Учреждение образования
«Гомельский государственный университет имени Франциска Скорины»

Факультет математики и технологий программирования
Кафедра фундаментальной и прикладной математики

СОГЛАСОВАНО

Заведующий кафедрой
фундаментальной и прикладной
математики
_____ Л.Н.Марченко
30 01 2020 г.

СОГЛАСОВАНО

Декан факультета математики и
технологий программирования
_____ С.П.Жогаль
30 01 2020 г

УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС
ПО ДИСЦИПЛИНЕ

УПРАВЛЕНИЕ ПРОЕКТАМИ В СФЕРЕ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

для специальности
1-40 80 04 Информатика и технологии программирования

Составитель:
Л.Н.Марченко, кандидат технических наук, доцент

Рассмотрено и утверждено
на заседании кафедры фундаментальной
и прикладной математики
30 01 2020 г.,
протокол № 6

Рассмотрено и утверждено
на заседании научно-методического совета университета
26 02 2020 г.,
протокол № 4

Гомель, 2020

ОГЛАВЛЕНИЕ

1 ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к учебно-методическому комплексу дисциплины «УПРАВЛЕНИЕ ПРОЕКТАМИ В СФЕРЕ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ» для специальности 1-40 80 04 Информатика и технологии программирования

Учебно-методический комплекс дисциплины «Управление проектами в сфере информационных технологий» представляет собой комплекс систематизированных учебных, методических и вспомогательных материалов, предназначенных для использования в образовательном процессе специальности 1-40 80 04 «Информатика и технологии программирования». ЭУМК разработан в соответствии со следующими нормативными документами.

1. Положением об учебно-методическом комплексе на уровне высшего образования, утвержденном постановлением Министерства образования Республики Беларусь от 26.07.2011 №167.

2. Требованиями Образовательного стандарта Республики Беларусь ОСВО-40 80 04 2019. Высшее образование. Вторая ступень (магистратура). Специальность 1-40 80 04 «Информатика и технологии программирования».

3. Учебным планом УВО специальности 1-40 80 04 «Информатика и технологии программирования», регистрационный № I 40-20-01/Д-19, дата утверждения 09.04.2019.

4. Учебной программой по учебной дисциплине «Управление проектами в сфере информационных технологий» для специальности 1-40 80 04 «Информатика и технологии программирования», регистрационный № УД 10-2019-231/уч., дата утверждения 22.06.2019.

Данные документы предусматривают получение магистрантами теоретических и практических основ деятельности в области управления проектами в сфере информационных технологий; формирование умений и навыков использования инструментов управления проектами в деятельности субъектов хозяйствования. В них заложена задача формирования профессиональных и личностных компетенций в области экономической деятельности, связанной с созданием и продвижением информационных технологий и продуктов.

В Республике Беларусь возрастает количественная и качественная потребность в подготовке квалифицированных специалистов для существующих ИТ-компаний, в формировании новой предпринимательской среды (возникновение стартапов, инновационного предпринимательства, инфраструктуры коммерциализации и трансфера технологий). Специалистам в сфере информационных технологий необходимы знания и практические навыки в области управления проектами, особенно для продвижения собственных ИТ-продуктов. Изучение дисциплины «Управление проектами в сфере информационных технологий» позволит магистрантам быстрее адаптироваться к ведению бизнеса по своей профессии, принимать более

обоснованные решения по выбору будущего конкретного направления своей деятельности, проанализировать свой предпринимательский потенциал.

Целью учебно-методического комплекса «Управление проектами в сфере информационных технологий» является обеспечение теоретической и практической подготовки елей, активизации их учебно-познавательной деятельности по вопросам управления проектами, развитие профессионально-личностных компетенций в данной сфере, совершенствование умений и навыков самостоятельной учебной работы.

Задачами учебно-методического комплекса являются:

– раскрытие требований к содержанию учебной дисциплины «Управление проектами в сфере информационных технологий»;

– обеспечение успешного усвоение теоретического материала по дисциплине,

– актуализирование использования традиционных форм и методов контроля знаний и стимулировать инновационные подходы к проверке и оценке знаний, умений и навыков магистрантов;

– повышение уровня творческой активности магистрантов через выполнение различных видов творческих заданий и проблемно-поисковых задач.

В структурном отношении учебно-методический комплекс «Управление проектами в сфере информационных технологий» включает в себя четыре раздела: теоретический, практический, раздел контроля знаний, вспомогательный.

Теоретический раздел содержит основные положения, выносимые на лекции и предназначенные как для аудиторной работы с магистрантами, так и для самостоятельного изучения за рамками аудиторных часов. Посредством этих тем магистранты могут получить представление об основных направлениях в области управления проектами; об инструментах менеджера; об управлении рисками проекта, о проблемах формировании команды. При подготовке теоретической части широко использовалась учебная и научная литература по управлению проектами, а также информация из Интернет-ресурсов.

Практический раздел включает в себя в соответствии с учебным планом дисциплины практические занятия. Каждая тема занятия состоит из перечня вопросов для устного обсуждения содержания темы занятия (уровень узнавания), перечня задач (уровень формирования умений и навыков), заданий для письменного контроля знаний (уровень воспроизведения), заданий творческого характера (уровень применения знаний на практике). Дается список основной и дополнительной литературы для самостоятельного изучения и более глубокой подготовки к практическим занятиям. Предполагаются различные формы работы с магистрантами на практических занятиях: устный опрос, защита рефератов, тестовые задания, групповая дискуссия, подготовка презентаций и другие. Особое место занимает использование метода проектов. Магистрантам при разработке проекта предлагалось раскрыть следующие моменты: бизнес-идея; команда, роли,

название; цель, миссия, предмет и объект; маркетинговые исследования в выбранной сфере деятельности; проект и его описание; планирование работ проекта (этапы); представление и защита проекта. Работа над проектами требует от магистрантов дополнительного самообучения, креативного мышления, актуализации полученных знаний из различных областей в решении конкретной задачи. Указанные формы работы способствуют не только усвоению знаний и их репродуктивному воспроизведению, но и видеть закономерности управления проектами.

