

4 УПРАВЛЕНИЕ КОМАНДОЙ ПРОЕКТА

Успех проекта напрямую связан с используемыми талантами, и, что более важно, способом, в соответствии с которым руководство использует эти таланты в проекте.

Джон Макдоналд

В спиральной модели удалось более четко и естественно определить контрольные точки проекта, в определенной степени, подчеркнув эволюционную природу жизненного цикла. Теперь же пора взглянуть на жизненный цикл в контексте методологий, не просто детализирующих ту или иную модель, но добавляющих к ним *ключевой элемент - людей*. *Роли*, как представление различных функциональных групп работ, связывает создание, модификацию и использование активов проектов с конкретными участниками проектных команд. В совокупности с процессами и активами (артефактами) они позволяют нам создать целостную и подробную картину жизненного цикла (рисунок 4.1).



Рисунок 4.1 – Модель производственной архитектуры

Управление программным проектом включает решение трех основных задач:

1. Подбор и управление командой (персонала).
2. Выбор процесса.
3. Выбор инструментальных средств.

Хотя все три задачи одинаково важны для успеха проекта, ведущую роль играет правильный подбор и управление командой. Успех проекта во многом зависит от того, насколько состав участников проекта сможет быть преобразован в команду единомышленников, насколько эта команда будет активной и инициативной с одной стороны и управляемой с другой. Из множества вопросов управления командой проекта мы рассмотрим три:

- Ролевая модель команды;
- Модели организации команд;
- Общение в команде.

4.1 Ролевая модель команды

Состав команды определяется опытом и уровнем коллектива, особенностями проекта, применяемыми технологиями и уровнем этих технологий. На рисунке 4.1 представлен один из вариантов состава команды, описанный в [34]. Выделенные позиции не обязательно представлены конкретными людьми. Это список основных функциональных ролей в команде (ролевая модель команды). В малых командах роли могут совмещаться. В больших – выделяться группы или отделы (отдел проектирования, отдел тестирования, отдел контроля качества, отдел подготовки документации, ...).

Состав команды определяется также типом выполняемых работ: под заказ или коробочное производство (продукт на рынок). *Инженерный психолог и инженер по маркетингу* нужны в последнем случае.

В представленной модели выделены следующие основные роли (рисунок 4.2):

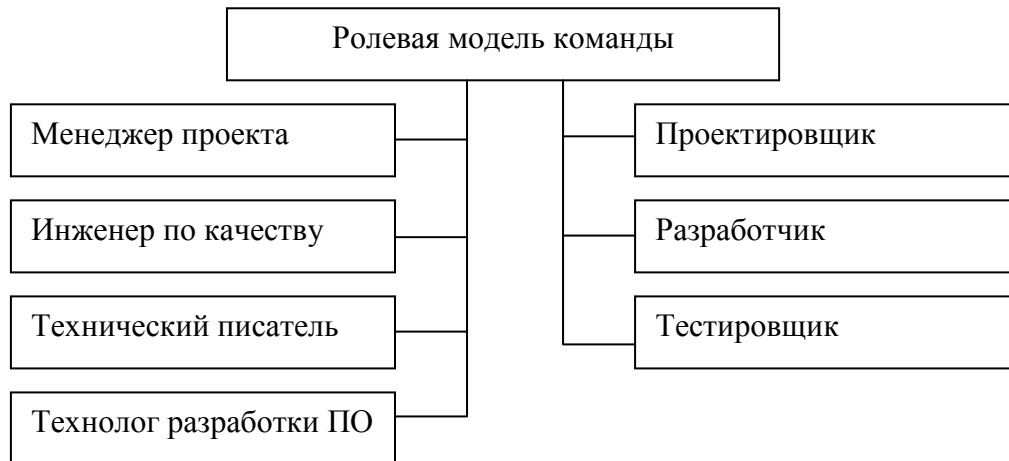


Рисунок 4.1 – Ролевая модель команды

- **Менеджер проекта** - главное действующее лицо, обладающее знаниями и навыками, необходимыми для успешного управления проектом. Его основные функции:
 - ✓ Подбор и управление кадрами;
 - ✓ Подготовка и исполнение плана проекта;
 - ✓ Руководство командой;
 - ✓ Обеспечение связи между подразделениями;
 - ✓ Обеспечение готовности продукта.
- **Проектировщик** - это функция проектирования архитектуры высокого уровня и контроля ее выполнения. В небольших командах функция распределяется между менеджером и разработчиками. В больших проектах это может быть целый отдел. Основными функциями проектирования являются:

- ✓ Анализ требований.
 - ✓ Разработка архитектуры и основных интерфейсов.
 - ✓ Участие в планировании проекта.
 - ✓ Контроль выполнения проекта.
 - ✓ Участие в подборе кадров.
- **Разработчик** – роль, ответственная за непосредственное создание конечного продукта. Помимо собственно программирования (кодирования) в его функции входит:
- ✓ Контроль архитектурных и технических спецификаций продукта.
 - ✓ Подбор технологических инструментов и стандартов.
 - ✓ Диагностика и разрешение всех технических проблем.
 - ✓ Контроль за работой разработчиков документации, тестирования, технологов.
 - ✓ Мониторинг состояния продукта (ведение списка обнаруженных ошибок).
 - ✓ Подбор инструментов разработки, метрик и стандартов. Контроль их использования.
- **Тестировщик** – роль, ответственная за удовлетворение требований к продукту (функциональных и нефункциональных). В функции тестировщика входит:
- ✓ Составление плана тестирования. План тестирования составляет один из элементов проекта и составляется до начала реализации (разработки) проекта. Время, отводимое в плане на тестирование может быть сопоставимо с временем разработки.
 - ✓ Контроль выполнения плана. Важнейшая функция контроля – поддержка целостности базы данных зарегистрированных ошибок. В этой базе регистрируется:
 - кто, когда и где обнаружил, описание ошибки, описание состояния среды;
 - статус ошибки: приоритет, кто разрешает
 - состояние ошибки: висит, в разработке, разрешена, проблемы
- Эта база должна быть доступна всем, т.к. в тестировании принимают участие все члены команды.
- ✓ Разработка тестов. Самая трудоемкая часть в работе тестировщика. Тестирование должно обеспечить полную проверку функциональности при всех режимах работы продукта.

