Глава 2

ОСНОВЫ УПРАВЛЕНИЯ ПРОГРАММНЫМИ ПРОЕКТАМИ

2.1 Основные понятия и определения

Классическое управление проектами выделяет два вида организации человеческой деятельности: операционную и проектную. *Операционная деятельность* (например, функционирование службы поддержки пользователя) применяется, когда внешние условия хорошо известны и стабильны, когда производственные операции хорошо изучены и неоднократно испытаны, а функции исполнителей определены и постоянны. В этом случае основой эффективности служат узкая специализация и повышение компетенции. *Проектная деятельность* (например, создание нового программного продукта) используется в том случае, когда разрабатывается новый продукт, внешние условия и требования к которому постоянно меняются, где применяются в основном оригинальные методы и технологии разработки, где постоянно требуются поиск новых решений, интеллектуальные усилия и творчество. Задача проектной деятельности — достижение конкретной бизнес-цели, задача операционной деятельности — обеспечение нормального течения бизнеса.

Другими словами, операционная и проектная деятельности различаются, главным образом, тем, что операционная деятельность — это продолжающийся во времени и повторяющийся процесс, в то время как проекты являются уникальными по содержанию и всегда ограничены во времени.

Основные особенности управления программными проектами заключаются в следующем:

- программный продукт не материален, его нельзя увидеть в процессе конструирования и, следовательно, оперативно повлиять на его реализацию;
- жизненный цикл ПП в существующих стандартах описан в общем виде и прямо не ориентирован на специфику конкретного продукта, необходимо адаптировать его к виду и типу проекта и разработать методики его выполнения исполнителями;

- программные продукты как результаты творческого труда не поддаются точному оцениванию как по времени создания, так и по требуемому бюджету.

Вследствие этих особенностей только 35% проектов завершились в срок, не превысили запланированный бюджет и реализовали все требуемые функции и возможности.

46% проектов завершились с опозданием, расходы превысили запланированный бюджет, требуемые функции не были реализованы в полном объеме. Среднее превышение сроков составило 120%, среднее превышение затрат 100%, обычно исключалось значительное число функций.

19% проектов полностью провалились и были аннулированы до завершения по следующим причинам: требования заказчика не выполняются; проект не вложился в стоимость; проект не вложился в заданные сроки; этапы работ оказались нескоординированными друг с другом [11].

В литературе приводятся следующие определения проекта как законченного продукта [3, 12, 13]:

1) проект — комплекс взаимосвязанных мероприятий, предназначенных для достижения целевых результатов при решении многокритериальных задач в течение заданного периода времени при установленном бюджете в условиях ограниченных ресурсов;

2) проект — это ограниченное по времени целенаправленное изменение исследуемой системы с установленными требованиями к качеству результатов, возможными объемами расхода средств, ресурсов и специфической организацией управления. Управление проектом — это управление этими изменениями с высокой степенью уверенности в успешном исходе;

3) проект — временное предприятие, предназначенное для создания уникальных продуктов, услуг или результатов;

4) проект — это комплекс усилий, предпринимаемых с целью получения конкретных уникальных результатов в рамках отведенного времени и в пределах утвержденного бюджета, который выделяется на оплату ресурсов, используемых или потребляемых в ходе проекта;

5) проект есть комплекс действий, направленных на получение уникального результата, будь то продукт или услуга. Суть результата — его содержание. Для информационной системы — ее функциональность;

6) проект представляет собой комплекс уникальных действий, не опирающийся на организационную структуру, имеющий определенные дату начала и окончания, расписание, стоимость и технические задачи.

Следует различать следующие понятия: цели, результаты, ограничения и допущения.

Цель проекта описывает, какие задачи должны быть решены в результате проекта, желаемый результат, достигнутый в пределах некоторого интервала времени, другими словами, должен отвечать на вопрос, *зачем* данный проект нужен, какие потребности бизнеса он удовлетворяет. Например, целями проекта могут быть:

- завоевание значительной доли растущего рынка за счет вывода на него нового продукта;
- разработка ПО под потребности конкретного заказчика;
- доработка программного продукта в целях приведения его в соответствие с изменениями в законодательстве.

Цели должны быть значимыми, конкретными, измеримыми и достижимыми. Четкое определение бизнес-целей важно, поскольку существенно влияет на все процессы и решения в проекте. Проект должен быть закрыт, если признается, что достижение цели невозможно или стало нецелесообразным. Например, если реальные затраты на проект будут превосходить будущие доходы от его реализации.

Результаты проекта отвечают на вопрос, *что* должно быть получено после его завершения. Результаты проекта должны определять: какие именно бизнес-выгоды получит заказчик в результате проекта; какой продукт или услуга будут получены по окончании проекта; краткое описание и при необходимости ключевые свойства и/или характеристики продукта/услуги.

Следует помнить, что результаты проекта должны быть измеримыми. Это означает, что при оценке результатов проекта должна иметься возможность сделать заключение: достигнуты оговоренные в концепции результаты или нет.

Ограничения являются неотъемлемой частью проекта и сокращают возможности проектной команды в выборе решений. В частности, они могут содержать:

- требования обязательной сертификация продукта, услуги на соответствие определенным стандартам;
- требования на использование конкретной заданной программно-аппаратной платформы;
- специфические требования к защите информации.

Допущения, как правило, тесно связаны с управлением рисками, о которых заказчик должен знать заранее. Например, оценивая проект по схеме с фиксиро-

ванной ценой, можно записать в допущении предположение о том, что стоимость лицензий на ПО, приобретаемое на стороне, не изменится до завершения проекта.

Оптимальная реализация программного проекта должна обеспечить достижение конкретной бизнес-цели при соблюдении ограничений «железного треугольника» (рис. 2.1) [22].

Рис. 2.1 – «Железный треугольник» ограничений проекта

Ни один из углов треугольника не может быть изменен без изменения других. Например, чтобы уменьшить время, потребуется увеличить бюджет и/или сократить содержание.

В приложении к индустрии программного обеспечения к трем основным ограничениям «железного треугольника» добавляют четвертое ограничение — *приемлемое качество* [22]. В этом случае система ограничений должна строиться на основе приоритетов проекта и должна учитывать требования потребителей к создаваемому продукту или услуге. Если необходим жесткий предопределенный набор функциональности — понятно, что «плавающими» характеристиками проекта (вторичными по своей природе, требующими компромисса в контексте требуемого объема функциональности) будут требуемое время и необходимый бюджет (рис. 2.2).

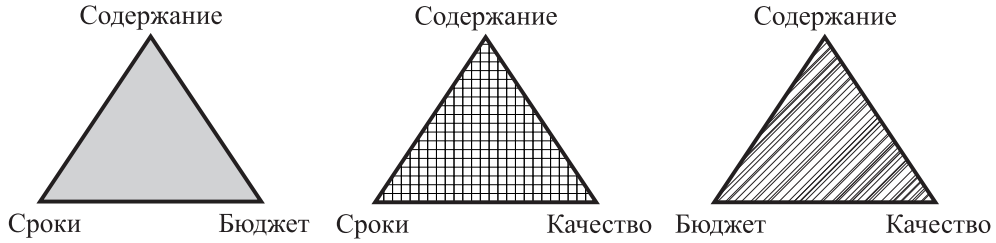


Рис. 2.2 – Возможные варианты «железного треугольника» ограничений проекта

Если бюджет не является жестко определенным и может быть предметом обсуждения, другие ограничения все равно будут играть свою роль. Поэтому считается, что неотъемлемой частью любого программного проекта является *анализ компромиссов*, направленный на выделение функциональных требований, которые можно реализовать в заданных ограничениях проекта, будь то сроки, бюджет, качество или другие характеристики. В общем случае, *задача такого анализа* - нахождение баланса, приемлемого для всех сторон, связанных с проектом; заказчиков, которым нужна определенная функциональность в конкретные сроки, при имеющемся бюджете; исполнителей, которые обладают бюджетом, достаточным для определенной стоимости использования ресурсов, трудоемкости проекта и достижения качества в заданные сроки. Согласно стандарту PMBOK проект считается успешным, если он выполнен в срок в соответствии со спецификациями и в пределах запланированного бюджета.

