказчиком после сбора и анализа требований к «железу», выбора вариантов «железа»;

- приобретение лицензионного программного обеспечения (СУБД, операционные системы, программа для управления проектами и т. д.) – для некоторых проектов затрат по этой статье может не быть, т. к. все необходимое ПО уже есть у заказчика;
- резерв на риски проекта – обычно, этот резерв можно рассчитать после анализа рисков проекта, либо рассчитать, используя процент от затрат на заработную плату команды проекта;
- накладные расходы на содержание офиса;
- плановая прибыль проекта – определяется руководством компании, которая является исполнителем проект.

Пример расчетов оценки стоимости

Команда проекта разработки программного продукта от заказчика проекта получила техническое задание на разработку программного продукта, которое содержит описание процессов, описание сценариев использования и требования к программному продукту. Задача команды проекта состоит в том, чтобы рассчитать бюджет проекта.

Первое, что надо сделать, – разработать список работ по проекту. Как правило, у руководителя проекта не хватает компетенций, чтобы сделать детализированный список работ, поэтому он приглашает компетентных в предметной области экспертов и просит их сделать переход от функциональных требований к подробному списку задач.

Можно отдать требования на изучение системному архитектору, чтобы он на черновике набросал архитектуру продукта, после чего команда определит список работ. Далее команда делает оценку трудоемкости методом покерного планирования и получает оценку трудоемкости задач по разработке необходимых функций.

Допустим, полученная в результате покерного планирования оценка работ по разработке показала, что на кодирование команде нужно будет 1000 чел.-часов.

Исходя из полученных данных рассчитаем плановое значение статьи затрат «Зарплаты команды проекта»:

Вид работ	Трудозатраты, чел.-часов	Ставка чел.-часа на данный вид работ, $	Стоимость, $
Кодирование функционала, описанного в ТЗ	1000	15	15 000
Тестирование	1000	12	12 000
Проектирование ПО	600	20	12 000
Анализ требований	400	14	5 600
Конфигурирование	400	15	6 000
Управление проектом	400	20	8 000
Документирование	200	10	2 000
Итого	4000	–	60 600

Рассчитаем оставшиеся статьи затрат по проекту:

Статья затрат	Формула расчета	Стоимость, $	Комментарий
Заработная плата участников проекта		60 600	
Налоги с заработной платы	45% от статьи «Заработная плата»	27 270	
Закупка «железа»	По стоимости счета на приобретение товара	10 000	По инвойсу от поставщика сервера
Приобретение лицензионного программного обеспечения		0	В данном проекте не требуется закупка
Резерв на риски проекта	10% от заработной платы участников	6 060	
Плановая прибыль проекта	25% от заработной платы участников	15 150	
Накладные расходы на содержание офиса	5% от заработной платы участников	3 030	
Итого: бюджет проекта		**116 050**	

Исходя из полученных выше расчетов, руководитель проекта может сформировать ставку 1 чел.-часа работы участника команды проекта для заказчика проекта:

Ставка 1 чел.-часа = 116 050 / 4000 = $29

Итак, бюджет проекта рассчитан, осталось согласовать его с заказчиком проекта.

Планирование сроков проекта. Метод критического пути

Для планирования сроков ИТ-проекта можно использовать метод критического пути, который предполагает использование сетевого графика проекта и некоторых расчетов.

Команда проекта делает прогноз по трудоемкости задач, а руководитель проекта получает информацию о том, сколько времени может уделять задачам проекта каждый участник. После этого, руководитель проекта делает расчет сроков реализации каждой задачи, используя формулу:

Срок по задаче = Трудоемкость задачи * 100 % / % доступности исполнителя

где

% доступности исполнителя – доля от рабочего дня, которую исполнитель может уделить работе над задачей.

Далее команда проекта определяет зависимости в реализации задач и проектирует сетевой график.

При заполнении данных в поле «Предшественник» команда указывает название задачи, которая должна быть завершена до старта этой задачи:

Название задачи	Длительность, дней	Предшественник
A	2	–
B	15	A
C	10	A

Название задачи	Длительность, дней	Предшественник
D	13	A
E	18	A
F	15	C, D
G	10	F, B
H	5	E, G

Для того чтобы определить критический путь, команда создает сетевой график работ:

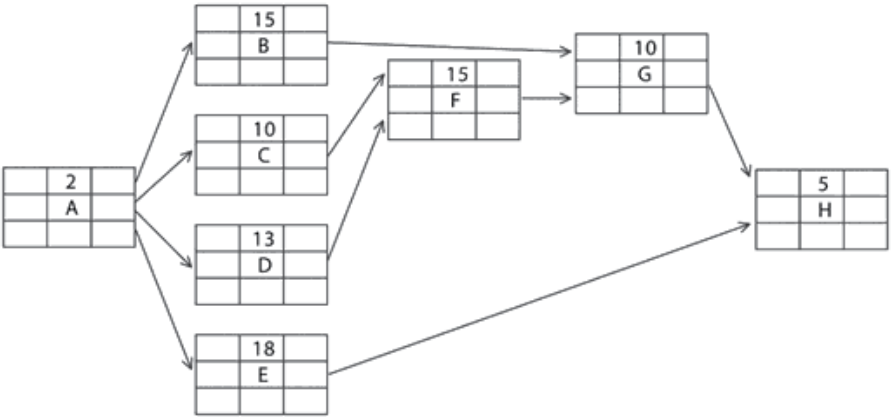


Рисунок 8

Параметры задач на диаграмме располагаются в прямоугольнике. Легенда для описания значений ячеек прямоугольника представлена ниже:

РАННИЙ СТАРТ	ДЛИТЕЛЬНОСТЬ	РАННИЙ ФИНИШ
КОД ЗАДАЧИ		
ПОЗДНИЙ СТАРТ	РЕЗЕРВ	ПОЗДНИЙ ФИНИШ

Очередность выполнения задач проекта отображается стрелками, например, задача C может начинаться только тогда, когда будет завершена задача A, а задача H стартует тогда, когда полностью будут завершены задачи E и G.

Чтобы упростить расчеты параметров графика, договоримся использовать не календарные даты, а номера дней. Также для простоты расчетов примем, что задача A может начинаться прямо сегодня, а сегодняшний день имеет номер 0 (ноль).

Для задачи A рассчитаем параметры ранних дат старта и финиша:

Ранний старт задачи A – день с номером 0.

Ранний финиш задачи A рассчитывается как «ранний старт» плюс «длительность», т. е. 0 + 2 = 2-й день.

Сделаем еще одно допущение: следующую задачу можно начинать в тот же день, когда заканчивается предыдущая задача.

Выполним расчеты для задачи B:

Ранний старт: день с номером 2.

Ранний финиш: 2 + 15 = 17-й день.

Таким же образом выполняются расчеты дат раннего старта и раннего финиша для всех задач, указанных в сетевом графике. В том случае, когда у одной задачи несколько задач-предшественников (например, у задачи H), при расчете дня раннего старта выбирается наибольшее значение из имеющихся.

Для определения самой длинной цепочки задач в проекте необходимо рассчитать даты позднего старта и позднего финиша по задаче.

Для их расчета используются следующие правила:

1. Поздняя дата финиша данной задачи равна поздней дате старта задачи-последователя.
2. Если задач-последователей несколько, как у задачи A, для определения поздней даты финиша выбирается наименьшее значение из имеющихся.
3. Поздняя дата старта задачи определяется как разница между ее поздним финишем и длительностью.

Остается определить критические задачи в графике. Для этого следует рассчитать резервы по задачам. Значение резерва по задаче рассчитывается как разница между значением позднего старта задачи и значением раннего старта данной задачи. Те задачи, у которых резерв времени равен нулю, называются критическими и определяют критический путь проекта. Задачи, у которых значение резерва отличается от нуля, не являются критическими. Это обозначает, что для этих задач срок реализации может быть увеличен на время их резерва или старт задачи можно перенести на дату позднего старта.

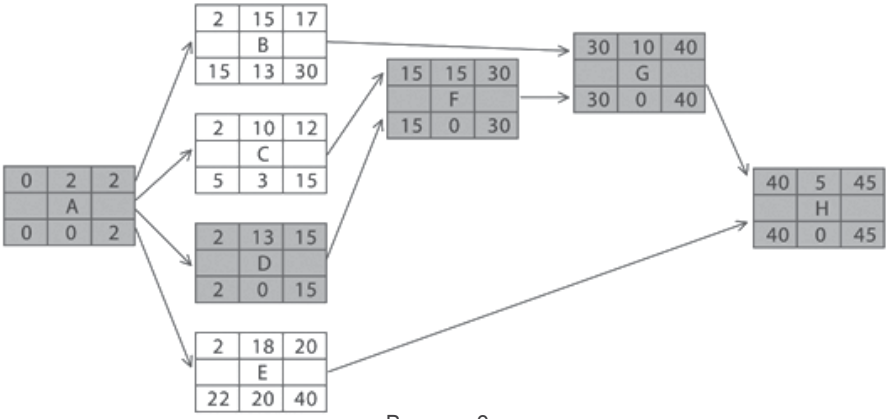


Рисунок 9

Для чего руководителю проекта полезно знать список и параметры задач, лежащих на критическом пути?