Раздел контроля знаний представлен критериями оценок результатов учебной деятельности магистрантов по дисциплине. Здесь приводятся вопросы к экзамену, критерии оценивания, примерные тестовые задания, критерий оценивания компьютерного тестирования. В данном разделе размещен также образец тестовых заданий, предназначенных для проверки уровня академических компетенций магистрантов по дисциплине. Они составлены в логической последовательности, и охватывают все темы учебной дисциплины. Представленные тестовые задания являются авторскими, и составлены на основе содержащегося в учебно-методическом комплексе материала к лекциям, а также дополнительных источников, рекомендуемых для самостоятельного изучения. Педагогические тесты включают в себя элемент выборы – варианты для выбора правильных ответов, которые формулируются в форме истинных или ложных высказываний.

Вспомогательный раздел содержит необходимые элементы учебно-программной документации: учебную программу по дисциплине «Управление проектами в сфере информационных технологий» учреждения образования с пояснительной запиской и содержанием учебного материала.

Учебно-методический комплекс учебной дисциплины «Управление проектами в сфере информационных технологий» предназначен для магистрантов 1-го курса специальности 1-40 80 04 «Информатика и технологии программирования» в 1-м семестре дневной формы получения высшего образования (вторая ступень).

Общее количество часов – 90 часов; аудиторное количество часов – 40 часов (3 зачетные единицы), из них лекций – 20 часов (из них управляемая самостоятельная работа студентов 4 часа), практические занятия – 20 часов. Форма отчетности – экзамен (1 семестр).

2 ТЕОРЕТИЧЕСКИЙ РАЗДЕЛ

2.1 Тематические планы лекций

ТЕМА 1 ОСНОВНЫЕ ПОЛОЖЕНИЯ УПРАВЛЕНИЯ ПРОЕКТАМИ

1. История и основные понятия.
2. Отличия программной инженерии от других отраслей.
3. Эволюция подходов к управлению программными проектами.
4. Модели процесса разработки ПО.

ТЕМА 2 УПРАВЛЕНИЕ ПРОЕКТАМИ. ОПРЕДЕЛЕНИЯ И КОНЦЕПЦИИ

1. Проект - основа инноваций.
2. Критерии успешности проекта.
3. Проект и организационная структура компании.
4. Организация проектной команды.
5. Жизненный цикл проекта.
6. Фазы и продукты.

ТЕМА 3 ИНИЦИАЦИЯ ПРОЕКТА

1. Управление приоритетами проектов.
2. Концепция проекта.
3. Цели и результаты проекта.
4. Допущения и ограничения.
5. Ключевые участники и заинтересованные стороны.
6. Ресурсы. Сроки. Риски. Критерии приемки.
7. Обоснование полезности проекта.

ТЕМА 4 ПЛАНИРОВАНИЕ ПРОЕКТА

1. Уточнение содержания и состава работ.
2. Планирование управления содержанием.
3. Планирование организационной структуры.
4. Планирование управления конфигурациям.
5. Планирование управления качеством
6. Базовое расписание проекта.

ТЕМА 5 УПРАВЛЕНИЕ РИСКАМИ ПРОЕКТА

1. Планирование управления рисками.
2. Идентификация рисков.
3. Качественный анализ рисков.
4. Количественный анализ рисков.
5. Планирование реагирования на риски.
6. Главные риски программных проектов и способы реагирования.
7. Управление проектом, направленное на снижение рисков.

8. Мониторинг и контроль рисков.

ТЕМА 6 ОЦЕНКА ТРУДОЕМКОСТИ И СРОКОВ РАЗРАБОТКИ ПО

1. Оценка - вероятностное утверждение.
2. Негативные последствия «агрессивного» расписания.
3. Прагматичный подход.
4. Метод PERT.
5. Обзор метода функциональных точек.
6. Основы методики COCOMO II.

ТЕМА 7 ФОРМИРОВАНИЕ КОМАНДЫ

1. Лидерство и управление.
2. Правильные люди.
3. Мотивация.
4. Эффективное взаимодействие.

ТЕМА 8 РЕАЛИЗАЦИЯ ПРОЕКТА.

1. Рабочее планирование.
2. Принципы количественного управления.
3. Завершение проекта

ТЕМА 9 ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ В ОБЛАСТИ УПРАВЛЕНИЯ ПРОЕКТАМИ

1. Microsoft Project для управления проектами.
2. Составление плана работ.
3. Учет вероятности выполнения работ.
4. Учет стоимости работ.

2.2 Краткий конспект лекций

Тема 1 Основные положения управления проектами

1 История и основные понятия

Программная инженерия есть применение определенного систематического измеримого подхода при разработке, эксплуатации и поддержке программного обеспечения.

Термин software (программное обеспечение, ПО) ввел в 1958 году всемирно известный статистик Джон Тьюкей (John Tukey). Термин software engineering (программная инженерия) впервые появился в названии конференции НАТО, состоявшейся в Германии в 1968 году и посвященной так называемому кризису программного обеспечения. С 1990-го по 1995 год велась работа над международным стандартом, который должен был дать единое представление о процессах разработки программного обеспечения. В результате был выпущен стандарт ISO/IEC 12207 [2]. В 2004 году в отрасли был создан основополагающий труд «Руководство к своду знаний по программной инженерии» (SWEBOK) [3], в котором были собраны основные теоретические и практические знания, накопленные в этой отрасли.

Во избежание двусмысленностей, но, не претендуя на академичность, позволю себе ввести рабочие определения ряда терминов, которые я буду в дальнейшем активно использовать.

Программирование - процесс отображения определенного множества целей на множество машинных команд и данных, интерпретация которых на компьютере или вычислительном комплексе обеспечивает достижение поставленных целей.