Общепринято, что не существует единственного правильного процесса разработки ПО. В каждом новом проекте в соответствии с «Законом 4-х П» (рис. 2.3) процесс должен определяться каждый раз заново, в зависимости от проекта, продукта и персонала.

Рис. 2.3 – «Закон 4-х П»

Совершенно разные процессы должны применяться в проектах, в которых участвуют 5 человек, и в проектах, в которых участвуют 500 человек. Если продуктом проекта является критическое ПО, например система управления атомной электростанцией, то процесс разработки должен сильно отличаться от разработки сайта. И, наконец, по-разному следует организовывать процесс разработки в команде начинающих программистов и в команде состоявшихся профессионалов [22].

Управление проектом — это руководство работами команды исполнителей проекта для реализации проекта с использованием общих методов, планирования и контроля работ (видение будущего продукта, стартовые операции, планирование итераций, мониторинг и отчетность), планирование и управление рисками, эффективной организацией работы команды и коммуникационными потоками в команде исполнителей.

В стандарте PMBOK (Project Management Body Of Knowledge — Свод знаний по управлению проектами) приведены 9 процессов (областей знаний) по управлению проектами, каждый из которых состоит из определенного набора работ, и пять этапов (фаз) жизненного цикла проекта: инициация, планирование, исполнение, мониторинг и управление, завершение [1]. При этом процессы взаимосвязаны, а *этапы проекта* могут накладываться во времени друг на друга. Распределение работ по этапам жизненного цикла, рекомендованное стандартом, приведено в табл. 2.1.

Таблица 2.1 – Распределение процессов управления проектом по этапам

Процессы и области знаний	**Группы процессов управления проектами по фазам проекта**				
	Инициация	**Планирование**	**Исполнение**	**Мониторинг и управление**	**Завершение**
1 Интеграция управления проектом	1.1 Разработка Устава проекта 1.2 Разработка предварительного описания содержания проекта	1.3 Разработка плана управления проектом	1.4 Руководство и управление исполнением проекта	1.5 Мониторинг и контроль (управление) работ проекта 1.6 Общее управление изменениями	1.7 Закрытие проекта
2 Управление содержанием проекта		2.1 Планирование содержания проекта 2.2 Определение содержания 2.3 Создание иерархической структуры работ (ИСР)		2.4 Подтверждение содержания 2.5 Управление содержанием	
3 Управление сроками проекта		3.1 Определение состава операций 3.2 Определение взаимосвязей операций 3.3 Оценка ресурсов операций 3.4 Оценка длительности операций 3.5 Разработка расписания		3.6 Управление расписанием	
4 Управление стоимостью проекта		4.1 Стоимостная оценка 4.2 Разработка бюджета расходов		4.3 Управление стоимостью	
5 Управление качеством проекта		5.1 Планирование качества	5.2 Процесс обеспечения качества	5.3 Процесс контроля качества	

продолжение на следующей странице

Таблица 2.1 – Распределение процессов управления проектом по этапам

Процессы и области знаний	**Группы процессов управления проектами по фазам проекта**				
	Инициация	**Планирование**	**Исполнение**	**Мониторинг и управление**	**Завершение**
6 Управление человеческими ресурсами проекта		6.1 Планирование человеческих ресурсов	6.2 Набор команды проекта 6.3 Развитие команды проекта	6.4 Управление командой проекта	
7 Управление коммуникациями проекта		7.1 Планирование коммуникаций	7.2 Распространение информации 7.3 Отчетность по исполнению	7.4 Управление участниками проекта	
8 Управление рисками проекта	8.1 Планирование управления рисками 8.2 Идентификация рисков 8.3 Качественный анализ рисков 8.4 Количественный анализ рисков 8.5 Планирование реагирования на риски		8.6 Мониторинг и управление рисками		
9 Управление поставками проекта		9.1 Планирование покупок и приобретений 9.2 Планирование контрактов	9.3 Запрос информации у продавцов 9.4 Выбор продавцов	9.5 Администрирование контрактов	9.6 Закрытие контракта

На этапе *инициации проекта* необходимо выбрать и обосновать вид (тип) программного продукта, который компания собирается разрабатывать, и определить рыночную нишу его распространения, разработать и утвердить концепцию проекта. Недостаточное внимание этого этапа проекта неизбежно приводит к существенным проблемам при планировании, реализации и завершении проекта.

Планирование программного проекта относится к работам, предпринимаемым для подготовки к успешному ведению программно-инженерной деятельности реализации программного продукта и представляет собой процессы структурного планирования проекта, распределения и назначения ресурсов (материальных и людских) с учетом стоимости и времени выполнения проекта в целом и его отдельных работ. Основополагающими методами планирования, применяемыми на практике, являются: *метод критического пути* — CPM (Critical Path Method) и *метод анализа и оценки программ* PERT (Program Evaluation and Review Technique).

Как правило, редкий проект выполняется в соответствии с первоначальными планами, поэтому важным элементом системы управления проектом является периодический *мониторинг* его состояния, анализ причин расхождения с планом и разработка корректирующих воздействий.

Завершение проекта относится к фиксированию результатов программного проекта после передачи полученного программного продукта в эксплуатацию. На этом этапе проводятся приемо-сдаточные испытания (ПСИ) продукта на предмет соответствия его свойств определенным ранее требованиям. Критерии приемки должны определять числовые значения характеристик системы, которые должны быть продемонстрированы по результатам приемо-сдаточных испытаний или опытной эксплуатации и однозначно свидетельствовать о достижении целей проекта. Для проведения процедуры приемки-сдачи создаются специальные документы — программа и методика испытаний программного продукта. Завершение наступает, когда достигнуты цели проекта; или осознано, что цели проекта не будут или не могут быть достигнуты; или исчезла необходимость в проекте и он прекращается.

2.2 Управление рисками проекта

В методологии по управлению IT-проектами Microsoft Solutions Framework (MSF) компании Microsof [14] под риском проекта понимается событие или условие, которое может оказать как негативное, так и позитивное влияние на итоги проекта, и отмечается, что риски не есть проблемы.

Проблемы — это нечто, имеющее 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 нега-

тивные факторы (убытки). К минимизации рисков стремятся все потенциальные участники проекта: разработчики (поставщики) ПП, заказчики (потребители), а также предполагаемые инвесторы. Управление рисками — это определенная деятельность, которая выполняется в проекте от его начала и до завершения. 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.

Процесс управления рисками состоит из логически взаимосвязанных этапов: *идентификация рисков, анализ рисков, планирование рисков, мониторинг и управление рисками* [12]. 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.

Идентификация рисков — этап, позволяющий определить и вынести на обсуждение команды факты наличия рисков, способных повлиять на проект, и документально оформить их характеристики. Выявление рисков является начальной стадией процесса управления ими. Это интерактивный процесс, который периодически повторяется на всем протяжении проекта, поскольку в рамках его жизненного цикла могут обнаруживаться и новые риски.

Исходные данные для выявления и описания характеристик рисков могут браться из разных источников. В первую очередь, это информация о выполнении прежних проектов. Следует помнить, что проблемы завершенных и выполняемых проектов — это, как правило, риски в новых проектах. Другим источником данных о рисках проекта может служить разнообразная информация из открытых источников, научных работ, маркетинговая аналитика и другие исследовательские работы в данной области. Каждый проект задумывается и разрабатывается на основании ряда гипотез, сценариев и допущений. Неопределенность в допущениях проекта следует также обязательно рассматривать в качестве потенциального источника возникновения рисков проекта.

Результатом идентификации рисков должен стать список рисков с описанием их основных характеристик. Рекомендуется каждый риск формулировать на естественном языке причинно-следственной связи между реально существующим фактором проекта и потенциально возможным, еще не случившимся событием или ситуацией [12]. Первая часть формулировки риска — условие — содержит описание существующего фактора или особенности проекта, которые, по мнению членов проектной группы, могут сделать результат проекта убыточным либо сократить получаемую от проекта прибыль. Вторая часть формулировки риска называется последствием. Она описывает ту нежелательную ситуацию, которой следует избежать. Пример формулировки выявленных рисков представлен в табл. 2.2.