- ✓ Автоматизация тестирования включает автоматизацию составления тестов, автоматизацию пропуска тестов и автоматизацию обработки результатов тестирования. В виду важности автоматизации тестирования, иногда вводят нового участника – инженера по автоматизации.
 - ✓ Выбор инструментов, метрик, стандартов для организации процесса тестирования.
 - ✓ Организация Бета тестирования - тестирования почти готового продукта внешними тестерами (пользователями). Эту важную процедуру надо продумать и организовать в случае разработки коробочного продукта.
- **Инженер по качеству.** В современном представлении рассматривается три аспекта (уровня) качества:
- ✓ –качество конечного продукта – обеспечивается тестированием,
 - ✓ –качество процесса разработки (тезис: для повышения качества продукта надо повысит качество процесса разработки),
 - ✓ –качество (уровень) организации (тезис: для повышения качества процесса надо повысить качество организации работ).

В некоторых случаях функции инженера по качеству возлагаются на тестировщика. На самом деле они шире – два следующих уровня качества. Здесь приведены функции, отличные от функции тестировщика:

- ✓ Составление плана качества. План качества включает все мероприятия по повышению качества (на всех уровнях). Имеет долговременный характер. План тестирования – его оперативная составляющая.
- ✓ Описание процессов. Описание процессов является их формализацией. При описании вводятся метрики процесса, влияющие на качество продукта.
- ✓ Оценка процессов включает регистрацию хода выполнения процессов и оценку значений установленных метрик процессов. Выявление «слабых» мест и выработка рекомендаций по улучшению процессов.
- ✓ Улучшение процессов - переопределение процесса, автоматизация части работ, обучение персонала.

Повышение качества процессов требует участия всех действующих лиц. Принятое решение должно быть обосновано, всем понятно и всеми принято. При повышении качества организации работа по улучшению процессов проводится по определенной схеме. На каждом шаге повышения уровня организации работ выделяются ключевые процессы и выполняются работы по улучшению этих процессов.

- **Технический писатель** или разработчик пользовательской (и иной) документации как части программного продукта. Функциями технического писателя являются:
 - ✓ Разработка плана документирования, который включает состав, сроки подготовки и порядок тестирования документов.
 - ✓ Выбор и разработка стандартов и шаблонов подготовки документов
 - ✓ Выбор средств автоматизации документирования
 - ✓ Разработка документации
 - ✓ Организация тестирования документации. Участие в тестировании продукта. Технический писатель все время работает с продуктом (его готовыми версиями) и выступая от имени пользователя видит все недочеты и несоответствия.
- **Технолог разработки ПО** обеспечивает выполнение следующих задач:
 - ✓ Поддержка модели ЖЦ - создание служб и структур по поддержке работоспособности принятой модели ЖЦ ПО. В поддержке модели ЖЦ принимают участие все. Но контроль возложен на технолога.
 - ✓ Создание и сопровождение среды сборки продукта. Функция особенно важна на завершающих этапах разработки или при использовании модели прототипирования. В такой ситуации сборка будет проводиться достаточно часто (в некоторых случаях - ежедневно). Среда сборки должна быть подготовлена заранее, сборка должна проводиться быстро и без сбоев. С учетом сборки версий это не простая задача.
 - ✓ Создание и сопровождение процедуры установки с тем, чтобы каждая сборка устанавливалась автоматически с учетом версии и конфигураций сред.
 - ✓ Управление исходными текстами - сопровождение и администрирование системы управления версиями исходных текстов.

Подводя итог, следует отметить, что ролевые модели проектных команд могут быть самыми разнообразными.

4.2 Модели организации команд

«...методологи разрабатывают сложные системы, в которых есть весьма изменчивые и нелинейные компоненты – люди»

Алистэр Коуберн

Как организовать работу команды? Команды из 10 человек и команды из 500 человек? Есть ли различия и в чем они состоят? Надо ли организовывать работу по жесткой технологии или надо предоставить свободу действий? Можно ли найти методологию (технологию) выполнения проекта, обеспечивающую успех?

Алистер Коубен [35] – специалист в области технологий выполнения ИТ проектов - приводит данные 23 проектов различной степени сложности, выполнявшихся по различным технологиям и имеющие различные результаты. Пытаясь проанализировать результаты применения различных технологий в тех или иных условиях, он приходит к выводу:

- Практически любую методологию можно с успехом применять в каком-нибудь проекте.
- Любая методология может привести к провалу проекта.

Главную причину он видит в том, прямо перед нами всегда находится нечто, чего мы не замечаем: люди. Именно человеческие качества обеспечивают успех тому или иному проекту, именно они являются фактором первостепенной важности, основываясь на котором надо строить прогнозы о проекте.

Исследованию вопросов человеческого фактора (Peopleware) уделяется достаточно много внимания, наиболее известными работами являются [36-39].

Проблемы человеческого фактора связаны с тем (проявляются в том), что участвующие в проекте **люди**:

- *Все разные* – по характеру, темпераменту, активности, целям – нет двух одинаковых людей.
- *Все похожие* – участие в проекте объединяет людей общностью целей, поиском путей достижения этих целей.
- *Различаются по типу*:
 - ✓ индивидуалисты - члены команды;
 - ✓ генераторы идей - исполнители;
 - ✓ ответственные – безответственные;

- *Постоянны и изменчивы* – люди, как правило, проявляют постоянство своих привычек и свойств характера, но при этом способны проявлять «противоположные» качества: индивидуалист – командные качества, исполнитель – генерировать идеи,
- *Многообразны* – надо понимать, что многообразие людей является основной гарантией выживания человечества вообще и возможности выполнять ИТ проекты в частности. Если бы все были индивидуалисты или все командники, все генераторы идей или все исполнители, то вряд ли удалось выполнить хотя бы один проект, а мир стал бы ужасен.