- Он получает представление об оптимистическом значении длительности проекта.
- Он понимает, для каких задач требуется усиленный контроль , т. к. задачи критического пути определяют срок всего проекта.
- Руководитель проекта может назначить исполнителей на задачи проекта, увидеть, есть ли перегрузки у некоторых исполнителей, выровнять перегрузки и узнать реальное значение критического пути и сроков проекта.
- В случае срыва сроков финиша по задачам критического пути, руководитель проекта может быстро спрогнозировать, как изменится срок проекта и продумать, как откорректировать план проекта, чтобы уложиться в срок.

Методу критического пути уже скоро 60 лет, но для проектов, в основе которых лежит водопадная модель жизненного цикла, он все еще актуален для планирования сроков проектов.

На сегодняшний день в мире существует огромное количество компьютерных программ, которые автоматизируют расчеты параметров сетевого графика.

Участники и персонал проекта

Для того, чтобы повысить вероятность успешности ИТ-проекта, в составе участников проекта должны выполняться обязательные роли.

Чаще всего в ИТ-проектах есть две стороны проекта: компания-заказчик и компания-исполнитель, но иногда, ИТ-проект делается силами внутреннего ИТ-подразделения компании. Рассмотрим, как распределяются роли в ИТ-проекте в первом случае.

Участники проекта со стороны заказчика

В мире существует много подходов к управлению проектами и в каждом из них описывается свое видение того, какие роли должны выполняться в проекте.

Например, в PMBOK, описана следующая модель ролей для проекта:

Название роли	Ответственность	Полномочия
Руководитель проекта	За все параметры проектного треугольника: достижение целей в плановый срок и в плановый бюджет	В проектной организационной структуре: выбор исполнителей в команду, формирование команды, планирование и контроль выполнения задач
Заказчик проекта	Утверждение документа по требованиям к ИТ-проекту (например, SRS). Организация и выполнение работ по приемке полученных результатов проекта	Изменение требований к результатам проекта, определение приоритетов в реализации требований
Спонсор проекта	Утверждение целей проекта, плановых сроков и бюджета. Обеспечение того, что согласованные исполнители на проект выделяют запланированное на задачи проекта время	Решение вопросов, на которые не хватает власти у руководителя проекта и заказчика проекта

Название роли	Ответственность	Полномочия
Команда проекта	Выполнение задач проекта в согласованные сроки и запланированные трудозатраты	Максимально рано сигнализировать руководителю проекта о возникновении проблем при решении поставленных задач

Для примера рассмотрим проект по внедрению системы ERP (*англ. Enterprise Resource Planning, планирование ресурсов предприятия*) в некоторой компании, руководство которой приняло решение для реализации проекта привлечь компанию, специализирующуюся на внедрении ERP-систем.

Очевидно, что роль заказчика проекта нужно выполнять кому-то из сотрудников компании, в которой внедряется ERP.

Может ли роль заказчика проекта выполнять несколько человек, например, все руководители подразделений, которые будут использовать ERP? Вы слышали поговорку: «У семи нянек дитя без глаза»? Эта поговорка про проект, в котором роль заказчика выполняет несколько человек, среди которых нет главного. Заказчиком проекта должен быть один человек, правда – это не мешает ему собрать рабочую группу (или команду), с которой он будет советоваться по поводу принятия важных решений по проекту, но ответственность за принятые решения он должен нести один.

Чтобы принять решение о назначении конкретного сотрудника на роль заказчика в проекте внедрения ERP, стоит ответить на следующие вопросы:

1. Сотрудники каких подразделений компании будут использовать ERP?
2. Кто из руководителей этих подразделений больше всего заинтересован в том, чтобы ERP-система помогла ему улучшить показатели, за которые он отвечает?
3. Может ли кандидат на роль заказчика проекта тратить на него минимум 4-6 часов в неделю?

Для расчета требуемого времени заказчику проекта на выполнение его функций можно использовать данные компании Standish Group, о том, что на каждую 1000 долларов США от такой статьи затрат как «Стоимость времени людей» в ИТ-проекте приходится принимать 1,5 решения. Предположим, проект внедрения ERP-системы оценен в 250 000$, из них стоимость лицензий программного обеспечения коробочной ERP-системы составляет 100 000$, а 150 000$ составляет стоимость доработок и внедрения ERP (а это и есть. стоимость времени людей). Если верить расчетам Standish Group, то в этом проекте нужно будет принять около 225 важных решений, причем Standish Group указывает на то, что заказчик должен принять участие как минимум в 20% из этих решений. В итоге получается, что заказчик примет решение в 45 случаях. Предположим, что в среднем на принятие одного решения заказчик тратит 4 часа, получается 45 важных решений по 4 часа на каждое – это 180 часов.

Если продолжительность проекта составляет примерно 1 год, то 180 часов деленное на 48 недель, дает результат – 3,75 часа в неделю заказчик должен тратить на принятие решений в проекте.

При этом, очевидно, что заказчик будет тратить время не только на принятие важных решений, т. к. ему еще нужно:

1. Обеспечить наличие видения проекта, в котором описать, как в продукте будут удовлетворены ожидания разных заинтересованных сторон.
2. Согласовать ограничения проекта по срокам, бюджету и содержанию (эти ограничения должны быть прописаны в договоре на проект или в Уставе проекта).
3. Выбрать метод измерения прогресса проекта, который должен быть с одной стороны не требующим много времени на измерение, а с другой стороны – предоставляющим правильные данные о ходе проекта.
4. Согласовывать документы по требованиям.
5. Организовывать работы по приемке разработанного функционала.
6. Организовывать работы по обучению пользователей.
7. Анализировать отчеты руководителя проекта о ходе проекта.
8. Участвовать в создании и корректировке плана проекта – владелец продукта должен понимать прогнозы по завершению проекта (хватит ли бюджета, уложимся ли в срок) и принимать решения по приоритетам реализации требований к продукту проекта.

Вот и получается, что 4 часа в неделю – это минимум времени, который нужен на адекватное выполнение роли заказчика проекта ERP. Заметим, что в некоторых ИТ-проектах у заказчика может уходить все его рабочее время на выполнение роли заказчика проекта.

В некоторых подходах к управлению проектами (например, в Scrum) заказчика проекта называют «владельцем продукта проекта» и это название отражает суть его деятельности: создать продукт, который будет ценен для его потребителей. Владелец продукта должен быть заинтересован в получении выдающегося продукта в плановые сроки и в утвержденный бюджет.

Нужен ли спонсор проекту внедрения ERP?

На наш взгляд – это одна из ключевых ролей в успехе проекта. По статистике PMI слабая поддержка спонсора проекта попала на девятое место в списке факторов провала проектов (см. рис. 10).

Спонсор (куратор) проекта – руководитель высшего звена организации, в которой будут пользоваться результатами проекта. Спонсор (куратор) проекта отвечает за достижение проектом конечных целей и реализацию выгод для организации.

В чем отличие роли спонсора проекта от заказчика?

1. **Спонсор проекта** должен иметь больше власти, чем заказчик проекта иначе толку от него не будет.
2. **Спонсор** должен отвечать за создание ценности в проекте и передачи этой ценности в бизнес-подразделения для эксплуатации.

Рисунок 10

Спонсором проекта внедрения ERP, на наш взгляд, должен стать CEO компании или кто-то из его заместителей. Внедрение ERP – это дорогостоящий проект, результатом которого станет изменение многих бизнес-процессов компании и если этому проекту не придать нужного статуса, то он может оказаться провальным. Потому если роль спонсора проекта возьмет на себя CEO компании, проект только выиграет от такого решения.

Нужно ли компании, заказывающей проект, назначать **руководителя проекта** из числа своих сотрудников? Ответ зависит от того, какую долю работ от общего объема работ в проекте будут выполнять сотрудники компании-заказчика. Если это менее 20-25%, то на наш взгляд, руководитель проекта от заказчика не нужен, при

большей доле – его стоит назначить, т. к. кому-то надо будет планировать, организовывать и контролировать выполнение задач, которые будут поручены сотрудника компании-заказчика.

Персонал проекта со стороны фирмы разработчика

Со стороны компании-разработчика в ИТ-проекте обычно участвует команда, которая способна решить задачи проекта, стоящие перед ней. У этой команды обязательно должен быть руководитель проекта. Состав остальных участников сильно зависит от работ, которые необходимо выполнять в проекте. Но обычно в состав команды входят:

- разработчики;
- тестировщики;
- бизнес-аналитики;
- в некоторых случаях в команду проекта нужны еще дизайнеры интерфейсов, системный архитектор, технический писатель.