Таблица 2.2 – Описание рисков проекта

Причина	**Условия**	**Последствия**	**Ущерб**
Требования не ясны	Отсутствие описания сценариев использования системы	Задержка начала разработки ППО. Большой объем переработок	Задержки в сроках сдачи готового продукта и дополнительные трудозатраты
Недостаток квалифицированных кадров	Архитектура и код низкого качества	Большое число ошибок. Большие затраты на их исправление	Задержки в сроках сдачи готового продукта и дополнительные трудозатраты
Текучесть кадров	Частая смена участников команды	Низкая производительность при вводе новых участников в проект	Задержки в сроках сдачи готового продукта и дополнительные трудозатраты

Выявление рисков — ответственный и важный этап проекта. Знание о существовании рисков — необходимое условие эффективной работы по предотвращению рисков.

Анализ рисков — этап обработки данных, накопленных при идентификации рисков в формы, позволяющие осуществить качественную и количественную оценки рисков: вероятности наступления риска, его угрозы, ранжирование рисков по степени возможных угроз, ожидаемую величину потерь и т. д.

Качественный анализ рисков проекта включает определение вероятности наступления рисков, тяжести последствий от рисков, степени опасности (ранга) риска, близости наступления риска [12]. Для измерения параметров рисков применяются, как правило, порядковые шкалы либо шкалы интервалов. Определение тяжести последствий от рисков предлагается оценивать в шкале интервалов (табл. 2.3).

Таблица 2.3 – Относительная шкала оценки воздействия рисков

Количественное значение оценки	< 0,4	0,4–0,7	> 0,7
Качественное значение оценки	Умеренные	Критичные	Катастрофические
Потери от наступления риска	Потери менее...	Потери от... до...	Потери более...

Риск может воздействовать и на сроки проекта, и на качество получаемого продукта, но все эти отклонения могут быть оценены в денежном эквиваленте. Например, последствия задержки по срокам могут быть выражены в сумме денежных санкций в контракте.

Похожая шкала может быть применена для оценки вероятности наступления риска (табл. 2.4).

Таблица 2.4 – Относительная шкала измерения вероятности наступления риска

Количественное значение вероятности	< 0,4	0,4–0,7	> 0,7
Качественное значение вероятности	Маловероятно	Возможно	Очень вероятно
Возможность наступления риска	Наступление события весьма сомнительно	Шансы равны	Шансы наступления весьма велики

Ранжирование рисков позволяет проектной группе управлять наиболее важными из них, выделяя для этого необходимые ресурсы. Для определения ранга риска используется информация матриц вероятностей и воздействий (табл. 2.5). Ранг риска определяет его порядковый номер в полной совокупности рисков проекта. Чем выше ранг, тем более опасен риск.

Таблица 2.5 – Матрица рангов выявленных рисков проекта

Причина	**Вероятность**	**Воздействие**	**Ранг**
Требования не ясны	Очень вероятно	Катастрофическое	9
Недостаток квалифицированных кадров	Очень вероятно	Критичные	6
Текучесть кадров	Возможно	Критичные	4

Одной из важных характеристик риска является *близость его наступления.* Естественно, что при прочих равных условиях рискам, которые могут осуществиться уже завтра, следует сегодня уделять больше внимания, чем тем, которые могут произойти не ранее чем через полгода. Возможная шкала оценки близости риска представлена в табл. 2.6.

Таблица 2.6 – Относительная шкала измерения близости наступления риска

Количественное значение близости наступления	Больше чем через . . .	От . . . до	Меньше чем через. . . .
Качественное значение *близости наступления*	Очень нескоро	Не очень скоро	Очень скоро

Итоговые результаты анализа рисков подробно оформляются в виде следующих документов (табл. 2.7).

Таблица 2.7 – Пример карточки с описанием риска

Номер: R-101	Категория: технологический
Причина: недостаток квалифицированных кадров	Симптомы: Разработчики будут использовать новую платформу -J2EE.
Последствия: низкая производительность разработки	Воздействие: Увеличение сроков и трудоемкости разработки
Вероятность: очень вероятно	Степень воздействия: критическая
Близость: очень скоро.	Ранг: 6
Исходные данные: «Содержание проекта», «План обеспечения ресурсами», протоколы совещаний №21 от . . . , №27 от	

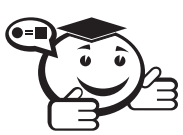
Оценка рисков должна вестись постоянно. Обстоятельства, в которых проектная группа работает над созданием решения, обладают постоянной изменчивостью, следовательно, команда должна регулярно проводить переоценку выявленных рисков и постоянно следить за появлением новых. Управление рисками должно быть интегрировано в общий жизненный цикл проекта.

Результаты качественного анализа используются в ходе последующей количественной оценки рисков и планирования мероприятий по реагированию на риски.

Количественная оценка рисков позволяет определять:

- вероятность достижения конечной цели проекта;
- степень воздействия риска на проект и объемы непредвиденных затрат и материалов, которые могут понадобиться;
- риски, требующие скорейшего реагирования и большего внимания, а также влияние их последствий на проект;
- фактические затраты и предполагаемые сроки окончания работ на проекте.

Количественная и качественная оценки рисков могут применяться в отдельности или вместе в зависимости от времени и бюджета.



Планирование рисков — это процесс определения конкретных действий (мероприятий) по управлению рисками проекта, тщательное и подробное планирование которыми позволяет:

- *определить возможные потери от наступления рисков;*
- *выделить достаточное количество времени и ресурсов для выполнения операций по управлению рисками;*
- *повысить вероятность успешного достижения результатов проекта.*

В соответствии с [8] исходными данными для планирования управления рисками служат:

- отношение и толерантность к риску организаций и лиц, участвующих в проекте, что оказывает влияние на план управления проектом. Это должно быть зафиксировано в основных принципах и подходах к управлению рисками;
- стандарты организации, в которых должны быть изложены подходы к управлению рисками, например категории рисков, общее определение понятий и терминов, стандартные шаблоны, схемы распределения ролей и ответственности, а также определенные уровни полномочий для принятия решений;
- подробное описание содержания проекта;
- план управления проектом, формальный документ, в котором указано, как будет исполняться проект и как будет происходить мониторинг и управление проектом.

План управления рисками обычно включает следующие элементы:

- определение подходов, инструментов и источников данных, которые могут использоваться для управления рисками в данном проекте;
- распределение ролей и ответственности выполнения каждого мероприятия (позиции), включенного в план управления рисками, назначение сотрудников на эти позиции и разъяснение их ответственности;
- оценка стоимости мероприятий и выделение ресурсов, необходимых для управления рисками. Эти данные включаются в базовый план по стоимости проекта;
- структуризация, систематизация и всесторонняя идентификация рисков с нужной степенью детализации;
- определение сроков и частоты выполнения процесса управления рисками на протяжении всего жизненного цикла проекта, а также определение операций по управлению рисками, которые необходимо включить в план реализации проекта;
- определение уровней вероятности наступления рисков, шкалы воздействия и близости рисков на проект.

В общем случае любой риск характеризуется [12] (рис. 2.4):

- причинами, обусловливающими наступление риска;
- симптомами, указывающими на то, что событие риска произошло или вот-вот произойдет;
- последствиями — проблемами, которые могут появиться при реализации проекта в результате произошедшего риска;
- воздействиями на возможность достижения целей проекта: изменения стоимости, графика и технических характеристик разрабатываемого ПП.