Цели могут быть любые: воспроизведение звука в динамике ПК, расчет траектории полета космического аппарата на Марс, печать годового балансового отчета и т.д. Важно то, что они должны быть определены. Это звучит банально, но сколько бы раз об этом не твердили ранее, по-прежнему, приходится сталкиваться с программными проектами, в которых отсутствуют какие-либо определенные цели.

Это отображение может быть очень простым, например, перфорирование машинных команд и данных на перфокартах. А может быть многоступенчатым и очень сложным, когда сначала цели отображаются на требования к системе, требования – на высокоуровневую архитектуру и спецификации компонентов, спецификации - на дизайн компонентов, дизайн - на исходный код. Далее исходный код при помощи компиляторов и сборщиков отображается на код развертывания, код развертывания – на вызовы функций ПО окружения (ОС, промежуточное ПО, базы данных), которое может располагаться на множестве компьютеров, объединенных в сеть, и только после этого – в машинные команды и данные.

Профессиональное программирование (синоним производство программ) – деятельность, направленная на получение доходов при помощи программирования.

Принципиальным отличием от просто программирования является то, что имеется или, по крайней мере, предполагается некоторый потребитель, который готов платить за использование программного продукта. Отсюда следует важный вывод о том, что профессиональное производство программ это всегда коллективная деятельность, в которой участвуют минимум два человека: программист и потребитель

Профессиональный программист – человек, который занимается профессиональным программированием.

Профессионального программиста следует отличать от профессионала (мастера в программировании). Разброс профессионального мастерства в программировании достаточно широк и далеко не каждый, кто зарабатывает на жизнь программированием, является мастером, но об этом позже.

Программный продукт – совокупность программ и сопроводительной документации по их установке, настройке, использованию и доработке.

Согласно стандарту [EEE Std 1074-1995, IEEE Standard for Developing Software Life Cycle Processes] жизненный цикл программы, программной системы, программного продукта включает в себя разработку, развертывание, поддержку и сопровождение. Если программный продукт не коробочный, а достаточно сложный, то его развертывание у клиентов, как правило, реализуется отдельными самостоятельными проектами внедрения. Сопровождение включает в себя устранение критических неисправностей в системе и реализуется часто не как проект а, как процессная деятельность. Поддержка заключается в разработке новой функциональности, переработке уже существующей функциональности, в связи с изменением требований, и улучшением продукта, а также устранение некритических замечаний к ПО, выявленных при его эксплуатации (рисунок 1.1). Жизненный цикл программного продукта завершается выводом продукта из эксплуатации и снятием его с поддержки и сопровождения.

Рисунок 1.1 - Жизненный цикл программного продукта

Процесс разработки ПО – совокупность процессов, обеспечивающих создание и развитие программного обеспечения.

Самый распространенный процесс разработки ПО, который пришлось наблюдать за годы работы в отрасли, можно назвать «как получится». Это не означает, что процесса как такового нет. Он есть и, как правило, обеспечивает разработку ПО при приемлемых затратах и качестве, но этот процесс не документирован, является «знанием стаи», держится на людях и передается из поколения в поколение. Целенаправленная работа по оценке эффективности и улучшению процесса не ведется.

Модель процесса разработки ПО – формализованное представление процесса разработки ПО. Часто при описании процессов вместо слова модель употребляется термин методология, что приводит к неоправданному расширению данного понятия.

Согласно SWEBOK 2004, программная инженерия включает в себя 10 основных и 7 дополнительных областей знаний, на которых базируются процессы разработки ПО. К основным областям знаний относятся следующие области:

1. Software requirements – программные требования.
2. Software design – дизайн (архитектура).
3. Software construction – конструирование программного обеспечения.
4. Software testing – тестирование.
5. Software maintenance – эксплуатация (поддержка) программного обеспечения.
6. Software configuration management – конфигурационное управление.

7. Software engineering management – управление в программной инженерии.
8. Software engineering process – процессы программной инженерии.
9. Software engineering tools and methods – инструменты и методы.
10. Software quality – качество программного обеспечения.

Дополнительные области знаний включают в себя:

1. Computer engineering – разработка компьютеров.
2. Computer science – информатика.
3. Management – общий менеджмент.
4. Mathematics – математика.
5. Project management – управление проектами.
6. Quality management – управление качеством.
7. Systems engineering – системное проектирование.

Все это необходимо знать и уметь применять, для того чтобы разрабатывать ПО. Как видим, управление проектами, о котором мы будем говорить далее, лишь одна из 17 областей знаний программной инженерии, и то вспомогательная. Однако основной причиной большинства провалов программных проектов является именно применение неадекватных методов управления разработкой.

2 Отличия программной инженерии от других отраслей

Standish Group, проанализировав работу сотен американских корпораций и итоги выполнения нескольких десятков тысяч проектов, связанных с разработкой ПО пришла к следующим выводам:

- 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.

То, что производят программисты нематериально – это коллективные мысли и идеи, выраженные на языке программирования. Просто в силу уникальности отрасли опыт профессионалов, накопленный в материальном производстве и изложенный в стандарте PMI PMBOK [«PMBOK. Руководство к Своду знаний по управлению проектами», 3-е изд., PMI, 2004], мало способствует успеху в управлении программным проектом. Управлять разработкой ПО надо иначе.

Творчество - это интеллектуальная деятельность человека, законы которой нам неизвестны. Если бы мы знали законы творчества, то и картины, и стихи, и музыку, и программы уже давно бы создавали компьютеры. Творческое начало это то, что роднит программирование с наукой и искусством.

Творчество в программировании начинается с определения целей программы и заканчивается только тогда, когда в ее коде, написанном на каком-либо языке программирования, поставлена последняя точка. Попытки разделять программистов на творческую элиту, архитекторов и проектировщиков, и нетворческих программистов-кодеров не имеют под собой объективных оснований. Даже если алгоритм программы строго определен математически, два разных программиста его закодируют по-разному, и полученная программа будет иметь разные потребительские качества.