Должен ли руководитель ИТ-проекта иметь опыт работы программистом, аналитиком или тестировщиком, чтобы возглавить команду?

На этот вопрос есть следующий ответ: чем масштабнее проект, тем меньше нужно руководителю проекта быть предметником и больше нужно навыков в управлении проектами (см. рис. 11).

Рисунок 11

Рассмотрим подробнее роль руководителя проекта.

Роль руководителя проекта и его ответственность за проект по-разному трактуется в различных подходах управления проектами. Так, подход **MSF (Microsoft Solutions Framework)** предполагает, что за успех проекта ответственна вся команда, состоящая из следующих ролевых кластеров:

- **управление программой (program manager)** – отвечает за разработку архитектуры решения, планирование и контроль сроков проекта;
- **разработка (developer)** – отвечает за разработку продукта;
- **тестирование (QAE)** – отвечает за планирование, разработку тестов и отчетность по тестам;

- **управление выпуском (release manager)** – отвечает за создание инфраструктуры, выпуск готового продукта;
- **удовлетворение заказчика (user experience)** – отвечает за обучение, эргономику, графический дизайн, техническую поддержку;
- **управление продуктом (product manager)** – отвечает за расстановку приоритетов, маркетинг продукта, соблюдение интересов заказчика.

Получается, что в MSF ответственность за успех проекта коллективная, но за реализацию проекта в срок и в бюджет отвечает роль Управление программой.

В подходе PMBOK за успех проекта отвечает руководитель проекта, а успех измеряется тем, смогла ли команда проекта выполнить его в плановый срок, уложиться в запланированный в бюджет и достичь поставленных перед проектом целей.

Какими компетенциями должен обладать руководитель ИТ-проекта? На наш взгляд, в первую очередь, он должен обладать мягкими навыками, такими как:

- умение подобрать в проект правильных людей и создавать из них команду;
- навыки ведения переговоров;
- умение замотивировать участников команды;
- умение выбрать наиболее подходящий подход (или гибрид подходов) для управления проектом и внедрить его на проекте;
- навыки внедрения средств автоматизации проектной работы.

Умение подбирать правильных людей и создавать из них команду – это самый сложный навык, который нужно приобрести руководителю проекта. Ниже мы рассмотрим подходы, которые могут помочь руководителю проекта при подборе людей в команду.

Принципы отбора сотрудников в команду проекта

Чем команда отличается от группы людей, совместно работающих над проектом?

На наш взгляд, команда должна обладать следующими характеристиками:

1. Доверие членов команды друг к другу, которое проявляется в том, что ни один участник команды не сомневается в том, что все остальные не хотят ему навредить или манипулировать им.
2. Конструктивные споры о деле. Если внутри команды люди не спорят о деле – им все равно, а это приводит к принятию не самых оптимальных решений.
3. Осознание общей ответственности за результат. Каждый участник стремится не просто выполнить свои задачи, но и оказать помощь коллегам.
4. Анализ производительности команды. Команда определяет метрики, с помощью которых она анализирует свою результативности.
5. Постоянное улучшение командной работы. Участники команды периодически встречаются и определяют, что еще можно и нужно улучшить, ставят задачи и внедряют изменения в процессах совместной работы.

При подборе сотрудников в команду важно учитывать не только их профессиональные навыки, но и личностные качества.

Во многих компаниях для оценки личностных качеств сотрудников используется **подход MBTI**, который имеет несколько практических аспектов:

1. Знание психотипа сотрудника позволяет менеджеру подбирать правильные каналы коммуникаций, спрогнозировать стиль поведения сотрудника в работе и конфликтах.
2. Знание MBTI помогает менеджеру понимать особенности своего коллектива и разрабатывать планы обучения участников команды мягким навыкам.
3. MBTI помогает понять при решении каких задач сотрудник проявит себя наилучшим образом, а какие решать ему будет трудно.

Методика MBTI позволяет определить предпочтения каждого человека по четырем аспектам, а затем содействовать наилучшему применению этих предпочтений в жизни и работе (см. рис. 12).

- **Источник энергии: экстраверты и интроверты**

Экстраверт получает энергию в коммуникациях с другими людьми, а интроверт черпает ее в уединении.

- **Функция сбора информации: сенсорики и интуиты**

Сенсорики, собирая информацию, доверяют своему опыту. У интуитов сбор информации основан на предположении о том, что любая информация может иметь определенную степень достоверности, даже если в опыте подобного не было. Интуит – стратег, а сенсорик – тактик.

Рисунок 12

- **Функция принятия решений: думающие и чувствующие**

Логики при принятии решений не думают о том, что почувствуют те, кого касается это решение. Чувствующие же наоборот, принимают решение с учетом того, какое воздействие оно окажет на других людей.

- **Стиль жизни: рационалы и иррационалы**

Рационалы любят принимать решения, наличие открытых вопросов, по которым нет ответов вызывает у них стресс. Иррационалы предпочитают собирать информацию для принятия решений и оттягивают момент принятия решения до последнего.

У каждого из нас есть предрасположенность к тому или иному полюсу по шкале экстраверсия-интроверсия, сенсорика-интуиция, логика-чувствование, рационализм-иррационализм. Каждый человек может проявлять себя как экстраверт в одних ситуациях и как интроверт в других, при этом склонность к одному из этих качеств будет выше.

Всего в MBTI существуют 16 типов:

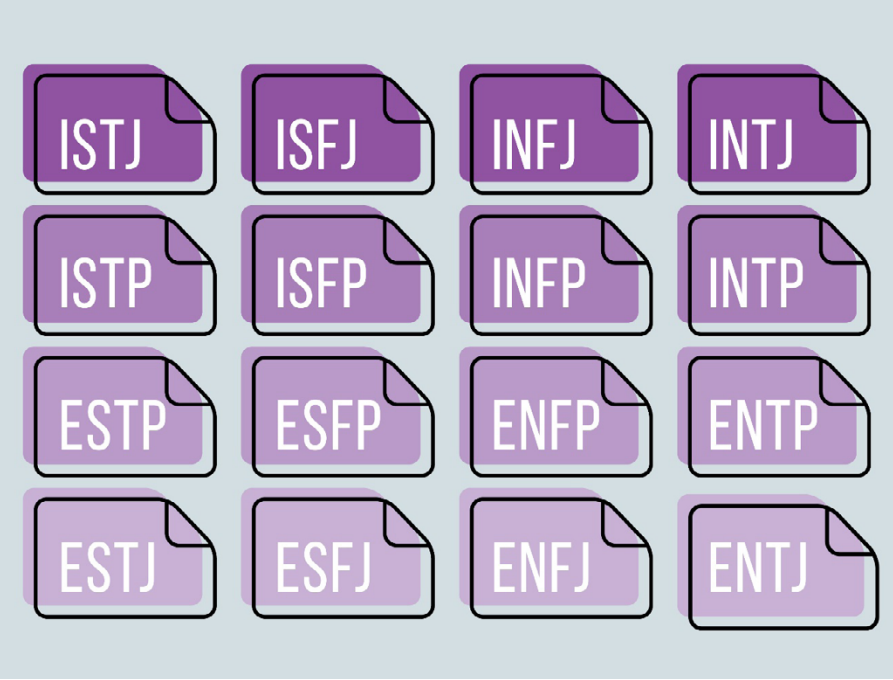


Рисунок 13

Что интересно, около 70% компаний Fortune 500 успешно применяют MBTI® это говорит о популярности подхода MBTI (http://www.ey.com/RU/ru/Services/Specialty-Services/Academy-of-Business/MBTI-Overview).

Еще одним важным моментом при отборе сотрудников в команду проекта является изучение того, что мотивирует каждого потенциального участника на работу в данном проекте.

Идеально, если ключевым мотиватором для кандидата являются не деньги, а внутренние мотивы, такие как: желание создать продукт мирового уровня, желание получить новый опыт успешной работы в команде и т. д.

Руководителю проекта необходимо обладать навыками определения доминирующих мотиваторов потенциальных участников проекта.

Управление командой проекта

Что такое управление командой? С нашей точки зрения, управление состоит в том, чтобы измерять показатели результативности команды и корректировать поведение команды так, чтобы эти показатели достигали своего максимума.

При использовании фреймворка Scrum команда может измерять два ключевых показателя: **Фокус фактор (Focus Factor)** и **скорость (Velocity)**.