Как же управлять такими людьми? Рассмотрим основные модели проектных групп, которые применяются на практике

4.2.1 Административная модель (теория X)

Это традиционный стиль управления, связанный с иерархической административно-командной моделью, которую используют военные организации. В основе лежит *теория X*, которая утверждает, что такой подход необходим, поскольку большинство людей по своей природе не любит работу и будет стремиться избежать ее, если у них есть такая возможность. Однако менеджеры должны принуждать, контролировать, направлять сотрудников и угрожать им, чтобы получить от них максимальную отдачу. *Девиз теории и модели: Люди делают только то, что вы контролируете.* Или в более мягком варианте: Люди делают то, что они не хотят делать, только если вы их контролируете. В конце концов, теория утверждает, что большинство людей предпочитают, чтобы им говорили, что следует делать и им не придется ничего решать самим.

Характерные черты модели:

- Властная пирамида – решения принимаются сверху-вниз;
- Четкое распределение ролей и обязанностей;
- Четкое распределение ответственности;
- Следование инструкциям, процедурам, технологиям;
- Роль менеджера: планирование, контроль, принятие основных решений.

Преимущества модели: ясность, простота, прогнозируемость. Модель хорошо сочетается с каскадной моделью жизненного цикла и применима в тех же случаях, что и каскадная модель. Модель эффективна в случае установившегося процесса.

Недостатки модели связаны с тем, что административная система стремится самосохранению (стабильности) и плохо восприимчива к изменению ситуации – новые типы проектов, применение новых технологий, оперативная реакция на изменение рынка.

Кроме того, в административной модели плохо уживаются индивидуалисты и генераторы идей.

Административная система (модель) – это тяжелый паровоз, идущий в «середине» и не поддающийся на «крайности» поиска новых путей и решений. Она воспринимает новые решения и технологии, но только проверенные, отработанные и стандартизированные. В этом ее сила, слабость и проявление принципа многообразия. Видимо, именно к ней в наибольшей степени применим термин «промышленное программирование».

4.2.2 Модель хаоса (теория Y)

В основе модели хаоса лежит *Теория Y*, которая является полной противоположностью Теории X. *Основной тезис: Работа — естественная и приятная деятельность и большинство людей*, на самом деле, очень ответственные и не уваливают от работы.

Характерными чертами модели хаоса являются:

- Отсутствие явно выраженных признаков власти;
- Роль менеджера – поставить задачу, обеспечить ресурсами, не мешать и следить, чтобы не мешали другие;
- Отсутствие инструкций и регламентированных процедур;
- Индивидуальная инициатива - решения по проблеме принимаются там, где проблема обнаружена;
- Процесс напоминает творческую игру участников на основе дружеской соревновательности.

Преимущества такой модели в том, что творческая инициатива участников ничем не связана и потенциал участников раскрывается в полной мере. Это бывает особенно эффективно в случае, когда для решения проблемы требуется поиск новых подходов, методов, идей и средств. Команда становится командой «прорыва», а работа проходит в форме игры, цель которой – поиск наилучшего результата. Процесс напоминает случайный поиск, когда идеи и решения рождаются при живом и как бы случайном обсуждении проблем в коридоре, столовой на пикнике. Собрать такую команду в рабочей комнате и устроить обсуждение по регламенту часто просто не удастся – это команда творческих индивидуалистов.

Недостатки модели связаны с тем, что при определенных условиях команда прорыва может стать командой провала. Причинами провала могут быть:

- Творческая соревновательность переходит в конкуренцию сначала идей, а потом - личностей.

- Процесс начинает преобладать над целью проекта – высказанные идеи не доводятся до конца и сменяются новыми идеями, преобладание получают «красивые» идеи, лежащие в стороне от основных целей проекта.
- Люди, способные к генерации идей, редко обладают терпением доведения идей до полной реализации.

Модель хаоса – это то, что нужно для освоения новых земель. Модель хаоса не противоречит административной модели – она ее дополняет и может эффективно с ней соседствовать (но в разных комнатах!). Многие мускулистые корпоративные бегемоты полагаются на исследовательские «отделы скунсов», откуда они черпают новые идеи, технологии и продукты.

4.2.3. Открытая архитектура (теория Z)

Административная и хаотическая модели являются двумя «крайностями», между которыми находятся множество моделей, сочетающих преимущества «крайних» моделей. Одной из таких моделей является модель открытой архитектуры, основанная на Теории Z. Эта теория была сформулирована Уильямом Оучи на основе изучения опыта японского стиля управления (Theory Z: How American Business Can Meet the Japanese Challenge,» Perseus Publishing, 1981). Теория Z предполагает (но не декларирует) наличие внутреннего механизма управления, основанного на влиянии со стороны коллег и группы в целом. Дополнительное воздействие оказывают культурные нормы конкретной корпорации. *Основной принцип* модели можно сформулировать так: *«Работаем спокойно. Работаем вместе»*.

Особенностями этой модели являются:

- Адаптация к условиям работы – если делаем независимые модули, то расходимся и делаем, если нужна архитектура базы данных, то собираемся вместе и обсуждаем идеи.
- Коллективное обсуждение проблем, выработка консенсуса и принятие решения – не все могут согласиться, но принятое решение является коллективным и в силу этого – обязательным для всех.
- Распределенная ответственность – отвечают все, кто обсуждал, вырабатывал, принимал.
- Динамика состава рабочих групп в зависимости от текущих задач.
- Отсутствие специализации – участники меняются ролями и функциями и могут при необходимости заменить друг друга.

- Задача менеджера – активное (но рядовое, не руководящее) участие в процессе, контроль конструктивности обсуждений, обеспечение возможности активного участия всех.

Открытая архитектура является более гибкой, адаптируемой, настраиваемой на ситуацию. Она дает возможность проявить себя всем членам команды – в ней могут уживаться и индивидуалисты и коллективисты. Коллективное обсуждение высказанных идей позволяет оставлять только прагматичные идеи.

4.2.4 Модель проектной группы MSF for Agile Software Development

Основные принципы построения команды

Рассмотрим, наверное, самую главную из отличительных черт MSF в сравнении с другими методологиями - модель проектной группы и *принципы*, положенные в основу построения команды.