Творчество неразрывно связано с вдохновением, И чем сложнее задача, тем труднее извлечь это вдохновение из подсознания. Иногда для этого требуются часы, а иногда недели.

Программирование - это не искусство, в том смысле, что оно не является творческим отражением и воспроизведением действительности в художественных образах. Об

искусстве в программировании можно и должно говорить только в смысле умения, мастерства, знания дела, как и в любой другой профессии. И как в любой другой профессии программистское мастерство может доставлять истинное эстетическое наслаждение, но только для людей, причастных к этой профессии.

Программирование - это не наука. Наработки математиков в области логики, теории информации, численных методов, реляционной алгебры, теории графов и некоторых других дисциплинах на долю процента не покрывают сложность программистских задач. В программировании нет системы знаний о закономерностях создания программ. Даже выдающиеся программисты не возьмут на себя смелость утверждать об архитектуре новой программной системы то, что она будет успешной. Хотя в программировании уже накоплен определенный опыт провалов, который может позволить искушенному программисту увидеть в архитектуре новой системы антипаттерны - источники серьезных будущих проблем. Но не более того.

Существующее состояние программной инженерии напоминает большую поваренную книгу с многочисленными описаниями рецептов однажды успешно приготовленных блюд из ингредиентов, которых в будущем уже не будет. Завтра в новой системе будут другие вычислительные машины, технологии, языки программирования, инструменты и окружающее ПО, новые проблемы взаимодействия с которыми обязательно придется решать.

Профессиональное творчество программиста принципиально отличается от творчества в науке и искусстве. Программистские задачи с каждым годом становятся все сложнее и объемнее, а сроки, за которые требуется решить эти задачи, наоборот, с каждым годом сокращаются. Поэтому современные программы создаются коллективами от нескольких до тысяч программистов, в то время как творческие деятели науки и искусства работают, как правило, в одиночку.

Программист тоже работает с абстракциями, но ему приходится держать в голове гораздо больше абстракций, чем любому ученому. Абстракции сопутствуют программисту на всех уровнях разработки программы от описания ее целей до исполняемого машинного кода. И этих уровней могут быть десятки. И на каждом уровне абстракций их деталей становится все больше и больше.

Дополнительно к абстрактному мышлению, программист должен обладать сильно выраженным системным мышлением, чтобы удерживать многочисленные взаимосвязи, существующие на всех уровнях программистских абстракций, а также взаимосвязи между этими уровнями. Еще одной сложностью является то, что все эти абстракции и взаимосвязи между ними изменяются во времени, и программист должен учитывать эту динамику.

Кроме того, программист должен обладать маниакальной усидчивостью, сосредоточенностью и упорством для перебора всех возможных вариантов поведения своих абстракций и досконалной проработки всех деталей. Проработка должна быть абсолютно точной и не должна содержать ни одной ошибки, неправильного, лишнего или отсутствующего символа исходного кода (а это порой миллионы строк). Инструменты программирования: синтаксические анализаторы, компиляторы и проч., - лишь незначительно помогают в этой работе.

Еще одна особенность, которая присуща программистскому творчеству, это постоянное обновление информационных технологий, которые программисту необходимо знать и успешно применять в своей работе. Поэтому профессиональный программист должен, как сказал один из наших прежних вождей, «учиться, учиться и учиться». Программист должен удерживать в голове, постоянно пополнять и активно применять на практике гигабайты профессиональной информации. Это устройство компьютеров, компьютерных сетей и сетевые протоколы. Это операционные системы и языки программирования. Это программные интерфейсы промежуточного ПО и прикладных библиотек с особенностями и багами их реализации в конкретных продуктах. Это технологические стандарты, технологии разработки и инструменты, которые их

поддерживают. Это архитектуры программных систем, паттерны и антипаттерны проектирования и много-много другой информации.

Программирование – это проектирование и только проектирование. Роль конструкторского бюро для программного проекта выполняют компилятор и сборщик программ. А программистским аналогом завода, который переводит конструкторскую документацию в продукт, доступный потребителю, служит вычислительный комплекс, на котором развертывается и выполняется созданная программа.

3 Эволюция подходов к управлению программными проектами

За 50 лет развития программной инженерии накопилось большое количество моделей разработки ПО. Интересно провести аналогию между историей развития методов, применяемых в системах автоматического управления летательными аппаратами, и эволюцией подходов к управлению программными проектами.

«Как получится». Разомкнутая система управления. Полное доверие техническим лидерам. Представители бизнеса практически не участвует в проекте. Планирование, если оно и есть, то неформальное и словесное. Время и бюджет, как правило, не контролируются. Аналогия: баллистический полет без обратной связи. Можно, но недалеко и неточно.

«Водопад» или каскадная модель. Жесткое управление с обратной связью. Расчет опорной траектории (план проекта), измерение отклонений, коррекция и возврат на опорную траекторию. Лучше, но не эффективно.

«Гибкое управление». Расчет опорной траектории, измерение отклонений, расчет новой попадающей траектории и коррекция для выхода на нее. «Планы - ничто, планирование - все» (Эйзенхауэр, Дуайт Дэвид)

«Метод частых поставок». Самонаведение. Расчет опорной траектории, измерение отклонений, уточнение цели, расчет новой попадающей траектории и коррекция для выхода на нее.

Классические методы управления перестают работать в случаях, когда структура и свойства управляемого объекта нам не известны и/или изменяются со временем. Эти подходы так же не помогут, если текущие свойства объекта не позволяют ему двигаться с требуемыми характеристиками. Например, летательный аппарат не может развить требуемое ускорение или разрушается при недопустимой перегрузке. Аналогично, если рабочая группа проекта не может обеспечить требуемую эффективность и поэтому постоянно работает в режиме аврала, то это приводит не к росту производительности, а к уходу профессионалов из проекта.