Планирование мероприятий по снижению (ликвидации) рисков следует начинать после проведения качественного и количественного анализа рисков, при

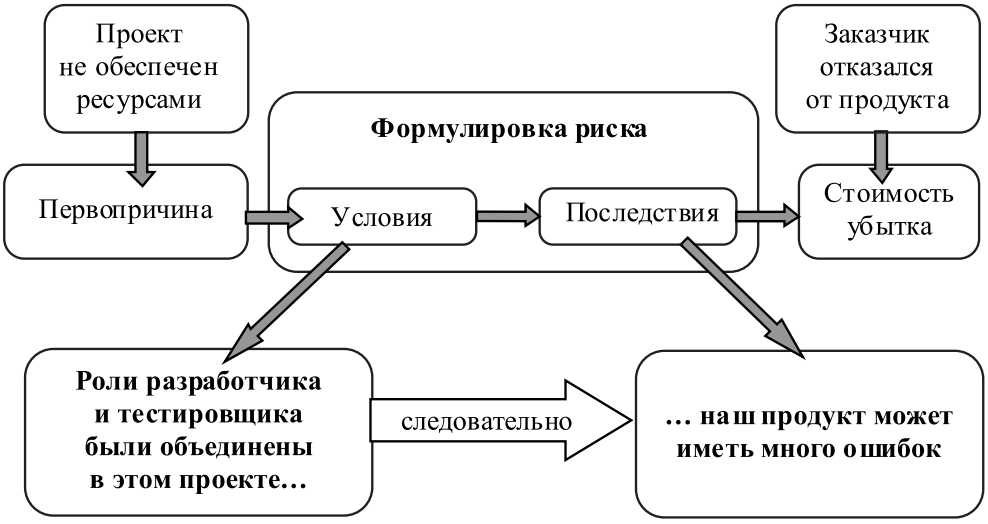


Рис. 2.4 – Основные характеристики риска и их взаимосвязи

этом все мероприятия должны соответствовать серьезности риска, быть экономически эффективными, своевременными, реалистичными и согласованными со всеми участниками проекта.

Согласно [12] возможны четыре вида таких мероприятий: уклонение от риска, передача риска, снижение рисков, принятие риска.

Уклонение от риска предполагает изменение плана управления проектом таким образом, чтобы исключить угрозу, вызванную негативным риском, оградить цели проекта от последствий риска или ослабить цели, находящиеся под угрозой (например, уменьшить содержание проекта). Некоторых рисков, возникающих на ранних стадиях проекта, можно избежать при помощи уточнения требований, например отказаться от реализации рискованного функционального требования или самостоятельно разработать необходимый программный компонент, вместо ожидания поставок продукта от субподрядчика.

Передача риска подразумевает переложение негативных последствий угрозы с ответственностью за реагирование на риск на третью сторону (например, заказ на стороне разработки рискованного компонента). Передача риска просто переносит ответственность за его управление другой стороне, но риск при этом никуда не девается. Передача риска практически всегда предполагает выплату бонусов за риск стороне, принимающей на себя риск.

Снижение рисков предполагает понижение вероятности и/или последствий негативного рискованного события до приемлемых пределов. Принятие предупредительных мер по снижению вероятности наступления риска или его последствий часто оказывается более эффективным, нежели усилия по устранению негативных последствий, предпринимаемые после наступления события риска. Например, если высока вероятность увольнения сотрудников, то введение на начальной стадии в проект дополнительных (избыточных) трудовых ресурсов снижает потери при увольнении членов команды, поскольку не будет затрат на привлечение к выполнению проекта новых участников.

Принятие риска означает, что команда проекта осознанно приняла решение не изменять план управления проектом в связи с появлением рисков или не нашла

подходящей стратегии реагирования на него. Если же компания приняла решение управлять рисками, то необходимо их страхование путем закладывания резерва в оценки срока завершения проекта и/или увеличения трудозатрат, управляемых рисков.

В [8] при разработке мер по предотвращению рисков предлагается объединить их в две категории:

1) «известные неизвестные». Это те риски, которые можно идентифицировать и подвергнуть анализу. В отношении таких рисков можно спланировать ответные действия;
2) «неизвестные неизвестные». Это риски, вызванные непредвиденными обстоятельствами, которые невозможно идентифицировать и, следовательно, спланировать ответные действия. Единственное, что можем в этом случае предпринять, это создать резерв бюджета проекта на случай незапланированных, но потенциально возможных изменений. На расходование этого резерва руководитель проекта, как правило, обязан получать одобрение вышестоящего руководства. Финансовые резервы на непредвиденные обстоятельства не входят в базовый план по стоимости проекта, но включаются в бюджет проекта. Они не распределяются по проекту, как бюджет, и поэтому не учитываются при расчете освоенного объема.

Множество рисков, относящихся к категории «известных неизвестных», в свою очередь, условно можно разбить на определенные группы.

Риски, обусловленные непредвиденными изменениями рыночной ситуации:

1) изменениями нормативного регулирования деятельности компании;
2) изменениями ситуации на рынке программно-аппаратных средств;
3) изменениями ситуации на финансовом рынке;
4) ненадежной работой аутсорсинговых компаний.

Первые три риска можно снизить путем внесения в договор условий, предусматривающих корректировку порядка его исполнения в случае кризисных и иных явлений, не зависящих от воли сторон, а в последнем случае — путем внесения в договор пункта о штрафных санкциях за нарушение условий поставки продукта аутсорсинговой компанией.

Риски, обусловленные конкурентной борьбой:

1) высокой рыночной конкуренцией и, как следствие, ошибкой в объемах продаж ПП;
2) дискредитацией программного продукта со стороны конкурентов.

Возможные ошибки в объемах продаж непосредственно связаны с непредвиденной конкуренцией. Рекомендации по минимизации рисков в этом направлении сводятся к следующему:

- поставляемый продукт не должен быть полным аналогом существующих на рынке программных систем;
- рекомендуется первым завершить разработку и захватить рынок, даже если это приведет к увеличению затрат на первый экземпляр.

Кроме того, риски сильно зависят от неоправданных увеличений затрат на разработку, рекламу, влияет на риски и компьютерное пиратство. Снижение рисков на этапе разработки может быть достигнуто за счет эффективного использования в проекте готовых компонентов, а также постоянной оценки себестоимости разработки с возможностью внесения корректировок в случае незапланированных отклонений.

Следует принять ряд мер по возможной дискредитации программного продукта со стороны конкурентов, которая может осуществляться в следующих направлениях: нарушение целостности продукта, кражи и присвоение алгоритмов и программных кодов, пиратское распространение копий, необоснованное обвинение в нарушении права интеллектуальной собственности.

Внутренние (собственные) риски проекта:

1) требования заказчика не всегда точны и подвержены частым изменениям;
2) разработка функционально неправильных программных элементов неудачного пользовательского интерфейса;
3) отсутствие эффективного взаимодействия с заказчиком;
4) недостатки планирования проекта, появление «забытых работ»;
5) недооценка сложности проекта, ошибки в оценках трудоемкостей и сроков работ и, как следствие, нереалистичные сроки и бюджет проекта;
6) отсутствие у команды необходимых ресурсов и опыта;
7) недостатки во внутренней организации работ, неумение работать в реальном времени;
8) дефицит специалистов и / или высокая текучесть кадров;
9) разрыв в квалификации специалистов разных областей знаний;
10) нехватка информации о внешних компонентах, определяющих окружение ПП;
11) недостаточная производительность получаемой системы.

К наиболее часто упускаемым требованиям к ПП, как правило, относят:

1) отсутствие функциональных требований к программам установки, настройке, конфигурации; миграция данных; интерфейсам к внешним системам;
2) отсутствие общесистемных требований к производительности; надежность; открытость; масштабируемость; безопасность; кросплатформенность; эргономичность программных продуктов.

Эти требования «всплывают» при подготовке и проведении приемо-сдаточных испытаний и могут сильно задержать завершение проекта и увеличить трудозатраты на его реализацию. Во избежание такой ситуации следует достигать соглашения с заказчиком по всем перечисленным пунктам еще на стадии инициации проекта.

Если вероятность изменений требований проекта высока, то возможны следующие подходы к реагированию на данный риск:

- переоценка проекта при каждом добавлении или изменении требований;
- интеграционная разработка проекта с периодическим уточнением требований, передача части рисков Заказчику;

- учет в оценках трудоемкости и сроков возможности роста требований, например на . . . % (резервирование риска).

Если у нас в проекте недостаточно квалифицированных специалистов, то снизить последствия этого риска можно за счет следующих действий:

- привлечь экспертов-консультантов на начальных этапах;
- учитывать в оценках трудоемкости издержки на обучение сотрудников;
- уменьшать потери от текучести кадров, привлекая на начальном этапе избыточное число участников.