Методология MSF считает, что успешная работа команды над проектом существенным образом зависит от ее структуры и распределения зон ответственности *ролевых групп* (более подробно о составе проектной группы далее) внутри команды. Построение команды в MSF соответствует ряду ключевых концепций (key concepts), часть которых кажутся самоочевидными (первые три), другие чем-то сродни «ноу-хау» (последние две):

- *Концентрация на нуждах заказчика* (customer-focused mindset) - главный приоритет любой хорошо работающей проектной группы. Означает обязательное понимание бизнес-задач заказчика и стремление к их решению со стороны команды. Не менее важным является активное участие заказчика в проектировании решения и получение его отзывов в ходе процесса разработки.
- *Нацеленность на конечный результат* (product mindset) - каждый участник проектной группы должен рассматривать собственную работу в качестве самостоятельного проекта или же вклада в какой-либо больший проект. Установка на конечный продукт означает, что получению конечного результата проекта уделяется больше внимания, чем процессу его достижения. Из этого не следует, что сам процесс может быть плох или непродуман - просто он существует для получения конечной цели, а не ради себя самого.
- *Установка на отсутствие дефектов* (zero-defect mindset) - это стремление к высочайшему уровню качества. Она означает, что цель команды - выполнение своей работы с максимально возможным качеством, в идеале таким образом, что если от команды потребуют поставить результат завтра, она будет способна

поставить что-то работающее. В успешной команде каждый сотрудник чувствует ответственность за качество продукта. Она не может быть делегирована одним членом команды другому или же от одной ролевой группы другой.

- «*Проектная группа - команда равных*» (*team of peers*). Концепция означает равноправное положение каждой из ролей в команде. Чтобы достичь успеха в рамках команды равных, каждый из ее членов, независимо от роли, должен нести ответственность за качество продукта, понимать интересы заказчика и сущность решаемой бизнес-задачи. В то же время, принятие решения методом консенсуса между ролями не тождественно принятию решения методом консенсуса между сотрудниками. Каждая ролевая группа требует определенной организационной иерархии для распределения работы и управления ее ресурсами.
- *Стремление к самосовершенствованию* (*willingness to learn*) - это приверженность идее неустанного саморазвития посредством накопления опыта и обмена знаниями. Оно позволяет членам проектной группы извлекать пользу из отрицательного опыта сделанных ошибок, равно как и воспроизводить успехи, используя проверенные методы работы других людей. По окончании основных фаз проекта и по завершению проекта в целом предполагается проведение открытых обсуждений его состояния и доброжелательный, но объективный анализ.

К концепции команды равных в MSF тесно примыкает идея о том, что каждая ролевая группа имеет зону ответственности и защищает интересы заинтересованных лиц из этой зоны (более подробно об этом далее).

Модель проектной группы в MSF может масштабироваться в зависимости от числа участников.

Ролевые группы и роли

Методология MSF основана на постулате о качественных целях, достижение которых определяет успешность проекта. Эти цели обуславливают модель проектной группы. В то время как за успех проекта ответственна вся команда, каждая из ее ролевых групп, определяемых моделью, ассоциирована с одной из целей и работает над ее достижением.

MSF for Agile Software Development выделяет **7 ролевых групп** (см. рисунок 4.2):

- Управление программой (*program management*);
- Управление продуктом (*product management*);
- Управление выпуском (*release operations*);

- Архитектура продукта (*architecture*);
- Разработка (*development*);
- Тестирование (*test*);
- Удовлетворение потребителя (*user experience*);

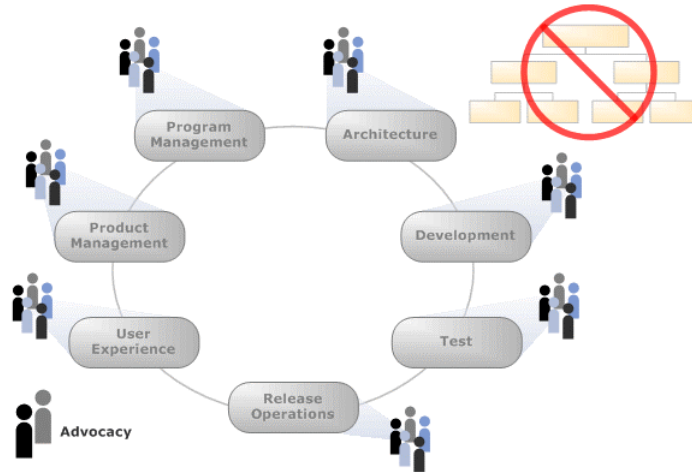


Рисунок 4.2 - Модель команды в MSF 4.0 – ролевые группы [36]

и **6 ролей** (см. рисунок 4.3):

- **менеджер проекта** (*project manager*) – ролевая группа «Управление программой», который отвечает за соблюдение ограничений проекта, его основная задача – вести процесс разработки таким образом, чтобы нужный продукт был выпущен в нужное время, он координирует деятельность всех членов группы, он должен полностью понимать все модели и процессы, повышающие эффективность труда;

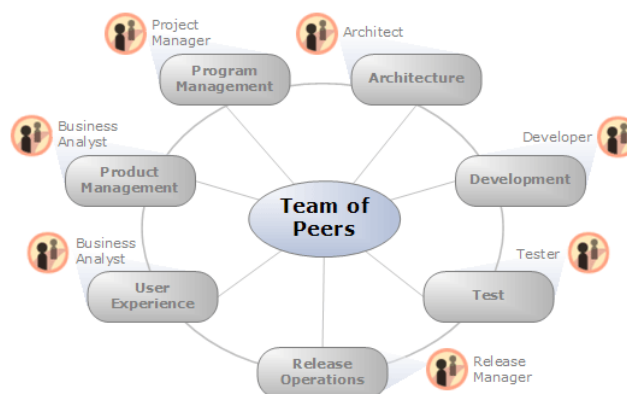


Рисунок 4.3 - Модель команды в MSF 4.0 – роли

- **бизнес-аналитик** (*business analyst*) или **менеджер продукта** – ролевые группы «Управление продуктом» и «Удовлетворение потребителя», который отвечает за удовлетворение требований заказчика. Для этого менеджер продукта выступает представителем заказчика в группе разработчиков и представителем команды у заказчика. Менеджер продукта совместно с менеджером проекта должны прийти к компромиссному решению относительно функциональных возможностей продукта, сроков его разработки и финансирования проекта, так называемый *треугольник ограничений проекта* (рисунок 4.4). Существует закон Лермана: «Любую техническую проблему можно преодолеть, имея достаточно времени и денег» (Следствие Лермана: «Вам никогда не будет хватать либо времени, либо денег»). Для того, чтобы решить проблему свойственной IT-проектам неопределенности и рискованности, одним из ключевых факторов их успеха являются эффективные компромиссные решения (*trade-offs*). Хорошо известна взаимозависимость между ресурсами проекта (людскими и финансовыми), его календарным графиком (временем) и реализуемыми возможностями (рамками). Эти три переменные образуют треугольник компромиссов (*tradeoff triangle*), приведенный на рисунке 3.17.