Пример расчета Focus Factor

Команда взяла ответственность реализовать за один спринт 20 задач трудоемкостью в 400 человеко-часов. По завершении спринта команда смогла продемонстрировать результат 14 задач из 20-ти запланированных. По плановым оценкам трудоемкости эти 14 задач оценивались в 200 человеко-часов. ***Focus factor*** *данного спринта составил 200 / 400 = 0,5, при этом* ***velocity*** *команды составила 200 человеко-часов.*

Если измерять этот показатель после каждого спринта и отслеживать его тренд, то можно судить о том, на какой стадии жизненного цикла находится команда проекта. Теорию жизненного цикла команды разработал **Б. Такман**. Он выделил следующие стадии групповой динамики в команде:

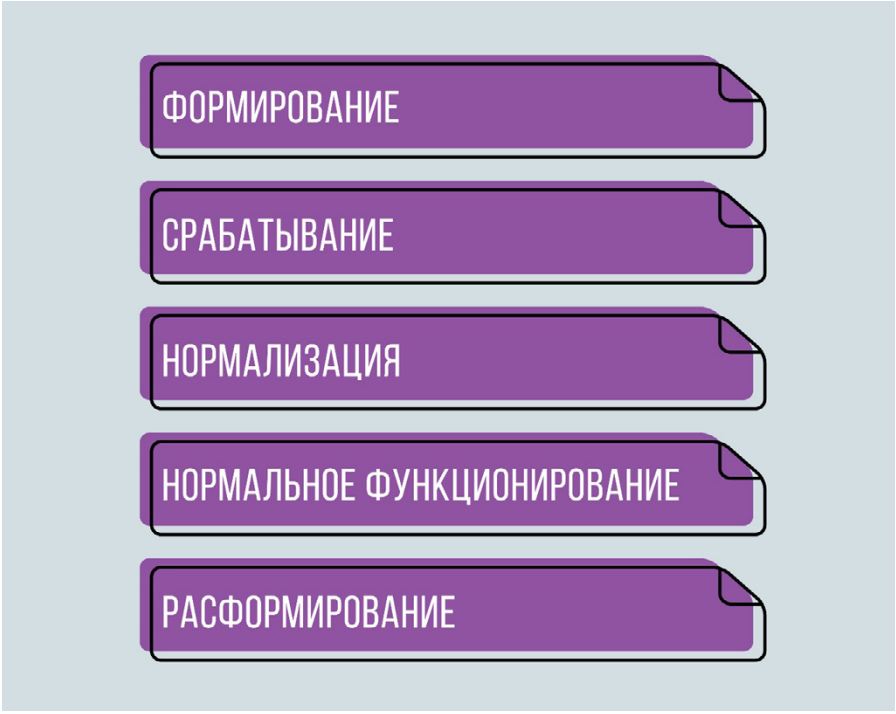


Рисунок 14

Производительность команды обычно изменяется так, как показано на рисунке 15.

Если фокус-фактор команды колеблется в первых спринтах от 0,3 до 0,5 и то растет, то падает, это говорит

о том, что команда находится на этапе срабатывания и никак не может преодолеть его. Если фокус-фактор на протяжении 2-3 спринтов устойчиво растет, то команда, скорей всего перешла на этап нормализации. И если фокус-фактор продолжает устойчивый рост на протяжении 3-6 спринтов, при этом достигает уровня 1 и находится на этом уровне несколько спринтов, то можно констатировать, что команда достигла этапа нормального функционирования.

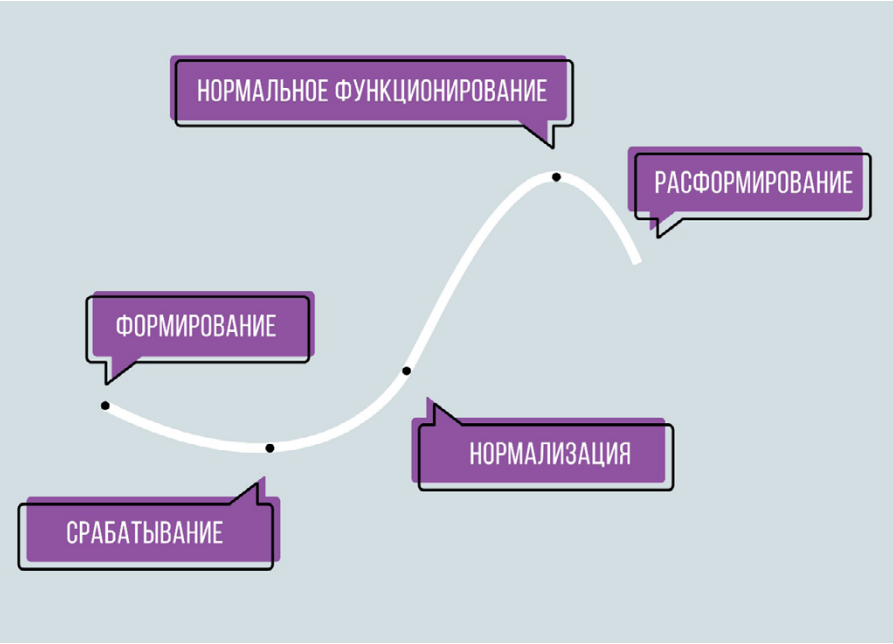


Рисунок 15

Роли в рамках проекта

В индустрии разработки и внедрения программных продуктов за десятилетия ее развития появилось достаточно много профессий. Эти профессии повлияли и на появление новых ролей в ИТ-проекте:

1. **Project manager** – отвечает за реализацию проекта в срок и бюджет и достижение стоящих перед проектом целей.
2. **Бизнес-аналитик** – занимается извлечение и анализом требований к программному продукту, поиском и анализом проблем, выработкой решения для выявленных проблем, документированием требований.
3. **Архитектор продукта** – отвечает за проектирование продукта и описание его архитектуры.
4. **Разработчики** – кодируют требования так, чтобы они соответствовали описанию и созданная функциональность продукта проходила тест-кейсы.
5. **Тестировщики** – разрабатывают сценарии тестирования требований, проводят тестирование и фиксируют выявленные в ходе тестирования отклонения от требований (в том числе ошибки).
6. **Team Leader** – в некоторых проектах количество участников превышает 9 человек и руководитель проекта принимает решение разбить их на несколько команд. Для планирования и контроля результатов команд в них выбираются Team Leader, которые отвечают иногда и за развитие нужных для проекта компетенций у участников своей команды.

2. РИСКИ В ПРОЕКТЕ

Что такое риски проекта?

В любом ИТ-проекте, несмотря на то, сколько времени руководитель проекта потратил на планирование, что-то пойдет не так. Для того чтобы не бороться с последствиями, а предвидеть и не допускать проблем в проекте, нужно научиться управлять рисками проекта.

Определений термина «риск» существует несколько:

> ***Риск** – это неопределенное событие или условие, наступление которого отрицательно или положительно сказывается на целях проекта.*
>
> © PMBOK

> ***Риск** – это влияние неопределенности на достижение целей.*
>
> © ГОСТ Р 56275-2014

По сути, риск для проекта – это то, что создаст проблему в будущем, потому нам импонирует такое определение: риск – это потенциальная проблема проекта, которая может возникнуть в будущем.

То есть, на текущий момент эта проблема еще не возникла, но если она возникнет, то окажет влияние на реализацию проекта в срок и в плановый бюджет, а, возможно, приведет даже к тому. Что цель проекта окажется не возможным реализовать в полной мере.

Суть управления рисками ИТ-проекта состоит в том, чтобы управлять «проактивно», не допуская возникно-

вения проблем, а не заниматься бесконечным «тушением пожаров».

По нашему мнению, большинство руководителей проекта слишком полагаются на свою интуицию и опыт при планировании проекта и не хотят посмотреть на проект «без розовых очков».

Если вернуться к исследованию Pulse of the profession 2017 года, то неопределенные риски проекты попали на 8-е место в списке причин провала проектов.

В большинстве случаев риск можно устранить или снизить его влияние на проект, приложив незначительные усилия или затратив небольшие деньги.