Для установления открытых и доверительных отношений с заказчиком необходимо предпринимать следующие шаги:

- постоянное взаимодействие по вопросам реализации проекта;
- согласование пользовательских интерфейсов и разработка прототипа продукта;
- периодические поставки текущих версий ПП конечным пользователям для их тестирования и оценки.

К числу наиболее часто забываемых работ при планировании проекта относятся: организация обучения пользователей; уточнение требований; управление конфигурациями; разработка и поддержка скриптов автосборки; разработка автотестов; создание тестовых данных; обработка запросов на изменения.

Ошибки в оценках трудоемкости и сроков проекта связаны с принципиальной невозможностью точно определить эти величины на стадии инициации проекта (оценка трудоемкости имеет погрешность от -50% до +100%). Если не прилагать специальных усилий, этот «дамоклов меч» неопределенности будет висеть над проектом на всем его протяжении.

Не стоит надеяться, что участники проекта будут работать только над одним проектом. Есть множество причин, по которым они не смогут тратить на проект 100% своего времени. К списку наиболее распространенных причин этого относятся: сопровождение действующих систем; повышение квалификации; участие в подготовке технико-коммерческих предложений; участие в презентациях; административная работа; отпуска, праздники, больничные и т. д.

Поэтому следует в первую очередь реализовывать ключевые функциональные требования, архитектурно-значимые решения, создавая «представительный» прототип будущей системы. Прототип позволит проверить и оценить общесистемные свойства будущего продукта: доступность, быстродействие, надежность, масштабируемость и т. д. Проработка ключевых функциональных требований и детальное планирование их реализации позволяет уменьшить разброс начальных оценок, детальное проектирование и разработка прототипа — получить более точные оценки трудоемкости.

Может оказаться так, что по результатам прототипирования уточненные оценки суммарной трудоемкости окажутся неприемлемыми. В этом случае проект придется закрыть досрочно, но потери при этом, будут значительно меньше, чем в случае, если то же самое произойдет, когда проект уже в разы превысит первоначальную оценку трудоемкости. Если с заказчиком не удается найти взаимоприемлемое решение при первоначальной оценке проекта, то разумно попытаться договориться о выполнении проекта в несколько этапов с самостоятельным финансированием.

Мониторинг рисков — это процесс наблюдения и контроля за ходом исполнения принятых в отношении рисков планов и инициирование изменений в проекте, если состояние рисков в соответствующих планах влияет на объемы дополнительных работ, требуемые ресурсы или сроки проекта в целом. Другими словами, мониторинг и управление рисками — это процесс идентификации, анализа и планирования реагирования на новые риски, отслеживания ранее идентифицированных рисков, а также проверки и исполнения операций реагирования на риски и оценка эффективности этих операций.

Мониторинг и управление рисками включают в себя следующие задачи: *пересмотр рисков; аудит рисков; анализ отклонений и трендов*.

Пересмотр рисков должен проводиться регулярно, согласно расписанию. Управление рисками проекта должно быть одним из пунктов повестки дня всех совещаний команды проекта.

Аудит рисков предполагает изучение и предоставление в документальном виде результатов оценки эффективности мероприятий по реагированию на риски, относящиеся к идентифицированным рискам, изучение основных причин их возникновения, а также оценку эффективности процесса управления рисками.

На основании анализа отклонений и трендов проекта можно прогнозировать потенциальные отклонения проекта на момент его завершения по показателям стоимости и расписания. Факты отклонения от базового плана могут указывать на последствия, вызванные как угрозами, так и благоприятными возможностями по завершению проекта. Контроль и анализ трендов может повлечь за собой выбор альтернативных стратегий, принятие корректив, перепланировку проекта для достижения базового плана.

2.3 Организация командной работы над проектом

Процесс разработки ПП имеет свою организационную структуру управления, которая определяет распределение ответственности и полномочий среди участников проекта. К участникам проекта относятся все заинтересованные стороны, которые участвуют в проекте или чьи интересы могут быть затронуты при исполнении или завершении проекта. В большинстве литературных источников выделяют следующий основной состав участников проекта:

инициатор проекта — физическое или юридическое лицо (группа лиц), являющееся автором главной идеи проекта, его предварительного обоснования. В качестве инициатора может выступать любой из участников проекта, но деловая инициатива по осуществлению проекта должна исходить либо от инвестора, либо от заказчика;

инвестор — физическое или юридическое лицо (группа лиц), предоставляющие в любой форме финансовые ресурсы для проекта;

заказчик — будущий владелец и пользователь результатов проекта, физическое или юридическое лицо, заинтересованное в осуществлении проекта и достижении его результатов. Следует учитывать, что заказчик и инвестор проекта не всегда совпадают;

куратор проекта — представитель исполнителя, уполномоченный принимать решение о выделении ресурсов и внесении необходимых изменений в проекте;

руководитель проекта — проект-менеджер, физическое лицо, которому заказчик и инвестор делегируют полномочия по руководству работами по осуществлению проекта. Участники также могут влиять на проект и его результаты поставки;

соисполнители проекта — физические или юридические лица, выполняющие на договорной основе отдельные виды работ по проекту;

команда проекта — формируется в зависимости от потребностей, условий проектирования и организационной структуры выполнения проекта.

В соответствии с методологий Microsoft Solutions Framework в команде проекта [11] предлагается выделить пять функциональных групп специалистов:

Группа разработки требований состоит из следующих специалистов, каждый из которых выполняет свойственные только ему роли:

- бизнес-аналитик — разрабатывает модели предметной области (онтологии);
- архитектор — определяет общее видение продукта, его концепцию, интерфейсы, функционал и ограничения;
- системный аналитик — отвечает за перевод требований к продукту в функциональные требования к ПО;

- специалист по требованиям — документирует и сопровождает требования к продукту на всех этапах жизненного цикла ПП;
- менеджер продукта (функциональный заказчик) — представляет в проекте интересы пользователей продукта.

В *группе управления проектом* роли распределяются следующим образом:

- руководитель проекта отвечает за достижение целей проекта при заданных ограничениях (по срокам, бюджету и содержанию), осуществляет управление и контроль реализации проекта и эффективное использование выделенных ресурсов;
- системный архитектор обеспечивает разработку технической концепции системы, принятие ключевых проектных решений относительно внутреннего устройства программного обеспечения и его технических интерфейсов;
- руководитель группы тестирования определяет цели и стратегии тестирования, обеспечивает управление тестированием.

Группа проектирования и разработки ПП состоит из проектировщиков и программистов и обеспечивает:

- проектирование базы данных системы, компонентов и подсистем в соответствии с общей архитектурой, разработку архитектурно значимых модулей, интерфейса пользователя;
- проектирование, реализацию и отладку отдельных модулей системы.

Группа тестирования в проекте выполняет следующее роли: разработку тестовых сценариев и автоматизированных тестов, тестирование продукта, анализ и документирование результатов.

Участники *группы обеспечения реализации проекта*, как правило, не входят в команду проекта. Они выполняют работы в рамках своей профессиональной деятельности. К этой группе можно отнести следующие проектные роли:

- разработчик документации;
- переводчик;
- дизайнер графического интерфейса;
- разработчик учебных курсов;
- специалист по маркетингу и продажам;
- специалист по инструментальным средствам.

За эффективную реализацию программного продукта в целом отвечает руководитель проекта, основная задача которого состоит в том, чтобы:

- найти нужных людей;
- дать им ту работу, для которой они лучше всего подходят;
- не забывать о мотивации;
- помогать им сплотиться в одну команду и работать так дальше.

При этом необходимо учитывать следующие специфические особенности программиста и как личности, и как профессионального участника команды [13].