После достижения равновесия в этом треугольнике изменение на любой из его сторон для поддержания баланса требует модификаций на другой (двух других) сторонах и/или на изначально измененной стороне. Нахождение верного баланса между ресурсами, временем разработки и возможностями – ключевой момент в построении решения, должным образом отвечающего нуждам заказчика;



Рисунок 4.4 - Треугольник ограничений проекта

- **релиз-менеджер** (*release manager*) или **логистик** – ролевая группа «Управление выпуском», который отвечает за простоту развертывания и постоянное сопровождение. Логистик обязан разбираться в инфраструктуре продукта и требованиях к его сопровождению, кроме этого он должен уметь координировать установку программного обеспечения и оценить ее результаты. От него требуется хорошая техническая подготовка, он должен уметь устанавливать и настраивать

пользовательские системы. Если проект крупный, то логистик должен иметь опыт развертывания крупномасштабных приложений на нескольких десятках или сотнях компьютеров. Одна из задач логистика – консультация группы сопровождения и пользователей после развертывания системы;

- **архитектор** (*archrect*) – ролевая группа «Архитектура», который отвечает за разработку дизайна проекта (продукта). Архитектор на самом деле входит в группу разработчиков, но на нем лежит ответственность за принятие проектных решений;
- **разработчик** (*developer*) – ролевая группа «Разработка», отвечает за реализацию системы и разрабатывает комплект проектной документации. Как программисты разработчики отвечают за низкоуровневое проектирование и оценку затрат на реализацию продукта. Разработчики, как правило, сами оценивают сроки выполнения своей работы. Такая концепция MSF – создание графиков ответственного за выполнение конкретного участка членами команды – называется *составлением расписания «снизу-вверх»* (рисунок 4.6). Такой подход повышает точность оценок (т.к. они основаны на конкретном опыте разработчика) и повышает ответственность исполнителя (т.к. план работ одновременно является обязательством члена команды). Планирование работ – задача всей проектной группы, но в основании пирамиды лежит работа разработчиков. Менеджер программы добавляет в общий план работ некоторый резерв времени, чтобы гарантировать соблюдение сроков выполнения работ даже при возникновении непредвиденных обстоятельств.

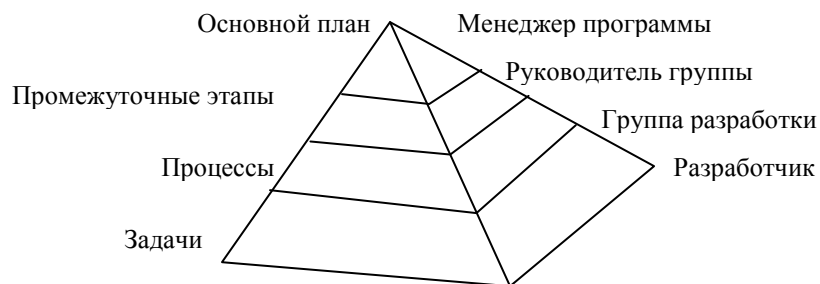


Рисунок 4.6 – Составление графика «снизу-вверх»

- **тестер** (*tester*) – ролевая группа «Тестирование», отвечает за выявление и устранение проблем (дефектов) не только на уровне кода продукта, но и его спецификации и документацию к нему. При этом испытание продукта происходит в реальных условиях.

4.2.4.3 Рекомендации по возможному объединению ролей

Модель проектной группы в MSF может масштабироваться в зависимости от числа участников. Масштабирование модели проектной группы MSF for Agile Software Development на случай больших команд выходит за рамки нашего курса. Мы рассмотрим ситуацию, когда один человек может выполнять в проектной группе несколько ролей (небольшая команда, относительно несложный проект). В этом случае MSF предлагает рекомендации по возможному объединению ролей (таблица 4.1).

Таблица 4.1 – Возможное объединение ролей

	Архитектура продукта	Управление продуктом	Управление программой	Разработка	Тестирование	Удовлетворение потребителя	Управление выпуском
Архитектура продукта		Нет	Да	Да	Не желательно	Не желательно	Не желательно
Управление продуктом	Нет		Нет	Нет	Да	Да	Не желательно
Управление программой	Да	Нет		Нет	Не желательно	Не желательно	Да
Разработка	Да	Нет	Нет		Нет	Нет	Нет
Тестирование	Не желательно	Да	Не желательно	Нет		Да	Да
Удовлетворение потребителя	Не желательно	Да	Не желательно	Нет	Да		Не желательно
Управление выпуском	Не желательно	Не желательно	Да	Нет	Да	Не желательно	

4.2.4.4 Координация работы с внешними группами

Проектная группа, рассчитывающая на успех, должна взаимодействовать с внешними группами – как с заказчиками, так и с пользователями, так и с другими разработчиками [40]. Этим занимаются менеджер программы, менеджер продукта (бизнес-аналитик) и релиз-менеджер (логистик). В обязанность этих ролей входят как внутренние, так и внешние контакты, в то время, как архитекторы, разработчики и тестеры изолированы от общения с внешним миром (такая изоляция приводит к повышению эффективности труда этих двух групп). Важно, чтобы взаимодействие этих групп было понятным и явным. На рисунке 4.7 проиллюстрирована координация взаимодействий, связанная как с бизнесом, так и с технологиями.