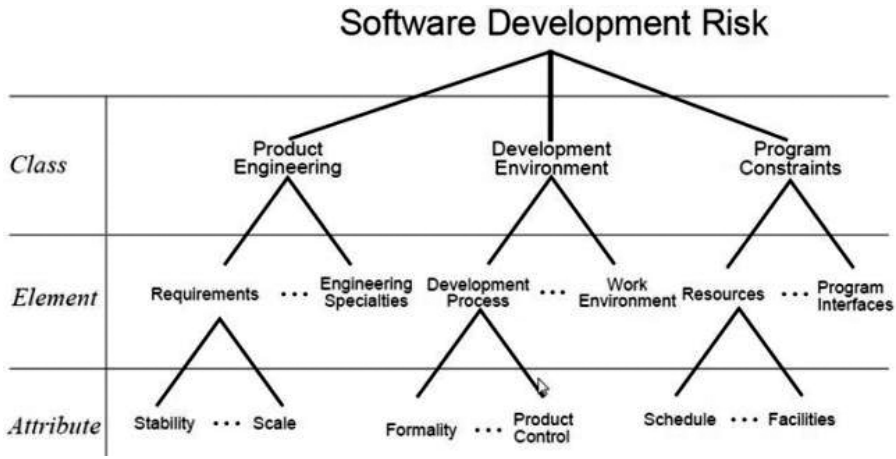
Если бы выгоды от управления рисками в проектах были очевидны и доказаны, то это стало бы стандартной практикой. Но, есть следующие стереотипы, которые отвлекают руководителей проекта от управления рисками*:*

1. Если заниматься идентификацией рисков в ИТ-проекте, то некогда будет работать.
2. Длинный список рисков отобьет всякое желание реализовывать проекта.
3. Плохие мысли притягивают плохие события. Лучше не думать о плохом.
4. Кто не рискует, тот не пьет шампанского.

На наш взгляд, проект только выиграет даже от того, если идентифицировать хотя бы пару десятков рисков, и предпринять по ним антирисковые мероприятия – это резко повысит устойчивость проекта к внешним потрясениям.

Типы рисков

В 1993 году **SEI (Software Engineering Institute)** опубликовал классификатор рисков ИТ-проектов – **Taxonomy-Based Risk Identification**. Несмотря на достаточно объемное содержание, этот документ достаточно прост в использовании. Он описывает **три класса рисков для софтверных** проектов, каждый класс декомпозируется на элементы и атрибуты.



© http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=11847

Рисунок 16

Для идентификации рисков ИТ-проекта можно изучить все элементы рисков, описанные в документе, и оставить только те из них, которые с вашей точки зрения, актуальны для вашего ИТ-проекта. После чего можно окунуться вглубь и изучить атрибуты и конкретные риски, связанные с этими атрибутами. В результате у вас останется сухая выжимка – список актуальных для вашего проекта рисков, которые следует внести в документ Реестр рисков, правильно сформулировав риски.

Процесс управления рисками

Процесс управления рисками состоит из следующих действий:

- идентификация рисков (поиск рисков и их фиксация);
- качественный анализ рисков (оценка важности рисков);
- планирование антирисковых мероприятий (добавление в список задач проекта новых задач, связанных со снижением, передачей или активным принятием рисков);
- мониторинг и контроль рисков.

Идентификация рисков

Проводится для того, чтобы обнаружить как можно больше рисков ИТ-проекта и записать их в документ Реестр рисков.

Идентификация рисков – это итеративный процесс, поскольку по мере развития проекта в рамках его жизненного цикла могут обнаруживаться новые риски. Частота итераций и состав участников итерации в каждом случае могут быть разными.

Для руководителя проекта важно научиться делать правильные формулировки риска.

В MSF предлагают следующий формат для формулирования рисков (см. рис. 17).

Качественный анализ рисков

Список рисков может получиться очень длинным, при этом разные риски могут иметь разную важность для проекта. Как оценить важность рисков?

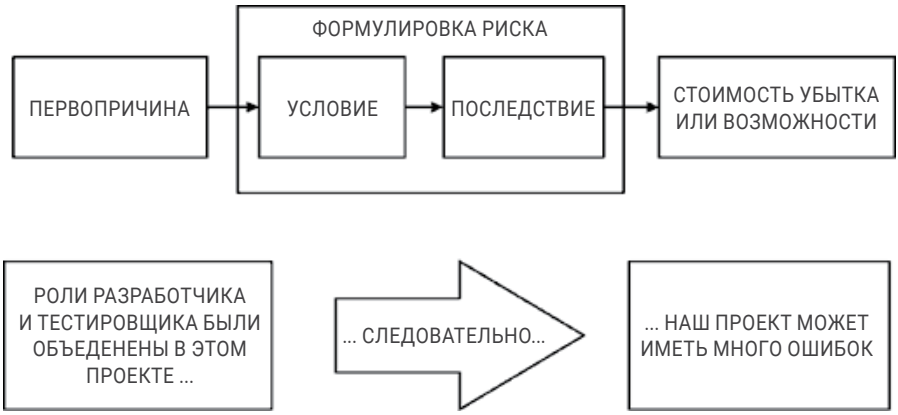


Рисунок 17

Для оценки важности рисков используется две характеристики: вероятность возникновения риска и его влияние на проект.

Зная эти два параметра, можно высчитать важность риска.

Как оценить вероятность возникновения риска?

Можно использовать статистику по случаям, которые приводят к материализации риска или экспертное мнение.

К примеру, вы нашли несколько профессионалов, которые готовы помочь вам с оценкой вероятности. Предложите им выбрать интервал вероятностей возникновения риска:

Интервал вероятностей	Словесная формулировка	Числовая оценка
От 0% до 33%	Низкая	1
От 34% до 67%	Средняя	2
От 68% до 100%	Высокая	3

Для того чтобы оценить влияние риска на проект, можно использовать следующую матрицу влияния:

Словесная формулировка	Числовая оценка
Низкое	1
Среднее	2
Высокое	3

А для расчета влияния на проект используется формула:

Важность риска = Вероятность x Влияние

Планирование реагирования на риски

Для рисков с самым высоким значением важности, руководитель проекта должен продумать такие мероприятия, которые позволят ему убрать риски из проекта или резко снизить их вероятность или влияние на проект.

Для выработки антирисковых мероприятий используется четыре возможные стратегии работы с рисками. Следует продумать конкретные варианты реагирования по каждой стратегии и выбрать те из них, которые стоят дешевле, чем вероятностный ущерб от возникновения риска.

В PMBOK описывают 4 стратегии работы с рисками:

1. **Уклонение.** Предполагает выбор таких мероприятий, которые позволят полностью исключить возникновение проблемы, и тем самым избавить проект от последствий риска.

2. **Передача**. Предполагает поиск возможности передать риск вместе с ответственностью за реагирование на риск на третью сторону (например, в страховую компанию). Передача риска означает, что в случае материализации риска третья сторона выплатит руководителю проекта или заказчику материальную компенсацию, однако это не позволяет быть уверенным, что риск не случится
3. **Снижение.** Подразумевает продумывание и планирование мероприятий, которые позволят снизить вероятность и/или последствия от возникновения риска до приемлемых пределов
4. **Принятие**. Есть два варианта для стратегии принятия – активное и пассивное принятие. Активное принятие предполагает создание резерва времени и денег на устранение последствий материализации риска. Пассивное принятие предполагает наличие запасного плана на случай материализации риска.

Контроль рисков

Периодически руководителю проекта рекомендуется оценивать статус рисков: случился риск или нет, пересматривать важность рисков и идентифицировать новые риски проекта.

Для этого необходимо вовлечь участников команды в процесс управления рисками и убедить их в том, что работа с рисками проекта сильно повышает вероятность уложиться в срок и в бюджет.

3. УПРАВЛЕНИЕ КАЧЕСТВОМ В ПРОЕКТЕ

Что такое управление качеством?

В ИТ-индустрии используется три понятия, которые иногда отождествляются, но на самом деле являются принципиально отличными: **контроль качества (QC)**, **обеспечение качества (QA)** и **тестирование продукта**.

Тестирование проводится после того, как продукт или его часть была создана, а контроль качества – это набор активностей, который гарантирует качество продукта на всех этапах его создания. Контроль качества (QC) сосредоточен на поисках дефектов в программном продукте и обеспечении действий по их исправлению. А тестирование является неотъемлемой частью контроля качества.

Обеспечение качества программного продукта (QA) – это набор действий, который отвечает за разработку и внедрение процессов для улучшения всего жизненного цикла разработки продукта.

Соотношение между тремя понятиями хорошо иллюстрирует рисунок 18.

В функциональные обязанности QA-команды входят следующие функции:

- подготовка планов обеспечения качества проекта;
- проверка соответствия процессов проекта планам качества;
- проведение регулярных проверок продуктов проекта;

QA, QC and Testing in software development process

Рисунок 18

- сообщения вышестоящему руководству о том, что есть отклонения от стандартов;
- контроль наличия процедур управления изменениями проектов;
- контроль наличия процедур управления конфигурациями продуктов;
- проведение извлечения уроков в процессе контроля качества и воплощением рекомендаций, основанных на усвоенных уроках.

В некоторых организациях функцию QA выполняет Проектный офис компании. Это отвечает критериям независимости, однако, организации, что следуют этой модели, необходимо убедиться, что эта команда состоит из обученных и/или специализированных аналитиков по обеспечению качества.

Метрики

Контроль качества программного продукта не возможен без наличия метрик, которые позволяют оценить достижение того или иного уровня качества программного продукта.