Когда структура и свойства управляемого объекта нам не известны, необходимо использовать адаптивное управление, которое, дополнительно к прямым управляющим воздействиям, направлено на изучение и изменение свойств управляемого объекта. Продолжая аналогию с управлением летательными аппаратами - это расчет опорной траектории, измерение отклонений, уточнение цели, уточнение объекта управления, адаптация (необходимое изменение) объекта управления, расчет новой попадающей траектории и коррекция для выхода на нее.

Для того чтобы понять структуру и свойства объекта и воздействовать на него с целью их приведения к желаемому состоянию, в проекте должен быть дополнительный контур обратной связи – контур адаптации.

4 Модели процесса разработки ПО

Модели (или, как еще любят говорить, методологии) процессов разработки ПО принято классифицировать по «весу» - количеству формализованных процессов

(большинство процессов или только основные) и детальности их регламентации. Чем больше процессов документировано, чем более детально они описаны, тем больше «вес» модели.

ГОСТ 19 «Единая система программной документации» и ГОСТ 34 «Стандарты на разработку и сопровождение автоматизированных систем» ориентированы на последовательный подход к разработке ПО. Разработка в соответствии с этими стандартами проводится по этапам, каждый из которых предполагает выполнение строго определенных работ, и завершается выпуском достаточно большого числа весьма формализованных и обширных документов. Таким образом, строгое следование этим гостам не только приводит к водопадному подходу, но и требует очень высокой степени формализованности разработки. На основе этих стандартов разрабатываются программные системы по госзаказам в России.

SW-CMM

В середине 80-х годов минувшего столетия Министерство обороны США крепко задумалось о том, как выбирать разработчиков ПО при реализации крупномасштабных программных проектов. По заказу военных Институт программной инженерии, входящий в состав Университета Карнеги-Меллона, разработал SW-CMM, Capability Maturity Model for Software в качестве эталонной модели организации разработки программного обеспечения.

Данная модель определяет пять уровней зрелости процесса разработки ПО.

1. Начальный — процесс разработки носит хаотический характер. Определены лишь немногие из процессов, и успех проектов зависит от конкретных исполнителей.

2. Повторяемый — установлены основные процессы управления проектами: отслеживание затрат, сроков и функциональности. Упорядочены некоторые процессы, необходимые для того, чтобы повторить предыдущие достижения на аналогичных проектах.

3. Определенный — процессы разработки ПО и управления проектами описаны и внедрены в единую систему процессов компании. Во всех проектах используется стандартный для организации процесс разработки и поддержки программного обеспечения, адаптированный под конкретный проект.

4. Управляемый — собираются детальные количественные данные по функционированию процессов разработки и качеству конечного продукта. Анализируется значение и динамика этих данных.

5. Оптимизируемый — постоянное улучшение процессов основывается на количественных данных по процессам и на пробном внедрении новых идей и технологий.

Документация с полным описанием SW-CMM занимает около 500 страниц и определяет набор из 312 требований, которым должна соответствовать организация, если она планирует аттестоваться по этому стандарту на 5-ый уровень зрелости.

RUP

Унифицированный процесс (Rational Unified Process, RUP) был разработан Филиппом Крачтеном (Philippe Kruchten), Иваром Якобсоном (Ivar Jacobson) и другими сотрудниками компании "Rational Software" в качестве дополнения к языку моделирования UML.. Модель RUP описывает абстрактный общий процесс, на основе которого организация или проектная команда должна создать конкретный специализированный процесс, ориентированный на ее потребности. Именно эта черта RUP вызывает основную критику - поскольку он может быть чем угодно, его нельзя считать ничем определенным. В результате такого общего построения RUP можно использовать и как основу для самого что ни на есть традиционного водопадного стиля разработки, так и в качестве гибкого процесса.

MSF

Microsoft Solutions Framework (MSF) - это гибкая и достаточно легковесная модель, построенная на основе итеративной разработки. Привлекательной особенностью MSF является большое внимание к созданию эффективной и небюрократизированной проектной команды. Для достижения этой цели MSF предлагает достаточно нестандартные подходы к организационной структуре, распределению ответственности и принципам взаимодействия

внутри команды.

PSP/TSP

Одна из последних разработок Института программной инженерии Personal Software Process / Team Software Process. Personal Software Process определяет требования к компетенциям разработчика. Согласно этой модели каждый программист должен уметь:

- учитывать время, затраченное на работу над проектом;
- учитывать найденные дефекты;
- классифицировать типы дефектов;
- оценивать размер задачи;
- осуществлять систематический подход к описанию результатов тестирования;
- планировать программные задачи;
- распределять их по времени и составлять график работы.
- выполнять индивидуальную проверку проекта и архитектуры;
- осуществлять индивидуальную проверку кода;
- выполнять регрессионное тестирование.

Team Software Process делает ставку на самоуправляемые команды численностью 3–20 разработчиков. Команды должны:

- установить собственные цели;
- составить свой процесс и планы;
- отслеживать работу;
- поддерживать мотивацию и максимальную производительность.

Последовательное применение модели PSP/TSP позволяет сделать нормой в организации пятый уровень CMM.

Agile

Основная идея всех гибких моделей заключается в том, что применяемый в разработке ПО процесс должен быть адаптивным. Они декларируют своей высшей ценностью ориентированность на людей и их взаимодействие, а не на процессы и средства. По сути, так называемые, гибкие методологии это не методологии, а набор практик, которые могут позволить (а могут и нет) добиваться эффективной разработки ПО, основываясь на итеративности, инкрементальности, самоуправляемости команды и адаптивности процесса.

Тяжелые и легкие модели производственного процесса имеют свои достоинства и свои недостатки (таблица 1.1).