1) *Высокая самооценка программиста*. Узкая специализация и высокая профессиональная квалификация в конкретной области часто вызывают завышенную самооценку своих возможностей у сотрудников. В связи с этим у руководителя возникают две проблемы:
   - он должен сам точно представлять реальные возможности своих сотрудников, в противном случае неприятные неожиданности неизбежны;
   - сотруднику с высокой самооценкой трудно что-либо приказать, его необходимо убедить, что бывает непросто, в силу того, что сам руководитель вряд ли может быть авторитетом в той области, в которой сотрудник является узким специалистом.
2) *Невысокая трудовая дисциплина*. Программистов часто трудно заставить приходить на работу вовремя, не опаздывать на совещания, своевременно отчитываться о выполненном задании, посылать отчеты. Это связано с индивидуальным характером труда, возможностью выполнять задания вне стен компании, работать во внерабочее время. Руководителю часто приходится доводить до сознания сотрудников тот факта что они работают в команде и разработка программного обеспечения — всегда коллективная деятельность.
3) *Творческий характер программирования*. Разработка программного обеспечения — творческий процесс, разработчики являются креативными личностями и способны привносить энтузиазм, инициативу и собственные нетривиальные решения в общее дело. При наличии сильной мотивации и ясной цели они, как правило, готовы работать с огромной самоотдачей. Это значит, что управление персоналом в программных проектах следует организовывать *по целям*, а не *по заданиям*.
4) *Высокая мобильность сотрудников*. В современных условиях спрос на квалифицированных программистов существенно превышает предложение. Эта тенденция сохранится в обозримом будущем. Из этого следует, что руководитель должен быть в принципе готов к внезапному уходу из команды (и из компании) любого из сотрудников. Процесс разработки следует организовать так, чтобы подобный уход не вызвал катастрофических последствий для проекта. Здесь необходимо учесть два аспекта: возможность утраты необходимой рабочей силы и возможность безвозвратной потери программного кода. Программисты часто не понимают, что программы, разработанные в рамках проекта организации, им не принадлежат.

Каждый специалист при работе над проектом в составе единой команды исполняет как формальные функциональные обязанности (*функциональные роли*), так и определенные неформальные *командные роли*, определяющие его статус и положение в команде.

Распределение функциональных ролей в команде руководитель производит с учетом профессиональных качеств программиста и его типа личности. Одна из первых попыток классификации людей по *типу личности,* которая берет начало с трудов Гиппократа — была сделана на основе разделения по темпераментам [11]:

- *Холерик* — имеет самые скоростные темпоритмы, много и быстро говорит, без промедления отвечает собеседнику, часто перебивает, когда собеседник только начал о чем-то говорить, холерик уже все понял и имеет готовый ответ.
- *Флегматик* — спокойный, миролюбивый и сдержанный человек, никогда не перебивает собеседника, умеет внимательно выслушать и кивает в знак согласия, у него мягкие и неторопливые движения, негромкий голос.
- *Сангвиник* — деловитый, выносливый и работоспособный человек, с хорошим самоконтролем, нередко трудоголик, любит хорошо зарабатывать и делать карьеру.
- *Меланхолик* — чувствительный, обидчивый и очень ранимый человек, легко расстраивается даже при мелких неудачах, любит жаловаться на судьбу, искренне верит, что самая «тяжелая доля» и «самые тяжкие испытания» из всех возможных выпали именно ему.

Наиболее продуктивными специалистами являются *флегматики*, при этом из них получаются как грамотные и настойчивые в реализации программисты, живущие в реальном мире; они конкретны, точны и практичны, стремятся к специализации, так и успешные руководители проектов, рассматривающие широкий спектр возможностей, абстрагирующиеся от технических деталей, склонные обобщать и теоретизировать. В [11] всех программистов — участников работы над проектом с учетом их квалификации и темперамента предлагается условно разбить на девять типов:

1) *Архитектор* мыслит объектами, посвящая себя без остатка решению бизнес-задач, строит абстракции, проводит анализ систем, после чего переходит к кодированию конкретных решений. Зачастую в высшей степени разумные замыслы архитектора воплощаются им в настолько общем и непонятном коде, что людей, могущих разобраться в нем и продолжить начинание, просто не находится. Архитектор любит набросать структуру программы, с тем чтобы впоследствии передать ее дальнейшее кодирование программистам более «низкой» квалификации.
2) *Конструктивисты* получают удовольствие от процесса написания кода и его результата. При этом с написанием кода они справляются быстро, причем в большинстве случаев ошибок в нем не обнаруживается даже на этапе начального тестирования. Основное внимание программисты этого типа уделяют процессу создания кода, поэтому остальное для них не так уж важно. При модернизации ПП конструктивист начинает судорожно искать новые, «заплаточные», решения, отчего надежность кода может резко снизится.
3) *Художник* как тип программиста сконцентрирован на процессе создания кода и искусном сведении объектов пользовательского интерфейса в одну изящную структуру. Работая с компонентами без видимого интерфейса, художники обнаруживают тенденцию к правильной и логичной организации программы. Недостаток художника в том, что очень часто он затягивает кодирование, вдаваясь в излишние украшения и оптимизацию программы. С другой стороны, если программист не культивирует в себе художника, результаты его деятельности зачастую теряют «изюминку».

4) *Инженер* способен на основе готовых программных компонентов создавать множество объектов и сводить их воедино, так что они прекрасно работают в первой же версии программы. Присущая им тяга к усложнению проявляется лишь тогда, когда речь заходит о создании последующих версий.
5) *Ученый* разрабатывает ПП всегда в соответствии с фундаментальными принципами компьютерных наук. Программисты такого типа очень полезны, когда речь заходит об особо трудных задачах кодирования. У инженеров и ученых есть одна общая черта — те и другие очень любят все усложнять. Отдавая должную оценку глубочайшим познаниям ученых, руководитель проекта не должен допускать их полновластия в вопросах написания кода — иначе могут сорваться сроки выполнения проекта.
6) *Лихач* способен разрабатывать ПП в кротчайшие сроки, забывая о комментариях и соглашениях, об поименовании переменных. Тем не менее созданные им продукты вполне успешно работают. Программисты-лихачи незаменимы, если сроки реализации проекта жестко заданы, а качество его исполнения не играет большой роли.
7) *Минималист* создает программы в виде скромного по объему кода, удовлетворяющего обычно всем функциональным требованиям, объекты выстроены четко и однозначно отображают свое назначение. К сожалению, минималисты, решив поставленную задачу, быстро теряют к ней всякий интерес и при обнаружении в ходе тестирования каких-либо проблем выказывают устойчивое нежелание их исправлять.
8) *Трюкач* постоянно осваивает разные новинки, но результат от этого не улучшается. Полагая, что его функции ограничиваются забавами с разными инструментальными средствами, трюкач отказывается учитывать те аспекты программирования, благодаря которым не затрачиваются в дальнейшем титанические усилия на сопровождение программы.
9) *Любителю* не хватает образования, ему нельзя поручать работу над критически важными приложениями. Тщательно изучив какой-нибудь инструментарий, он возводит себя в ранг хакеров. Единственная причина, по которой любители бросают уютные места в отделах тестирования и поддержки пользователей, заключается в том, что, по их мнению, быть программистом — это очень круто.

Командная роль, которую человек исполняет в команде, в основном зависит от состава и состояния команды, при этом люди, выполняющие одни и те же должностные функции в проекте, могут исполнять разные командные роли [11].

1) *Генератор идей* — независимый сотрудник с развитым воображением, оригинальный мыслитель, который решает сложные задачи, дает жизнь новым идеям, но подобно остальным людям может иметь негативные черты характера — чрезмерно чувствителен к критике. Для успеха генератору идей необходимы конструктивные отношения с руководителем или координатором группы.
2) *Координатор* — обычно формальный лидер группы, руководит и направляет группу на качественное выполнение проекта. Может заранее определить, кто из работников хорош для выполнения необходимых задач. Обыч-

но спокойный, уверенный и распорядительный, однако иногда склонен к излишнему доминированию, и группа становится продолжением его сильного «Я».