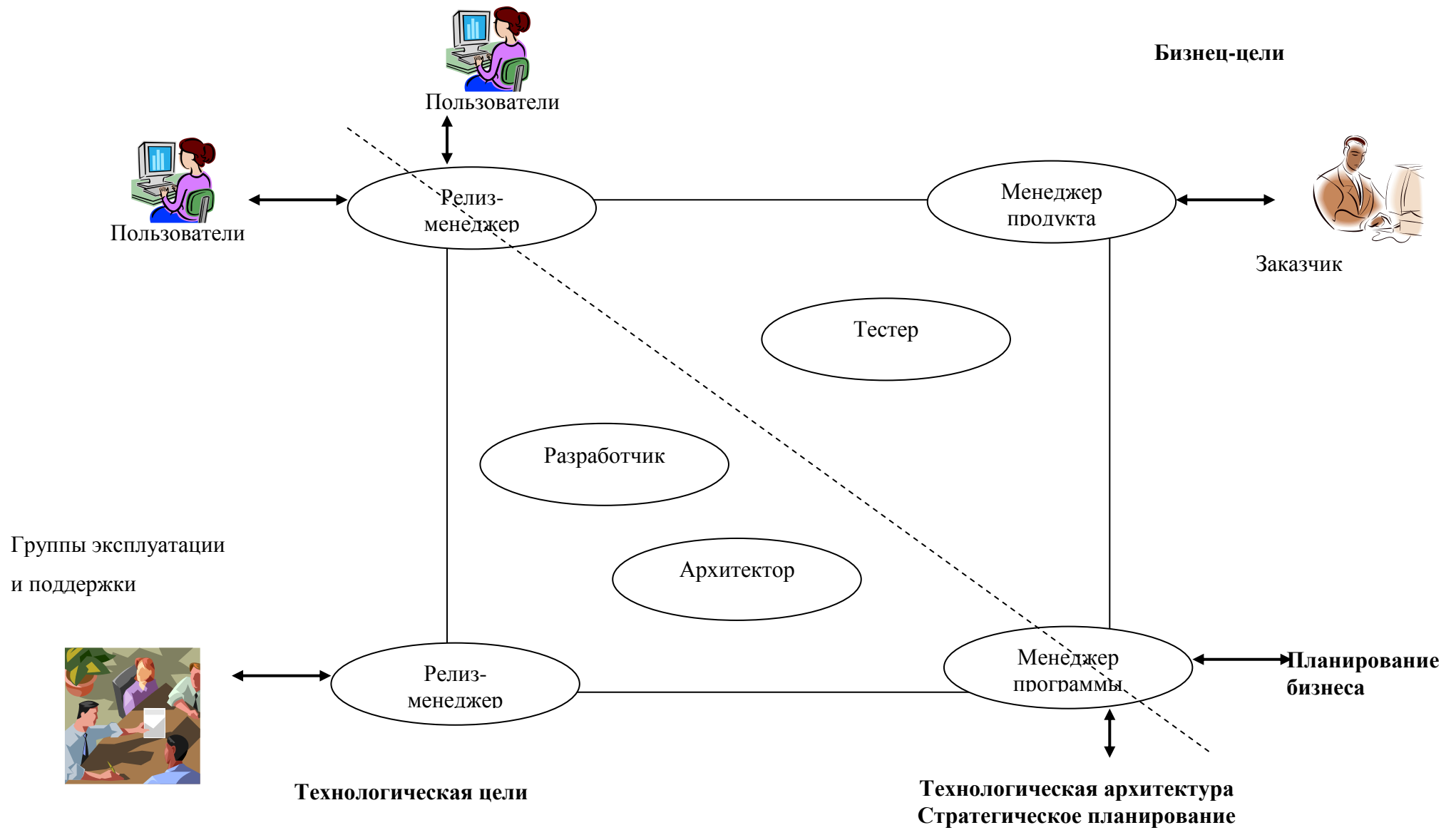


Рисунок 4.7 – Взаимодействие проектной группы с представителями заказчика

Учебный пример. Формирование команды

Пусть проектная группа состоит из 4 человек. Представим возможное распределение ролей, позволяющее выполнить поставленную учебную задачу.

Каждому участнику нашей команды невозможно выдать по одной роли. Разработчиков должно быть больше одного, а остальные роли придется совмещать. Как именно, сейчас обсудим.

Во-первых, следуя рекомендациям MSF по объединению ролей, дадим одному из разработчиков еще и роль архитектора.

Во-вторых, отбросим в сторону другую крайность – разработчиками не могут быть все. Отдельный участник команды должен заниматься тестированием. Ему же можно выдать «в нагрузку» роль бизнес-аналитика.

Незадействованными остались ролевые группы «Управление программой» и «Управление выпуском». Соответственно роли менеджер проекта и релиз-менеджер достаются еще одному участнику.

В итоге получаем следующее (возможное) распределение:

- ✓ Участник 1 – менеджер проекта и релиз-менеджер;
- ✓ Участник 2 – архитектор и разработчик;
- ✓ Участник 3 – бизнес-аналитик и тестер;
- ✓ Участник 4 – разработчик.

5 УПРАВЛЕНИЕ РИСКАМИ В MSF FOR AGILE SOFTWARE DEVELOPMENT

Дисциплина управления рисками MSF возводит процесс управления рисками в ранг стратегической задачи, затрагивающей все фазы проекта. В рамках MSF управление рисками – это процесс выявления, анализа и превентивной работы над рисками в целях избежания их превращения в проблемы, приносящие ущерб или иной вред.

В ходе всего проекта команда должна уделять внимание дисциплине управления рисками. Основные ее характеристики:

- она всеобъемлюща и принимает во внимание все составляющие проекта: людей, бизнес-процессы, технологические элементы и т.д.;
- она включает в себя пошаговый, систематический и воспроизводимый процесс управления рисками проекта;
- ее использование непрерывно на протяжении всего жизненного цикла проекта;
- она превентивна и не исходит из идеологии действия по факту случившегося.
- она вовлекает всю проектную группу в непрерывное извлечение уроков из полученного опыта.

5.1 Основные сведения о рисках

Прежде всего, определим, что такое «риск». Заглянем в «Толковый словарь русского языка» С.И. Ожегова: «Риск – 1. Возможность опасности, неудачи. 2. Действие наудачу в надежде на счастливый исход» [41]. Отметим, что понятие «риск» включает в себя два по сути противоположных толкования: одно со знаком минус, второе со знаком плюс.