Таблица 1.1 - Плюсы и минусы тяжелых и легких моделей процессов разработки ПО

Вес модели	**Плюсы**	**Минусы**
Тяжелые	Процессы рассчитаны на среднюю квалификацию исполнителей. Большая специализация исполнителей. Ниже требования к стабильности команды. Отсутствуют ограничения по объему и сложности выполняемых проектов.	Требуют существенной управленческой надстройки. Более длительные стадии анализа и проектирования. Более формализованные коммуникации.
Легкие	Меньше непроизводительных расходов, связанных с управлением проектом, рисками, изменениями, конфигурациями. Упрощенные стадии анализа и проектирования, основной упор на разработку функциональности, совмещение ролей. Неформальные коммуникации.	Эффективность сильно зависит от индивидуальных способностей, требуют более квалифицированной, универсальной и стабильной команды. Объем и сложность выполняемых проектов ограничены.

Алистер Коуберн, один из авторов «Манифеста гибкой разработки ПО» проанализировал очень разные программные проекты, которые выполнялись по разным моделям от совершенно облегченных и «гибких» до тяжелых (CMM-5) за последние 20 лет. Он не обнаружил корреляции между успехом или провалом проектов и моделями процесса разработки, которые применялись в проектах. Отсюда он сделал вывод о том, что эффективность разработки ПО не зависит от модели процесса, а также о том, чт :

 У каждого проекта должна быть своя модель процесса разработки.

 У каждой модели - свое время.

Это означает, что не существует единственного правильного процесса разработки ПО, в каждом новом проекте процесс должен определяться каждый раз заново, в зависимости от проекта, продукта и персонала, в соответствие с «Законом 4-х П» (рисунок 1.2). 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, например, сайта «отдохни.ру». И, наконец, по-разному следует организовывать процесс разработки в команде вчерашних студентов и в команде состоявшихся профессионалов.

Рисунок 1.2 - «Закон 4-х П». Процесс в проекте должен определяться в зависимости от проекта, продукта и персонала

Команда, которая начинала проект, не остается неизменной, она проходит определенные стадии формирования и, как правило, количественно растет по мере развития проекта. Поэтому процесс должен постоянно адаптироваться к этим изменениям. Главный принцип: не люди должны строиться под выбранную модель процесса, а модель процесса должна подстраиваться под конкретную команду, чтобы обеспечить ее наивысшую эффективность.

Что надо делать для успеха программного проекта

Стив Макконнелл в своей книге приводит тест программного проекта на выживание. Этот чек-лист из 33-х пунктов, который я считаю необходимым процитировать с небольшими корректировками. Руководитель программного проекта должен его

периодически использовать для внутреннего аудита своих процессов.

Чтобы программный проект стал успешным, необходимо:

1. Четко ставить цели.

1.1. Концепция определяет ясные недвусмысленные цели

1.2. Все члены команды считают концепцию реалистичной.

1.3. У проекта имеется обоснование экономической эффективности.

1.4. Разработан прототип пользовательского интерфейса.

1.5. Разработана спецификация целевых функций программного продукта.

1.6. С конечными пользователями продукта налажена двухсторонняя связь

2. Определять способ достижения целей.

2.1. Имеется детальный письменный план разработки продукта.

2.2. В список задач проекта включены «второстепенные» задачи (управление конфигурациями, конвертация данных, интеграция с другими системами).

2.3. После каждой фазы проекта обновляется расписание и бюджет.

2.4. Архитектура и проектные решения документированы.

2.5. Имеется план обеспечения качества, определяющий тестирование и рецензирование.

2.6. Определен план многоэтапной поставки продукта.

2.7. В плане учтены обучение, выходные, отпуска, больничные.

2.8. План проекта и расписание одобрен всеми участниками команды.

3. Контролировать и управлять реализацией.

3.1. У проекта есть куратор. Это такой топ-менеджер исполняющей компании, который лично заинтересован в успехе данного проекта.

3.2. У проекта есть менеджер, причем только один!

3.3. В плане проекта определены «бинарные» контрольные точки.

3.4. Все заинтересованные стороны могут получить необходимую информацию о ходе проекта.

3.5. Между руководством и разработчиками установлены доверительные отношения.

3.6. Установлена процедура управления изменениями в проекте.

3.7. Определены лица, ответственные за решение о принятии изменений в проекте.

3.8. План, расписание и статусная информация по проекту доступна каждому участнику.

3.9. Код системы проходит автоматическое рецензирование.

3.10. Применяется система управления дефектами.

4. Анализировать угрозы и противодействовать им.

4.1. Имеется список рисков проекта. Осуществляется его регулярный анализ и обновление.

4.2. Руководитель проекта отслеживает возникновение новых рисков.

4.3. Для каждого подрядчика определено лицо, ответственное за работу с ним.

5. Создавать команду.

5.1. Опыт команды достаточен для выполнения проекта.

5.2. У команды достаточная компетенция в прикладной области.

5.3. В проекте имеется технический лидер.

5.4. Численность персонала достаточна.

5.5. У команды имеется достаточная сплоченность.

5.6. Все участники привержены проекту.

Оценка и интерпретация теста

Оценка: сумма баллов, каждый пункт оценивается от 0 до 3:

 0 – даже не слышали об этом;

 1 – слышали, но пока не применяем;
 2 – применяется частично;
 3 – применяется в полной мере.

Поправочные коэффициенты:
 для малых проектов (до 5 человек) - 1.5;
 для средних (от 5 до 20 человек) – 1.25.

Результат:
 <40 – завершение проекта сомнительно.
 40-59 – средний результат. В ходе проекта следует ожидать серьезные проблемы.
 60-79 – хороший результат. Проект, скорее всего, будет успешным.
 80-89 – отличный результат. Вероятность успеха высока.
 >90 – великолепный результат. 100% шансов на успех.

Этот чек-лист перечисляет, что надо делать для успеха программного проекта, но не дает ответ на вопрос как это следует делать.

Выводы

То, что производят программисты нематериально – это коллективные мысли и идеи, выраженные на языке программирования. В силу уникальности отрасли опыт, накопленный в отраслях материального производства, мало способствует успеху в управлении программным проектом. Прямые аналогии с этими отраслями не работают. Управлять разработкой ПО надо иначе.