3) *Аналитик (критик)* — занимает позицию наблюдателя и критически оценивает состояние проекта, не дает группе двигаться неправильным путем. Осмотрительный, бесстрастный, имеет аналитический склад ума, иногда становится чрезмерно критичным.
4) *Вдохновитель команды* — стремится объединять и вносить гармонию в отношения между членами группы, занимает позицию понимающего чужие проблемы, стремится помочь и сглаживает конфликты. По натуре человек добрый, стремится налаживать неформальные отношения, однако бывает нерешительным в сложных или кризисных ситуациях.
5) *Реализатор* — лояльный и честный сотрудник, хороший организатор, методичный и прагматичный специалист, может преобразовать стратегический план в конкретные функциональные задачи, которые доступны для решения. Однако иногда может быть негибким, непреклонным.
6) *Контролер* — отлично умеет создавать отчеты о работе группы, озабочен точным выполнением взятых обязательств и старается не упускать из виду даже мелких деталей Личным примером заставляет сотрудников придерживаться плана работ над проектом, но может становиться излишне тревожным.
7) *Специалист* — профессионал, самостоятелен, стремится стать экспертом в своей области, обладает высокой профессиональной/технической экспертизой и знаниями, гордится своей работой. Привносит вклад только в узкой сфере своей профессиональной деятельности.

Немаловажную роль при командной работе над проектом играет *мотивация* как фактор сознательного и результативного труда любого сотрудника. Для того чтобы человек качественно и в надлежащие сроки исполнял порученную ему работу, он должен испытывать некую потребность и предполагать, что, выполнив эту работу, он в той или иной степени эту потребность удовлетворит. Именно потребности заставляют людей действовать определенным образом. Поэтому управление мотивами осуществляется, как правило, в двух направлениях:

1) формирование правильных потребностей;
2) формирование правильной оценки степени их удовлетворения.

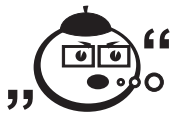
В зависимости от их возраста и опыта работы распределение мотивирующих потребностей у профессиональных разработчиков ПП бывает различным (табл. 2.8).

Для начинающих программистов хорошим стимулом является само участие в успешном проекте, возможность перенимать опыт у более опытных коллег. Для опытных программистов таким стимулом является новизна и востребованность на рынке труда технологий, используемых в проекте, сложность поставленных задач и самостоятельность (потребность самоуважения) в их решении. Для опытного программиста каждая новая задача предоставляет дополнительную возможность доказать свой профессионализм. Пропуск в той или иной графе свидетельствует, что данная потребность не является доминирующей и ее удовлетворение не принесет желаемого результата.

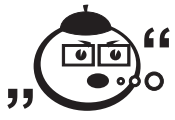
Таблица 2.8 – Зависимость мотивации участника команды от опыта и возраста

Потребности	Профессионализм		
	Начинающий	Опытный	Мастер
Материальные (зарплата, условия труда, социальный пакет)	50%	20%	
Безопасности (стабильность компании, востребованность)		20%	
Принадлежности (возможность учиться у более опытных коллег, опыт участия в успешном проекте, признание в коллективе)	40%	20%	10%
Самоуважения (развиваться, делать что-либо лучше других, повышение в должности, самостоятельность и ответственность в работе)	10%	30%	40%
Самоактуализации (амбициозность целей проекта — сделать то, что никто не делал или не смог сделать)		10%	50%

Ниже приводятся несколько цитат, характеризующих мотивации к труду программистов: [11]

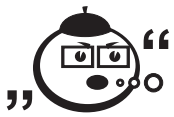


1) Э. Йордан: «Деньги, выгода, комфорт и тому подобное являются факторами «гигиены» — их отсутствие вызывает неудовлетворенность, однако они не могут заставить людей полюбить свою работу и дать им необходимые внутренние стимулы. Что действительно может дать такие стимулы, так это ощущение значительности достигнутых результатов, гордость за хорошо выполненную работу, более высокая ответственность, продвижение по службе и профессиональный рост — все то, что обогащает работу».



2) Программист состоит из четырех компонентов: тело, сердце, разум и душа.

- *Телу необходимы деньги и уверенность в завтрашнем дне.*
- *Сердцу — любовь и признание.*
- *Разуму — развитие и самосовершенствование. Профессионализм в своих глазах и во мнении коллег.*
- *Душе — самореализация [13].*



3) «Программист — это не профессия, это образ мышления». При командной работе над проектом программист должен [11]:

- *Занимать активную позицию, стремиться расширить свою ответственность и увеличивать личный вклад в общее дело.*
- *Постоянно приобретать новые профессиональные знания и опыт, выдвигать новые идеи, направленные на повышение эффективности реализации проекта, добиваться распространения своих знаний, опыта и идей среди коллег.*
- *Получать удовольствие от своей работы, гордиться ее результатами и стремиться, чтобы эти же чувства испытывали все коллеги.*
- *Четко осознавать свои личные и общие цели, понимать их взаимообусловленность, настойчиво стремиться к их достижению.*
- *Быть уверенным в себе и в своих коллегах, объективно оценивать их достижения и успехи, внимательно относиться к их интересам и мнениям, активно искать компромиссные решения в конфликтах.*
- *Всегда оставаться оптимистом, при этом твердо знать, что окружающий мир несовершенен; воспринимать каждую новую проблему, как дополнительную возможность продемонстрировать собственные знания и умения.*

2.4 Практические рекомендации по управлению жизненным циклом разработки программного проекта

Описанные ниже рекомендации предложены и использованы Дж. Маккартни при разработке приложений в среде MS + 4.0 для продукта Microsoft Solution Framework, созданного фирмой Microsoft. Рекомендации объединены в три раздела правил: «Выпустить», «Лучший проект», «Выпустить точно в срок» [15].

Раздел 1. «ВЫПУСТИТЬ»

1) *Вы не знаете того, чего вы не знаете*. В любом проекте всегда существуют неопределенности, но человеку свойственно отвергать свое незнание, подменяя истинное знание предположениями. Неизвестно, что является ошибкой, поэтому старайтесь быть критичными к себе. Найдите для своей команды мотивацию к тому, чтобы она постаралась обнаружить и понять максимум «белых пятен». Не пытайтесь обмануть себя предположениями.

2) *Старайтесь иметь четкое представление о состоянии проекта.* В проекте должны участвовать люди, способные объективно оценить готовность системы. Разработчики всегда более оптимистичны, чем тестировщики. Поощряйте плохие новости, и тогда снизится риск неожиданного провала в самом конце проекта. Самый лучший способ — полагаться на объективные данные, метрики (степень покрытия функциональности тестами, число активных дефектов, динамика их исправления и т. д.). Состояние осведомленности обязательно должно включать знание о всех составляющих проекта и содержать точную информацию о статусе проекта в определенный период времени.

3) *Помните о треугольнике приоритетов.* Существуют три взаимосвязанных компонента любого проекта: ресурсы (люди и деньги), функциональность и сроки. Изменение одного из них всегда влияет на остальные. Можно зафиксировать значения только двух из этих показателей: например, зафиксировать характеристики времени и команды, чтобы получить требуемую функциональность; зафиксировать характеристики времени и команды, чтобы получить нужное время; зафиксировать характеристики времени и команды, чтобы получить требуемую численность команды разработки. Не забывайте, что не все задачи можно выполнять параллельно.

4) *Старайтесь быть на виду.* При планировании надо разбивать задачи на более мелкие, на выполнение которых требуется полдня или день. Тогда о том, что вы начинаете отставать, вы узнаете достаточно рано и сможете предпринять корректирующие шаги. Неделя для проекта разработки — это вечность. Каждый проект, который отстает на месяц, вначале отставал на один день. Узнайте об отставании прежде, чем наступит момент, когда исправить что-то будет уже невозможно. Несомненно, подобный стиль управления весьма опасен и периодически вас будут в нем обвинять. Но если вам удается довести до участников проекта понимание того, что его цель — предоставить готовый продукт в определенное время, ваши коллеги примут этот стиль.

5) *Используйте контрольные точки с отсутствием дефектов.* В каждом приложении можно выделить 20% функциональности, которая будет использоваться 80% времени; в свою очередь, в оставшихся 80% функциональности можно опять выделить 20% и т. д. Реализуйте сначала первые 20% функций, но полностью, а также протестируйте и соберите программу установки с прототипом документации. И только когда все это будет готово, двигайтесь дальше. Этот метод основан на идее выделения контрольных точек, в которых продукт должен содержать некое, небольшое, но четко определенное количество дефектов. Их промежуточный контроль позволяет быстро понять, какие части проекта могут стать причиной проблем, и получить полную картину о проекте, а также дают возможность сфокусироваться на достижении целей каждой контрольной точки.