Риск проекта (*project risk*) понимается в MSF именно в таком полном виде – как всякое событие или условие, которое может оказать как негативное, так и позитивное влияние на итоги проекта [19]. Такое же расширенное понимание спекулятивного риска (*speculative risk*) присутствует, например, в финансовой индустрии, где решения в условиях неопределенности могут привести к получению прибыли точно так же, как и к убыткам. Указанное понимание противоположно понятию чистого риска (*pure risk*) в индустрии страхования, где неопределенности связываются только лишь с потенциальными убытками в будущем.

Важно отметить, что риски не есть проблемы. Проблемы – это нечто, имеющие место в настоящее время, в то время как риски относятся к будущему и носят вероятностный характер (могут и не состояться). Однако риски могут стать проблемами, если ими эффективно не управлять.

Цель управления рисками – максимизировать их положительное влияние (открывающиеся возможности), но при этом минимизировать связанные с ними негативные факторы (убытки).

Дисциплина управления рисками в MSF основана на убеждении, что такое управление должно выполняться превентивно; это часть формального и систематического процесса, трактующего усилия, затрачиваемые на управление рисками, с позитивной точки зрения.

Говоря о рисках, MSF выделяет несколько ключевых концепций:

1. *Риск – неотъемлемая часть всякого проекта или процесса.* Несмотря на то, что различные проекты могут быть связаны с большим или меньшим числом рисков, не существует ни одного проекта, полностью свободного от них. Цель состоит не в том, чтобы избежать рисков, а в том, чтобы предвосхищать потенциальные проблемы и заблаговременно готовиться к их решению, если они возникнут.
2. *Выявление рисков нужно всячески одобрять.* Проектная группа должна смотреть на выявление рисков как на позитивную деятельность. Знание о существовании рисков – необходимое условие эффективной работы над ними. Следовательно, перспективы успеха проектной группы от проведения работы по выявлению рисков лишь увеличиваются.
3. *Оценка рисков должна вестись постоянно.* Обстоятельства, в которых проектная группа работает над созданием решения, обладают постоянной изменчивостью, следовательно, команда должна регулярно проводить переоценку выявленных рисков и постоянно следить за появлением новых. Управление рисками должно быть интегрировано в общий жизненный цикл проекта.
4. О положении дел в проекте нужно судить не по количеству рисков, связанных с его выполнением, а по степени проработанности процедуры их выявления, анализа и управления ими.

5.2 Планирование управления рисками

Во время проектных фаз выработки концепции и планирования проектная группа должна разработать формальный документ, описывающий управление рисками в проекте. В этом документе должны быть даны ответы на целый ряд вопросов, из которых рассмотрим основные.

- Как будет реализовываться процесс управления рисками? Из каких шагов состоит этот процесс?

- Кто будет осуществлять действия по управлению рисками? Какие для этого требуются навыки/квалификация? Требуется ли дополнительное обучение?
- Как будут строиться планы управления рисками и планы мероприятий по смягчению возможных негативных последствий?
- Как деятельность по управлению рисками будет интегрирована в общий план проекта?
- Какие действия будут предпринимать отдельные члены проектной группы для управления рисками?
- Какие ресурсы доступны для управления рисками?
- Каковы временные ограничения в мероприятиях, связанных с управлением рисками?

Поскольку риски являются неотъемлемой частью всех фаз всех проектов от начала и до конца, должны быть изначально выделены и должным образом распределены ресурсы, необходимые для эффективного управления рисками. Планирование управления рисками осуществляется проектной группой во время фаз выработки концепции и планирования, и результирующий план управления рисками должен определять конкретные действия, ответственность за которые возложена на определенных членов проектной группы. Эти задачи должны быть интегрированы в сводный план проекта (*master project plan*) и в сводный календарный график проекта (*master project schedule*).

5.3 Процесс управления рисками

Дисциплина управления рисками MSF отстаивает превентивное управление рисками, непрерывную оценку имеющихся рисков и интеграцию этих процессов в общую деятельность по принятию решений на протяжении всего жизненного цикла проекта или бизнес-процесса.

Процесс управления рисками MSF определяет шесть логических шагов, посредством которых проектная группа управляет текущими рисками, разрабатывает и исполняет стратегии управления рисками и извлекает уроки из своего опыта для использования на уровне всего предприятия (см. рисунок 5.1).

1. *Выявление рисков (risk identification)* – этап, позволяющий членам проектной группы вынести на обсуждение всей команды факты наличия рисков. Выявление рисков является начальной стадией процесса управления ими. Оно должно быть осуществлено как можно раньше, и к нему необходимо постоянно возвращаться на протяжении всего жизненного цикла проекта.

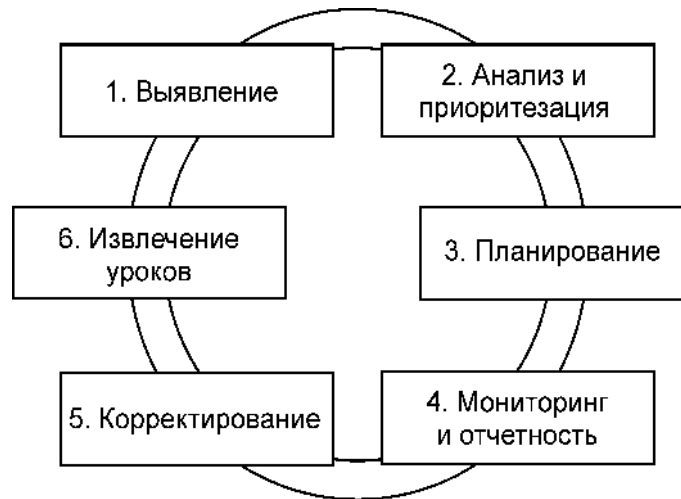


Рисунок 5.1 - Процесс управления рисками MSF [31]