> ***Метрика*** – *это количественный масштаб и метод, который может использоваться для измерения.*
>
> © ISO 14598

Один из способов группировки метрик является группировка по типам сущностей, участвующих в обеспечении качества и тестировании программного обеспечения:

1. Метрики по **тестовым случаям (Test Cases)**.
2. Метрики по багам/дефектам.

Рассмотрим каждую из групп метрик.

Метрики по тест-кейсам

Название метрики	Описание
Пройденные/провальные тест-кейсы	Соотношение количества удачно пройденных тест-кейсов к провальным. К концу проекта количество провальных тестов должно равняться нулю
Не запущенные тест-кейсы	Количество тест-кейсов, которые необходимо выполнить для данного этапа проекта. Имея данную информацию, мы можем проанализировать и выявить причины, по которым тест не были проведены

Метрики по багам

Название метрики	Описание
Открытые/ закрытые баги	Соотношение открытых багов к закрытым
Открытые заново / закрытые баги	Отношение количества багов, переведенных из состояния «закрыт» заново в состояние «открыт», к общему количеству закрытых
Отклоненные/ открытые баги	Отношение количества отклоненных багов к открытым

Практическое использование метрик

Для каждой метрики стоит продумать плановое значение и измерять фактическое.

Лучше всего вести графики по метрикам и отслеживать тренды. К примеру, если доля открытых заново багов к закрытым растет, необходимо провести анализ причинно-следственных связей и понять, что является коренными причинами открытия багов заново и как эти причины устранить.

Или если доля открытых к закрытым багам растет, то это сигнал для того, чтобы разобраться в причинах негативного тренда, которые могут быть следующими:

- неадекватная реализация программистами требований (непонимание требований);
- невнимательность разработчиков и плохое тестирование самим разработчиком перед передачей функционала в тестирование.

План контроля качества

План контроля качества – это документ, используемый, в частности, в методологии RUP. Данный документ содержит план оценки и контроля качества проекта и может ссылаться на ряд других артефактов, используемых в RUP.

Ниже представлены разделы, которые включает в себя шаблон Плана контроля качества, используемый в RUP:

- определения, акронимы и аббревиатуры;
- управление;
- документация;
- стандарты и рекомендации;
- метрики;
- план оценки и контроля;
- оценка и тестирование;
- устранение неполадок и корректирующие действия;
- инструменты, методики и методологии;
- управление конфигурацией;
- контроль за поставщиками и субподрядчиками;
- записи о качестве;
- обучение;
- управление рисками.

4. ДОКУМЕНТАЦИЯ И ДОКУМЕНТООБОРОТ

Цели и задачи документации в рамках проекта

Документирование проекта выполняется для достижения следующих целей:

1. Снижение вероятности того, что достигнутые договоренности между заказчиком проекта и руководителем проекта будут забыты или искажены через некоторое время после старта проекта.
2. Повышение вероятности того, что все заинтересованные стороны проекта будут одинаково понимать содержание проекта, планы реализации задач проекта и т. д.

Типы документации

Условно всю документацию в ИТ-проекте можно разделить на следующие типы:

1. Документы по управлению проектом:
   - Project Charter (Устав проекта);
   - WBS (ИСР – иерархическая структура работ проекта);
   - расписание проекта;
   - реестр рисков;
   - отчет о ходе проекта.

2. Документы по продукту проекта:
   - Vision and Scope;
   - Software Requirements Specification (SRS);
   - Product Backlog;
   - пользовательская документация.

Устав проекта

Устав проекта – документ, с которого начинается проект и по которому происходит оценка успешности проекта. Руководителю проекта Устав проекта нужен для того, чтобы зафиксировать в нем все важные аспекты проекта, согласованные с заказчиком. Устав проекта может стать приложением к договору и позволяет всем участникам проекта получить одинаковое представление о проекте.

В своих проектах мы используем шаблон Устава проекта, структура которого содержит следующие разделы:

Рисунок 19

Мы считаем, что черновик Устава проекта в большинстве случаев будет разрабатывать руководитель проекта, а согласовывать его будут спонсор и руководитель проекта.

Устав проекта не является юридически значимым документом, но может прекрасно дополнять договор или служить внутренним документом, которым будут пользоваться команда проекта и заказчик проекта в случае отклонения от первоначально согласованных параметров проекта.

WBS (ИСР – иерархическая структура работ проекта)

Для того чтобы перейти от результатов проекта к списку работ, ограничить рамки проекта и убедиться в том, что важные работы проекта не будут упущены, в некоторых методологиях управления проектами (например, PMBOK, MSF) предлагают использовать документ **Work Breakdown Structure** (WBS, Иерархическую структуру работ). WBS позволяет разделить проект на части (пакеты работ):

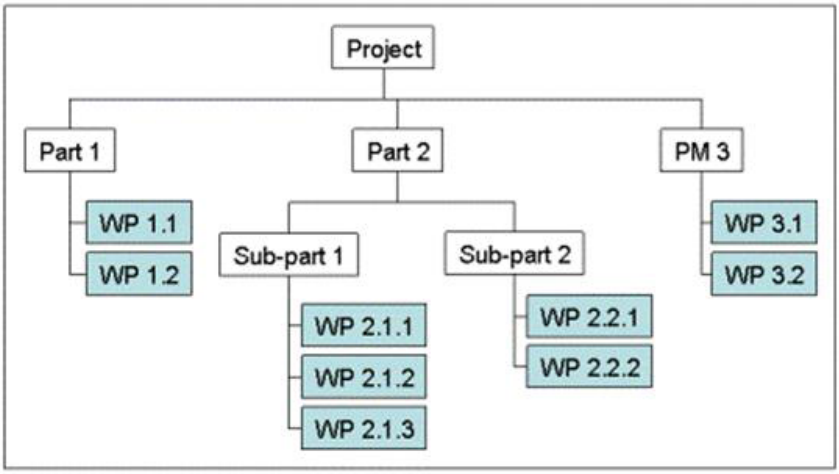


Рисунок 20

В MSF пишут о том, что WBS связывает функциональные спецификации и решения, которые необходимо создать, с задачами, которые должны быть для этого выполнены.

Существует 3 подхода к декомпозиции проекта:

1. **По результатам проекта.**
2. **По функциям (специализации труда).**
3. **По жизненному циклу**.

MSF дает следующую рекомендацию для выбора способа декомпозиции: «*Первый уровень структуры WBS может соответствовать фазам жизненного цикла проекта*».

Вот один из вариантов разбиения проекта внедрения CRM с использованием подхода по результатам проекта:

ПРОЕКТ ВНЕДРЕНИЯ CRM			
Требования к программному продукту	Программный продукт CRM, соответствующий требованиям	Нормативные документы по процессам CRM	Обученные работе в CRM пользователи

Описанные в WBS результаты проекта являются необходимыми для достижения конечной цели проекта – CRM в компании должна приносить эффект для бизнеса.

Рассмотрим второй вариант декомпозиции проекта – по этапам жизненного цикла:

ПРОЕКТ ВНЕДРЕНИЯ CRM			
Разработка требований к программному продукту	Выбор программного продукта и поставщика	Доработка программного продукта под требования	Внедрение программного продукта

Этот же проект внедрения CRM можно декомпозировать по специализации труда:

ПРОЕКТ ВНЕДРЕНИЯ CRM				
Сбор и анализ требований	Разработка архитектуры и технического проекта	Доработка программного продукта	Внедрение программного продукта	Управление проектом

Очевидно, что все три подхода, используемые для декомпозиции проекта на верхнем уровне, на нижних уровнях WBS должны привести примерно к тому же списку задач. При этом, никакие важные задачи проекта не должны быть упущены.

При создании WBS рекомендуется соблюдать следующие правила:

1. После декомпозиции родительской задачи нужно убедиться, что достижение результатов по дочерним задачам приведет к достижению результата по родительской задаче.

2. Дочерние задачи, декомпозирующие родительскую задачу, должны быть одинаковы по объему работ или срокам.
3. При переходе от одного уровня WBS к другому можно изменять подход к декомпозиции (например, на втором уровне использовался подход «от результатов», а на третьем можно использовать подход «по функциям»).
4. Использовать разные подходы к декомпозиции работ следует так, чтобы максимальная доля связей между задачами попала на нижние уровни WBS.

Теперь ответим на вопрос относительно глубины декомпозиции WBS. Во-первых, разные пакета работ WBS могут иметь разную глубину детализации.

Во-вторых, декомпозицию можно остановить в случае, когда задачи нижнего уровня WBS соответствуют следующим условиям:

- требования к задаче понятны потенциальному исполнителю и сотрудникам, участвующим в декомпозиции работ;
- на одну задачу можно назначить одного исполнителя;
- объем работ по задаче не превышает некоторого порогового значения.

К примеру, если трудоемкость задачи не превышает 40 чел.-часов, декомпозицию стоит остановить.