Не существует единственного правильного процесса разработки ПО. Эффективный производственный процесс должен основываться на итеративности, инкрементальности, самоуправляемости команды и адаптивности. Главный принцип: не люди должны строиться под выбранную модель процесса, а модель процесса должна подстраиваться под конкретную команду, чтобы обеспечить ее наивысшую производительность.

Чтобы программный проект стал успешным, необходимо:

1. Четко ставить цели.
2. Определять способ достижения целей.
3. Контролировать и управлять реализацией.
4. Анализировать угрозы и противодействовать им.
5. Создавать команду.

Тема 2 Управление проектами. определения и концепции

1 Проект - основа инноваций

Классическое управление проектами выделяет два вида организации человеческой деятельности: операционная и проектная.

Операционная деятельность 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. «Если водитель трамвая начнет искать новые пути, жди беды».

Там, где разрабатывается новый продукт, внешние условия и требования к которому постоянно меняются, где применяемые производственные технологии используются впервые, где постоянно требуются поиск новых возможностей, интеллектуальные усилия и творчество, там требуются проекты.

Проект - временное предприятие, предназначенное для создания уникальных продуктов, услуг или результатов.

У операционной и проектной деятельности есть ряд общих характеристик: выполняются людьми, ограничены доступностью ресурсов, планируются, исполняются и управляются. Операционная деятельность и проекты различаются, главным образом, тем, что операционная деятельность – это продолжающийся во времени и повторяющийся процесс, в то время как проекты являются временными и уникальными.

Ограничение по срокам означает, что у любого проекта есть четкое начало и четкое завершение. 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.

Уникальность так же важное отличие проектной деятельности от операционной. Если бы результаты проекта не носили уникальный характер, работу по их достижению можно было бы четко регламентировать, установить производственные нормативы и реализовывать в рамках операционной деятельности (конвейер). Задача проекта – достижение конкретной бизнесцели. Задача операционной деятельности – обеспечение нормального течения бизнеса.

Проект - это средство стратегического развития (рисунок 2.1). Цель – описание того, что мы хотим достичь. Стратегия – констатация того, каким образом мы собираемся эти цели достигать. Проекты преобразуют стратегии в действия, а цели в реальность.

Рисунок 2.1 - Проект – средство стратегического развития

Таким образом, каждая работа, которую выполняет конкретный сотрудник, привязывается к достижению стратегических целей организации.

Проекты объединяются в программы. Программа - ряд связанных друг с другом проектов, управление которыми координируется для достижения преимуществ и степени управляемости, недоступных при управлении ими по отдельности.

Проекты и программы объединяются в портфели. Портфель - набор проектов или программ и других работ, объединенных вместе с целью эффективного управления данными работами для достижения стратегических целей.

Проекты и управление ими существовали всегда. В качестве самостоятельной области знаний управление проектами начало формироваться в начале XX века. В этой дисциплине пока нет единых международных стандартов. Наиболее известные центры компетенции:

 PMI, Project Management Institute, PMBOK - американский национальный стандарт ANSI/PMI 99-001-2004.

 IPMA, International Project Management Association. В России - СОВНЕТ.

Примерно 50 лет назад человечество начало жить в новой общественноэкономической формации, которая называется информационное или постиндустриальное общество. Мы живем в эпоху перемен, глобализации и интеллектуального капитала.

Эпоха перемен. Все в мире стало непрерывно и стремительно изменяться. Изобилие стало причиной острейшей конкуренции. Инновации - неотъемлемый атрибут нашего времени. «Если у вас медленный доступ в Интернет, вы можете навсегда отстать от развития информационных технологий». Практика должна постоянно перестраиваться применительно к новым и новым условиям. Например, Hewlett-Packard получает большую долю прибыли на товарах, которые год назад даже не существовали.

Глобализация. Всеобщая взаимозависимость и взаимосвязанность. Транснациональные компании. Бизнес идет туда, где дешевле рабочая сила. Интернет. Конкуренция без границ. Например, Google. За ночь любой из нас в принципе может создать многомиллионную компанию у себя в гараже. С помощью Интернета вы можете выйти на рынок, на котором более 100 млн. потребителей.

Все решают таланты. Простая мобилизация средств и усилий уже не может обеспечить прогресс. Согласно Ф. Брукса, «Если проект не укладывается в сроки, то добавление рабочей силы задержит его еще больше». Идею богатства теперь связывают не с деньгами, а с людьми, не с финансовым капиталом, а с «человеческим». Рынок труда превращается в рынок независимых специалистов и его участникам все больше известно о возможных вариантах выбора. Работники интеллектуального труда начинают самостоятельно определять себе цену.

Человечеству известны два вида деятельности. Репродуктивная деятельность (труд) является слепком, копией с деятельности другого человека либо копией своей собственной деятельности, освоенной в предшествующем опыте. Такая деятельность, как, например, труд токаря в любом механическом цеху, или рутинная повседневная деятельность менеджера-управленца на уровне раз и навсегда усвоенных технологий. Продуктивная деятельность (творчество) - деятельность, направленная на получение объективно нового или субъективно нового (для данного работника) результата.

Репродуктивная деятельность уходит в прошлое. В постиндустриальном обществе интеллект - основная производственная сила. Сегодня от 70 до 80% всего, что сегодня делается людьми, производится при помощи их интеллекта. В любом товаре, сделанном в США, доля зарплаты составляет 70 процентов. Но это в среднем по всем товарам. Что касается разработки ПО, то почти все, что в этой отрасли производится, создается при помощи интеллекта.

Все меньший объем человеческой деятельности может быть организован в виде повторяющихся операций. Традиционный пример операционной деятельности – это работа бухгалтерии. Но жизнь так стремительно изменяется, что сегодня, по утверждению сведущих людей, подготовка и сдача годового финансового отчета каждый раз реализуется как самостоятельный проект.