6) *Бойтесь разработчиков, сидящих в башнях из слоновой кости.* Разработчики должны уметь работать в команде, демонстрировать свой прогресс, делиться опытом и проверять то, что уже сделано. Избегайте «примадонн»,

которые считают себя умнее всех. Несомненно, проект лишь выиграет, если в команде появится пара гениев или просто отличных специалистов, но еще лучше будет, если их талант признает и команда, и все лица, заинтересованные в результатах проекта. Разработка программ осуществляется силами команды, для определения состава которой может пригодиться правило, принятое в Microsoft: шесть разработчиков, три инженера по качеству, один менеджер, два технических писателя. Это правило — результат многочисленных проб и ошибок, через которые прошла корпорация при разработке и масштабных платформ, и совсем небольших утилит.

7) *Плохая дата — не просто плохая дата.* Обычно вы заранее знаете, что опоздаете, знают это и все вокруг. Вы настаиваете на смене даты, но с каждой отсрочкой теряете доверие клиента. Вот хорошее правило: перенос даты должен происходить только в тех случаях, когда точно известны все составляющие и причины задержки. Изменение сроков требует ресурсов, поэтому следующая контрольная точка должна быть реалистичной. В противном случае подобный «проект» никогда не закончится.

8) *Сдвинув сроки, не проваливайте их.* Перенос срока — это симптом, который свидетельствует о выявлении «белых пятен». Такое случается весьма часто, поскольку разработка ИТ-решений является экспериментальным видом деятельности, в котором используются новые технологии. Все это ведет к неопределенности по поводу сроков проекта. Если сдвиг сроков выполнения проекта стал неожиданностью — значит, используемые методы общения с сотрудниками и управления командой проекта надо менять. В то же время задержка дает возможность переоценить ресурсы и возможности продукта; в итоге это приносит положительный эффект. Поэтому, сдвигая сроки, внимательно анализируйте ошибки и старайтесь их больше не допускать.

9) *Чем проще — тем лучше.* Лучше обещать меньше, но сделать больше, чем пообещать больше и не выполнить. Для заказчика важнее результат, а не обещания. А для успеха проекта стоит выбирать максимально простые, но надежные решения.

10) *Время для проектирования — это время для проектирования.* Оценивая способы проектирования, всегда учитывайте временной фактор и выбирайте из альтернативных решений наименее рискованное и оптимальное по времени реализации. Часто в погоне за красотой реализации, проектировщик склонен выбрать вариант, совершенно не укладывающийся в сроки проекта. Помните, что время — весьма ограниченный ресурс, который дан для того, чтобы достичь успеха, а не удивить.

11) *Если вы не можете что-то собрать, значит, вы не сможете это выпустить.* Продукт должен постоянно собираться (компилироваться вместе со всеми компонентами системы, документацией и программой установки). Это необходимо потому, что с самого начала нужно определить все составляющие будущего продукта и адекватно оценить степень их готовности. В противном случае вы обязательно что-нибудь забудете, и это станет весьма неприятной неожиданностью, особенно перед окончанием работ. Пусть

на начальном этапе это будут скромные зачатки будущей системы — придет время, и из них вырастет отличный продукт. С другой стороны, промежуточная версия продукта — хороший способ продемонстрировать заказчику факт готовности того или иного фрагмента работы.

12) *Мысли о многоплатформенности.* Старайтесь «не перебарщивать» с числом платформ, на которых будет работать система. Держите в голове тот факт, что разработка для каждой из платформ зачастую представляет собой переработку большей части функциональности, а также ее тщательное тестирование и последующее исправление ошибок. Соответственно, это увеличивает как стоимость системы, так и затраты на будущую поддержку. Помните, что пользователям чаще всего нужен продукт, а не его удивительная гибкость.

Раздел 2. « ЛУЧШИЙ ПРОДУКТ»

1) *Заказчик — это ваше все.* Старайтесь в следующих версиях учитывать даже весьма странные, нечетко выраженные пожелания клиентов. Часто в этих требованиях скрывается то, что делает продукт уникальным и неотличимым от других, добавляет ему маркетинговой привлекательности и заставляет пользователей использовать его долгие годы. Помните, что заказчик лучше вас знает, что ему нужно.
2) *Самое главное — единство и интеграция.* Единство причины и единство исполнения должны стать девизами команды разработчиков.
3) *Двигайтесь правильным курсом.* Цель — основная идея вашей разработки. Все оценки продукта основываются именно на ней, поэтому она должна быть очень четкой. Старайтесь, как можно раньше наметить цель и сохранить веру в нее вплоть до конца проекта.
4) *Будьте гибким.* Часто по ходу проекта требования к системе могут изменяться — будьте готовы к этому. Старайтесь постоянно проверять, насколько мнение пользователя соответствует поставленной цели. Используйте для этого промежуточные версии продукта, вовлекая заказчика в процесс работы с системой как можно раньше. Однако, собираясь менять курс, помните: цель должна остаться прежней.
5) *Соблюдайте баланс.* Правильно расставьте акценты на разных составляющих проекта. Ни в коем случае не увлекайтесь наращиванием свойств какой-либо из возможностей продукта, это станет причиной дискомфорта пользователей и может привести к серьезным проблемам со сроками реализации продукта.
6) *Развивайте продукт постепенно.* Правильное развитие выглядит так: ранние стадии разработки определяют более поздние, ошибки не повторяются, а результат отвечает потребностям конечного пользователя. Плавное развитие вселяет в вас ощущение предсказуемости и стабильности процесса разработки.
7) *Продукт — это иерархия компонентов.* Следуя этому принципу, элементам проекта уделяют внимание пропорционально их важности, что обеспечи-

вает стабильность и сбалансированное развитие. Иерархию очень удобно использовать как скелет для постепенного расширения системы, сначала вы реализуете основу, а затем наполняете ее будущим содержимым; новые компоненты опираются на уже разработанные.

8) *Все должны разделять общее видение продукта.* Все члены команды должны знать, какие цели должны быть достигнуты, как продукт должен выглядеть, какова стратегия его разработки. Если в команде появляются противники текущей цели, постарайтесь дать им слово и аргументировать свое видение, быть может, оно лишь улучшит будущую систему. Все противоречия должны быть разрешены, а видение предмета приведено к единству.

Раздел 3. «ВЫПУСТИТЬ ТОЧНО В СРОК»

Ваша главная задача — выпустить продукт. Помните:

- команда обязана поставить продукт в срок, а все члены команды должны верить в то, что это возможно;
- каждый должен понимать, что от него для этого требуется, а менеджер должен сделать все от него зависящее, чтобы иметь все шансы сделать то, что требуется;
- любой должен не просто хотеть, а гореть желанием достичь цели.

Выпуск продукта должен стать целью каждого члена команды, самым ожидаемым событием для всех!

Контрольные вопросы по главе 2

1) Приведите возможные определения проекта, его цели, результаты, ограничения.
2) Раскройте смысл «железного треугольника» при управлении программными проектами.
3) Перечислите и прокомментируйте содержание процессов и этапов управления проектами стандарта PMBOK.
4) Приведите основные этапы управления рисками программных проектов.
5) Перечислите и прокомментируйте риски, обусловленные непредвиденными изменениями рыночной ситуации.
6) Перечислите и прокомментируйте риски, обусловленные конкуренцией на рынке.
7) Перечислите и прокомментируйте внутренние риски программного проекта.
8) Перечислите и опишите роли участников проекта.
9) Перечислите и прокомментируйте существующие подходы к выделению функциональных ролевых групп программного проекта.

10) Перечислите и прокомментируйте содержание практических рекомендаций по управлению циклом программного проекта. Раздел: «Выпустить».
11) Перечислите и прокомментируйте содержание практических рекомендаций по управлению циклом программного проекта. Раздел: «Лучший проект».
12) Перечислите и прокомментируйте содержание практических рекомендаций по управлению циклом программного проекта. Раздел: «Выпустить точно в срок».
13) Перечислите и прокомментируйте командные роли участников проекта.
14) Перечислите и прокомментируйте функциональные роли участников проекта.
15) Перечислите и прокомментируйте особенности программиста как участника команды проекта.