В MSF относительно размера задач в WBS есть следующая рекомендация для IT-проектов: «*Оценка времени исполнения каждой задачи не должна быть менее одного или более 40 дней*».

Расписание проекта

Расписание проекта нужно руководителю проекта и заказчику проекта для того, чтобы в самый ранний момент обнаружить отклонения от утвержденного базового расписания и принимать важные решения по корректирующим действиям.

Хорошее расписание проекта позволяет:

1. Понять, какая есть потребность в исполнителях на любой период времени, например, на следующий месяц, неделю или день.
2. Вносить в него расписание изменения в объеме доступного времени исполнителей, выделенных на задачи проекта, изменения в датах старта задач, изменения в списке задач.
3. Получить прогноз того, успевает ли команда реализовать проект в плановый срок.

Руководитель проекта для создания хорошего расписания проекта использует следующие подходы:

1. Разработка WBS проекта для получения полного списка работ по проекту.
2. Оценка объемов работ и назначение исполнителей на задачи проекта для определения сроков выполнения задач.
3. Метод критического пути для определения общей длительности проекта.

Разработка расписания проекта, как правило, выполняется в несколько итераций, а для того чтобы редактировать его быстро, нужно создать модель проекта.

Модель проекта – список работ по проекту с определенными связями в последовательности выполнения задач, с оценкой трудозатрат по задачам, назначенными ресурсами и оценками длительности по каждой задаче.

Модель проекта позволяет руководителю проекта определить потребность в исполнителях на любой горизонт планирования проекта, без особых усилий вносить изменения и получать прогнозную дату завершения проекта, проигрывать сценарии типа: «Что будет со сроком проекта, если в команду проекта взять еще одного программиста?», делать прогнозы по срокам завершения проекта.

Для создания и работы с моделью проекта существует специальное программное обеспечение, такое как MS Project, Oracle Primavera, Easy Projects, Clarizen и т. д.

Обычно используются следующие виды расписания проекта:

1. Сетевая диаграмма проекта.
2. Диаграмма Ганта.
3. План по вехам.
4. Календарный план проекта.

Далеее представлена одна из форм Календарного плана проекта.

Календарный план проекта

название проекта

Плановая дата начала проекта:
Плановая дата завершения проекта:

№ п/п	Наименование задачи	Дата старта задачи	Дата финиша задачи	Исполнитель	Критерии приемки результата

Реестр рисков

Реестр рисков не имеет единой общепризнанной структуры. В интернете можно найти разные шаблоны для данного документа, мы приведем один из них.

Пример Реестра рисков проекта:

Описание риска	Задача по реагированию на риск	Исполнитель	Срок	Статус риска
Недостаточно проработанные требования приведут к большому объему доработок и срыву сроков	Отобрать только критичные для запуска системы истории и сконцентрироваться на их реализации	Якубович	21.04.17	Актуален

Отчет о ходе проекта

Руководитель проекта должен информировать Заказчика о ходе проекта. Как сделать отчет о ходе проекта в такой форме, чтобы у заказчика уходило 10-15 минут на получение информации по проекту?

Есть простая форма отчета, состоящая из 3-х разделов:

1. Полученные результаты проекта:

Описание результата	Статус результата	Фактические затраты

2. Задачи в реализации:

Задача	%	Прогноз затрат

3. Проблемы (риски):

Описание проблемы (риска)	План реагирования

В некоторых ИТ-проектах заказчику предлагают использовать форму отчета на одном листе, которые включают до 7-10 областей внимания. Пример отчета размещен на следующей странице (см. рис. 21 на стр. 70).

Раздел «Цели проекта». Здесь нужно сформулировать одну или несколько целей проекта.

Раздел «Задачи проекта». Сюда нужно внести 20-25 задач из расписания проекта. Если задач в проекте больше, вносить нужно задачи второго или третьего уровня WBS.

На пересечении столбца «Цели» и строк с задачами проекта нужно указать, какие задачи приведут к достижению данной цели проекта. Когда все кружочки в столбце с названием «Цели» будут закрашены цветом – цель считается достигнутой.

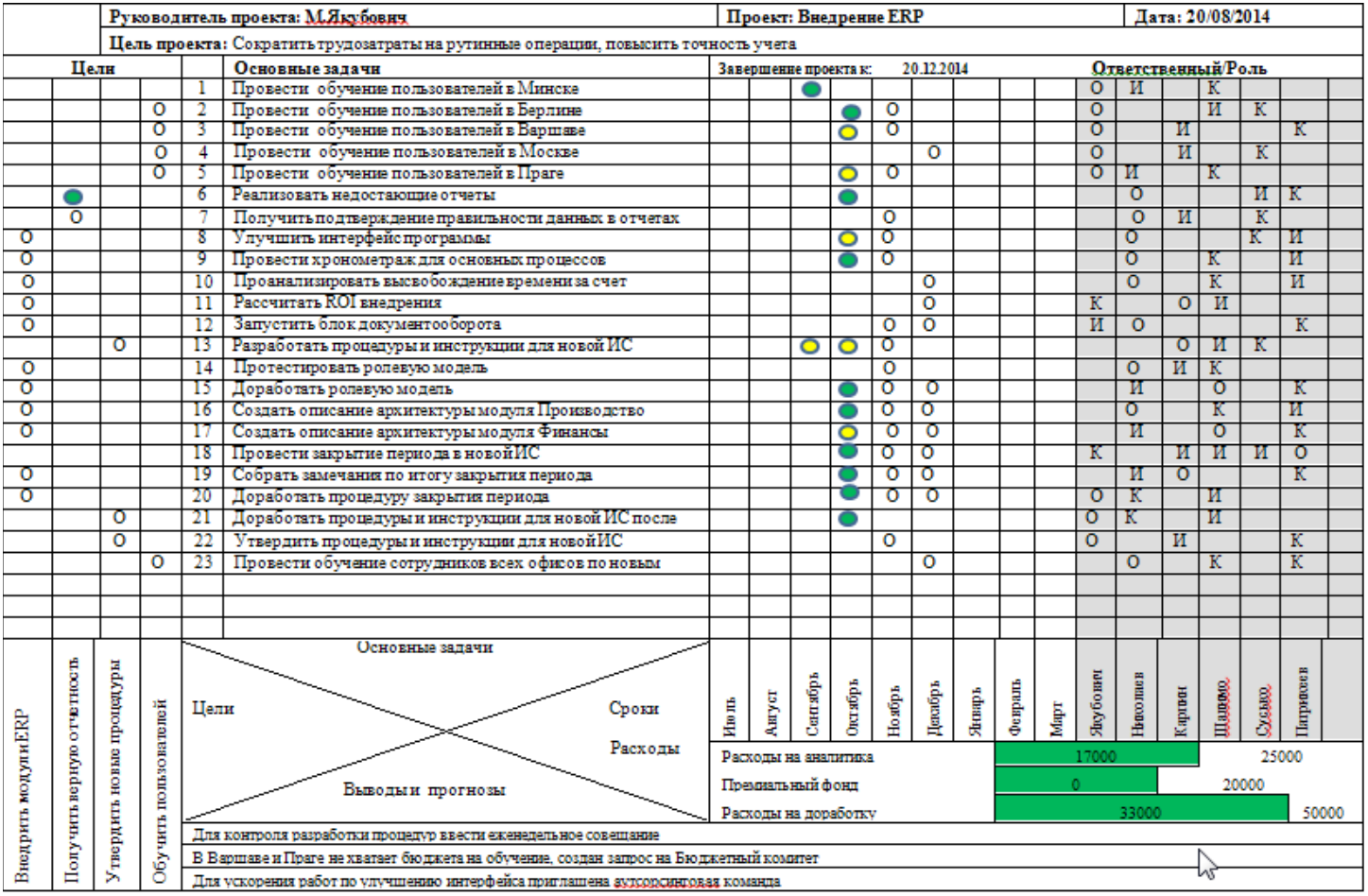

Руководитель проекта: М.Якубович
Проект: Внедрение ERP
Дата: 20/08/2014
Цель проекта: Сократить трудозатраты на рутинные операции, повысить точность учета
Цели
Основные задачи
Завершение проекта к: 20.12.2014
Ответственный/Роль
1 Провести обучение пользователей в Минске
2 Провести обучение пользователей в Берлине
3 Провести обучение пользователей в Варшаве
4 Провести обучение пользователей в Москве
5 Провести обучение пользователей в Праге
6 Реализовать недостающие отчеты
7 Получить подтверждение правильности данных в отчетах
8 Улучшить интерфейс программы
9 Провести хронометраж для основных процессов
10 Проанализировать высвобождение времени за счет
11 Рассчитать ROI внедрения
12 Запустить блок документооборота
13 Разработать процедуры и инструкции для новой ИС
14 Протестировать ролевую модель
15 Доработать ролевую модель
16 Создать описание архитектуры модуля Производство
17 Создать описание архитектуры модуля Финансы
18 Провести закрытие периода в новой ИС
19 Собрать замечания по итогу закрытия периода
20 Доработать процедуру закрытия периода
21 Доработать процедуры и инструкции для новой ИС после
22 Утвердить процедуры и инструкции для новой ИС
23 Провести обучение сотрудников всех офисов по новым
Внедрить модули ERP
Получить верную отчетность
Утвердить новые процедуры
Обучить пользователей
Основные задачи
Цели
Сроки
Расходы
Выводы и прогнозы
Июль
Август
Сентябрь
Октябрь
Ноябрь
Декабрь
Январь
Февраль
Март
Якубович
Николаев
Карпин
Шапово
Сусько
Патрикеев
Расходы на аналитика
17000
25000
Премиальный фонд
0
20000
Расходы на доработку
33000
50000
Для контроля разработки процедур ввести еженедельное совещание
В Варшаве и Праге не хватает бюджета на обучение, создан запрос на Бюджетный комитет
Для ускорения работ по улучшению интерфейса приглашена аутсорсинговая команда