2. *Анализ рисков (risk analysis)* – этап преобразования накопленных во время предыдущего шага оценок и данных в форму, позволяющую осуществить приоритезацию рисков. Приоритезация рисков (*risk prioritization*) позволяет проектной группе управлять наиболее важными из них, выделяя для этого необходимые ресурсы.
3. *Планирование рисков (risk planning)* выполняется исходя из информации, полученной на этапе их анализа, и имеет целью выработку стратегий, планов и конкретных шагов. Календарное планирование рисков (*risk scheduling*) интегрирует эти планы в повседневный процесс управления проектом, обеспечивая непрерывность управления рисками. Эта стадия напрямую увязывает планирование рисков с планированием проекта в целом.
4. *Мониторинг рисков (risk tracking)* выполняется для наблюдения за конкретными рисками и прогрессом в осуществлении составленных планов. Мониторингу должны быть подвергнуты сделанные *оценки вероятности (probability)* риска, его *угрозы (impact)*, *ожидаемая величина риска (exposure)* и прочие факторы, способные повлиять на приоритет рисков. Наблюдению подвергаются также составленные планы, имеющиеся ресурсы и принятый календарный график.
5. *Корректирование ситуации (risk control)* представляет собой процесс исполнения принятых в отношении рисков планов и контроля за ходом их исполнения. Этот процесс также включает в себя инициирование изменений всего проекта (*project change control requests*), если изменения в состоянии рисков либо в соответствующих планах влияют на прогнозируемый объем работы, требуемые ресурсы или сроки.

6. *Извлечение уроков* (risk learning) формализует процесс усвоения накопленного за время работы над проектом опыта.

Необходимо отметить, что описанные этапы являются логическими шагами и не обязательно должны следовать друг за другом в строгом хронологическом порядке. Проектные группы могут циклически повторять шаги выявления-анализа-планирования по мере обнаружения дополнительных факторов, влияющих на проект.

5.4 Управление рисками как составная часть жизненного цикла проекта

Процесс управления рисками в MSF тесно интегрирован с общим жизненным циклом проекта. Оценка рисков может быть начата даже на этапе выработки концепций, поскольку в этот момент проектная группа и заинтересованные стороны начинают формировать видение проекта, его границ и рамок. По результатам анализа и планирования рисков необходимые планы по предотвращению и смягчению последствий должны быть сразу включены в календарный график проекта и его сводный план.

В ходе выполнения проекта команда постоянно проводит мониторинг рисков, осуществляет необходимые корректирующие действия в соответствии с расписанием и планом проекта, и при наступлении связанных с триггерами рисков событий.

Действия по выявлению и анализу новых рисков могут проводиться по достижении *вех* (*milestones*) проекта (подробнее о вехах далее в этой лекции). Также в этот момент можно резюмировать извлеченные уроки.

По ходу проекта тип рисков, которым уделяется внимание, также должен изменяться. На ранних этапах доминируют риски связанные с бизнесом, рамками проекта, требованиями к конечному продукту и проектированием этого продукта. С течением времени начинают играть важную роль технологические риски, связанные с реализацией проекта. Затем внимание переходит к рискам поддержки и сопровождения. Для организации деятельности по выявлению рисков в моменты основных фазовых переходов полезно использовать контрольные перечни (*checklists*) и классификации рисков.

5.6 Пример «Выделение рисков»

Рассмотрим учебный пример «Система бронирования билетов для авиакомпании» и выделим некоторые возможные риски (таблица 5.1).

Таблица 5.1 – Возможные риски

	Наименование риска	Комментарий
1	Не успеем сдать проект во время	Из-за неправильной организации работ затратим больше времени, чем заявлено в контракте
2	Не хватит квалификации персонала	При создании продукта используется новая технология (JNL), персонал с ней не работал и может не разобраться
3	Один из членов команды заболит	Команда не многочисленна и отсутствие одного из членов команды ведет задержке в работах
4	Заказчик изменит требования	В данный момент требования согласованы с заказчиком и утверждены, но в процессе работы заказчик может захотеть добавить в решение новую функциональность
5	Авария на подстанции, отключение электричества	Потеряем время, пока авария будет устраняться.
6	Отключат доступ к Интернету	Ухудшится возможность быстрого получения необходимых сведений, пропадет электронная почта и другие средства коммуникации
7	Заказчик вовремя не оплатит счета	Задержка в покупке необходимого оборудования
8	Из-за необнаруженной вовремя ошибки система нанесет урон заказчику	На время устранения ошибки пассажиры не смогут заказывать билеты через систему. Потребуется “ручная” работа персонала. Компания потеряет возможных клиентов и часть прибыли.
9	На стороне заказчика нет заявленной в требованиях аппаратуры	Не сможем адекватно развернуть систему
10	Не сможем подобрать необходимые кадры	Придется совмещать роли

Как видно, рисков не так уж много, однако можно заметить, что они происходят из самых разных областей и связаны с людьми (риск №3), с процессами (риски №1, №8, №10), с технологиями (риск №2), с внешними условиями (риски №5, №6), с заказчиком (риски №4, №7, №9). Помимо представленных риски могут иметь и другие источники.

Далее MSF рекомендует для каждого риска представить формулировку – выражение на естественном языке причинно-следственной связи между реально существующим фактором проекта и потенциально возможным, еще не случившимся событием или ситуацией. Первая часть формулировки риска называется *условием* (*condition*) и содержит описание существующего фактора или особенности проекта, которые, по мнению проектной группы, могут сделать результат проекта убыточным либо же сократить получаемую от проекта прибыль. Вторая часть формулировки риска называется *(по)следствием* (*consequence*). Она описывает ту нежелательную ситуацию, которой следует избежать.

В качестве примера представим формулировки некоторых выявленных выше рисков (таблица 5.1).

Таблица 5.1 – Перечень причин и последствий возникновения рисков

Первопричина	Условие	Последствие	Приносимый ущерб
Нехватка кадров	Один из членов команды заболеет	Функции заболевшего придется передать другому	Потери времени
Форс-мажор	Авария на подстанции, отключение электричества	Потеряем время, пока авария будет устраняться	После устранения аварии придется увеличить нагрузку, чтобы наверстать потери времени
Организация работы	Не сможем подобрать необходимые кадры	Участникам придется совмещать роли	Дополнительные трудозатраты, снижение качества продукта, увеличение времени разработки решения