Рисунок 21

Раздел «Сроки». Этот раздел позволяет визуализировать на диаграмме периоды, в которые запланированы работы по задаче. Если задачу планируется выполнять в нескольких периодах, то напротив этих периодов рисуются кружочки.

Раздел «Ответственный/Роль». Внизу указываются фамилии ключевых участников проекта, а выше фамилий заполняется матрица ответственности. В данной форме используются следующие значения:

- Ответственный (буква **О**) – с него спросят за получение результата по задаче;
- Исполнитель (**И**) – непосредственно исполняет задачу;
- Контролер (**К**) – принимает результаты задачи и формулирует список ошибок.

Раздел «Расходы». Вносятся плановые статьи бюджета проекта и плановый размер расходов по ним. Чем больше плановый бюджет, тем длиннее отрезок.

Раздел «Выводы и прогнозы». Руководителю проекта нужно убедить заказчика, что он знает, как «вырулить» проект.

Как заполняется отчет в части выполнения задач?

Если объем фактически выполненной работы был не меньше запланированного, то в отчете в строке для задачи кружок отмечается зеленым цветом, если фактический объем оказался меньше запланированного, но есть шанс наверстать расписание – используется желтый цвет, а если сроки по задаче ничего не делалось – используется красный цвет.

Та же система «светофоров» используется и для закрашивания отрезков для статей бюджета. Если бюджет осваивается по плану – зеленый цвет, в отчетном периоде был перерасход – используется желтый, плановый размер по статье бюджета превышен – красим в красный.

Данная форма отчета является очень удобной и нравится заказчикам проектов.

Product Backlog

Данный документ используется в подходе Scrum для сбора и управления очередностью всех требований к программному продукту. Ответственным за то, какие требования попадут в Product Backlog и за наличие приоритетов у требований обычно является Product Owner (*подробнее о роли PO смотри в уроке по Scrum*).

Структура Product Backlog не является строгой и не описана в Scrum Guide. Ниже приведен один из вариантов структуры данного документа:

Требование	Важность	Сценарий тестирования	Оценка

Требование – описание Epic или User story (*см. урок про Scrum*).

Важность – это значение заполняет Product owner, он же определяет шкалу, по которой измеряется важность требования. В литературе встречается рекомендация относительно длинны списка требований, которая является адекватной для эффективной работы со списком – около 100 штук. Исходя из этого, PO может выбрать шкалу от 100 (максимальная важность) до 1 (минимальная важность).

Сценарий тестирования – в этом поле описывается сценарий, при воспроизведении которого требование будет работать так, как ожидает Product owner.

Оценка – в этот столбец вносятся значения оценки сложности реализации требований. Оценки делает, как правило, один из участников команды проекта и они носят предварительный характер, это значит, что они могут уточняться при планировании спринта.

Vision and Scope (Документ об образе и границах проекта)

Данный документ позволяет понять ответы на вопросы:

- Какие проблемы или возможности решает продукт?
- Каковы ключевые бизнес-требования к продукту?
- Кто будет пользователями продукта?
- Какие бизнес-требования попадут в какой релиз продукта?

Структуру данного документа можно позаимствовать в книге **К. Вигерса «Разработка требований к программному обеспечению»**:

1. Бизнес-требования
1.1. Исходные данные, возможности бизнеса, нужды клиентов
1.2. Бизнес-цели и критерии успеха
1.3. Факторы бизнес-рисков
2. Образ решения
2.1. Положение об образе проекта

2.2. Основные функции (характеристики)
2.3. Предположения и зависимости
3. Масштабы и ограничения
3.1. Объем первого и последующих выпусков
3.2. Ограничения и исключения
4. Бизнес-контекст
4.1. Профили заинтересованных в проекте лиц (участников)
4.2. Приоритеты проекта

Заметим, что некоторые разделы этого документа пересекаются с документом Устав проекта, поэтому руководителю проекта стоит адаптировать один из документов, чтобы избежать дублирования информации.

Software Requirements Specification SRS

Этот документ должен содержать все требования к программному продукту. Классификация требований к программному продукту будет рассмотрена в следующем уроке.

Не существует единой версии структуры документа SRS, но есть два шаблона, которые используются довольно часто.

Структуру данного документа можно позаимствовать в книге **К. Вигерса «Разработка требований к программному обеспечению»**:

1. Введение
1.1. Назначение
1.2. Объем проекта и функции продукта

1.3. Ссылки
2. Общее описание
2.1. Общий взгляд на продукт
2.2. Классы и характеристики пользователей
2.3. Операционная среда
2.4. Ограничения дизайна и реализации
2.5. Документация для пользователей
2.6. Предположения и зависимости
3. Функции системы
3.1. Функция 1
3.2. Функция 2
3.3. Функция 3
3.4. …..
3.5. Функция N
4. Требования к внешнему интерфейсу
4.1. Интерфейсы пользователей
4.2. Интерфейсы оборудования
4.3. Программные интерфейсы
4.4. Интерфейсы передачи данных
5. Другие нефункциональные требования
5.1. Требования к производительности
5.2. Требования к охране труда
5.3. Требования к безопасности
5.4. Атрибуты качества ПО
Приложение A: Словарь и модель данных
Приложение B: Модели анализа

Второй шаблон SRS был предложен в некогда популярном для ИТ-проектов подходе – RUP и имеет следующую структуру:

1. Introduction
1.1. Purpose
1.2. Scope
1.3. Definitions, Acronyms, and Abbreviations
1.4. References
1.5. Overview
2. Overall Description
3. Specific Requirements
3.1. Functionality
3.1.1. <Functional Requirement One>
3.2. Usability
3.2.1. <Usability Requirement One>
3.3. Reliability
3.3.1. <Reliability Requirement One>
3.4. Performance
3.4.1. <Performance Requirement One>
3.5. Supportability
3.5.1. <Supportability Requirement One>
3.6. Design Constraints
3.6.1. <Design Constraint One>
3.7. On-line User Documentation and Help System Requirements
3.8. Purchased Components
3.9. Interfaces

3.9.1. User Interfaces
3.9.2. Hardware Interfaces
3.9.3. Software Interfaces
3.9.4. Communications Interfaces
3.10. Licensing Requirements
3.11. Legal, Copyright, and Other Notices
3.12. Applicable Standards
4. Supporting Information

А вот структура SRS, рекомендуемая стандартом IEEE 830:

- Введение
  - Цели
  - Соглашения о терминах
  - Предполагаемая аудитория и последовательность восприятия
  - Масштаб проекта
  - Ссылки на источники
- Общее описание
  - Видение продукта
  - Функциональность продукта
  - Классы и характеристики пользователей
  - Среда функционирования продукта (операционная среда)
  - Рамки, ограничения, правила и стандарты
  - Документация для пользователей
  - Допущения и зависимости

- Функциональность системы
  - Функциональный блок X (таких блоков может быть несколько)
    - Описание и приоритет
    - Причинно-следственные связи, алгоритмы (движение процессов, workflows)
    - Функциональные требования
- Требования к внешним интерфейсам
  - Интерфейсы пользователя (UX)
  - Программные интерфейсы
  - Интерфейсы оборудования
  - Интерфейсы связи и коммуникации
- Нефункциональные требования
  - Требования к производительности
  - Требования к сохранности (данных)
  - Критерии качества программного обеспечения
  - Требования к безопасности системы

Пользовательская документация

В большинстве случаев для пользователей программного продукта необходимо иметь следующие виды документации для работы с программой:

- встроенная справка по работе с программой;
- материалы с описанием кейсов по использованию продукта (чаще всего их создают в формате видеоуроков, но встречаются еще и описания в виде документов).