Проект это основа инноваций. Сделать то, до чего другие компании еще не додумались, сделать это как можно быстрее, иначе это сделают другие. Предложить потребителю более качественный продукт или такой продукт, потребность в котором потребитель даже не может пока осознать.

2 Критерии успешности проекта

Задача проекта – достижение конкретной бизнес-цели, при соблюдении ограничений «железного треугольника» (рисунок 2.2. Это означает, что ни один из углов треугольника не может быть изменен без оказания влияния на другие. Например, чтобы уменьшить время, потребуется увеличить стоимость и/или сократить содержание.

Рисунок 2.2 - «Железный треугольник» ограничений проекта

Согласно текущей редакции стандарта PMBOK, проект считается успешным, если удовлетворены все требования заказчика и участников проекта. Поэтому у проекта разработки ПО сегодня не три, а четыре фактора успеха:

1. Выполнен в соответствие со спецификациями.
2. Выполнен в срок.
3. Выполнен в пределах бюджета.
4. Каждый участник команды уходил с работы в 18:00 с чувством успеха.

Этот четвертый фактор успеха должен стать воспроизводимым, если предприятие хочет быть эффективным. Для успешного проекта характерно постоянное ощущение его участниками чувства удовлетворения и гордости за результаты своей работы, чувства оптимизма. Нет ничего более гибельного для проекта, чем равнодушие или уныние его участников.

Эффективность это отношение полученного результата к произведенным затратам. Нельзя рассматривать эффективность, исходя только из результативности: чем больше ты производишь, чем больше делаешь, тем выше твоя эффективность. С таким подходом можно «зарезать на ужин курицу, несущую золотые яйца». Затраты не следует путать с инвестициями. Оплата аренды, электроэнергии, коммунальные платежи – затраты. Создание и закрепление эффективной команды - это стратегическое приобретение компании. Обучение участников проекта – инвестиции. Вложение в людей - это увеличение числителя в формуле эффективности. Уход из компании всех профессионалов после проекта, выполненного по принципу «любой ценой», – затраты, причем очень тяжело восполняемые. Нарастающая конкуренция указывает на совершенно четкий тренд в мировой экономике - персонал - это форма инвестиций, активов, которые нужно уметь наращивать, управлять и сохранять. Сегодня люди - это капитал.

Современное предприятие обязано относиться к своим работникам так же, как к своим лучшим клиентам. Главный капитал современной компании – это знания. Большая часть этих знаний неотъемлема от их носителя – человека. Те предприятия, которые этого не поняли, не выживут потому, что не смогут быть эффективными. Сегодня эффективное предприятие – это сервис. Предприятие, с одной стороны, предоставляет услуги и продукты своим клиентам, а с другой, - рабочие места для профессионального персонала. Принципы «Одно предприятие на всю жизнь», «Работай продуктивно, а предприятие о тебе позаботится» - уходят в прошлое. Посмотрите на рынок рабочей силы в ИТ - правила устанавливают профессионалы.

Проект и организационная структура компании

Организационная структура компании отражает ее внутреннее устройство, потоки управляющих воздействий, распределение труда и специфические особенности производства. Функциональная и проектная организации – противоположные полюса, а матричная организация – промежуточные состояния. Нет одной лучшей организационной структуры. Нет смысла противопоставлять функциональные структуры и проектные организации.

Синоним функциональной структуры - иерархическая структура (рисунок 2.3).

Рисунок 2.3 - Функциональная структура

Функциональная структура имеет следующие особенности:

- Сохраняется принцип единоначалия
- Понятные и стабильные условия работы
- Хорошо приспособлены для операционной деятельности.
- Специализация подразделений позволяет накапливать экспертизу.
- Затруднено принятие решений и коммуникации между исполнителями. Осуществляются только через руководство.
- Управление сконцентрировано и держится на компетенции высшего руководства
- Как правило, неэффективен контроль за ходом проекта (нет целостной картины)

Функциональная структура предполагает многоуровневую иерархию. Руководители функциональных подразделений это начальники управлений, начальники подчиненных им служб, отделов, лабораторий, секторов, групп. А еще у каждого начальника есть заместитель и, порой, не один. Примеры: министерства, ведомства, научные институты и предприятия советского периода.

На другом краю спектра организационных структур находится проектная структура (рисунок 2.4).

Рисунок 2.4 - Проектная структура

В чисто проектных организациях:

 Проект организуется как самостоятельное производственное подразделение.

 Персонал на проект набирается по временным контрактам.

 После завершения проекта персонал увольняется.

 Медленный старт.

 Опыт не аккумулируется.

 Команды не сохраняются.

Проектные организации не самые эффективные, но порой единственно возможные для выполнения проектов, которые физически удалённы от исполняющей организации, например, строительство нового нефтепровода.

В разработке ПО наиболее распространена матричная организация. Различают три вида матричной организационной структуры: слабая, сбалансированная и сильная (рисунок 2.4 - рисунок 2.6). Причем, в компаниях, которые занимаются продуктовой разработкой ПО, функциональные подразделения определяются в соответствие с линейкой продуктов. Например, отдел разработки CRM-систем, отдел разработки финансовых систем, отдел разработки дополнительных продуктов.

В компаниях, которые ориентированы в основном на заказную разработку ПО, функциональные подразделения чаще объединяются в соответствие с используемыми информационными технологиями. Например, отдел разработки баз данных, отдел разработки J2EE-приложений, отдел веб-разработок, отделы тестирования, документирования и т.д.

Рисунок 2.4 - Слабая матрица

В слабой матрице роль и полномочия сотрудника, который координирует проект, сильно ограничены. Реальное руководство проектом осуществляет один из функциональных руководителей. Координатор проекта, его еще часто называют «трекер», помогает этому руководителю собирать информацию о статусе выполняемых проектных работ, учитывает затраты, составляет отчеты.

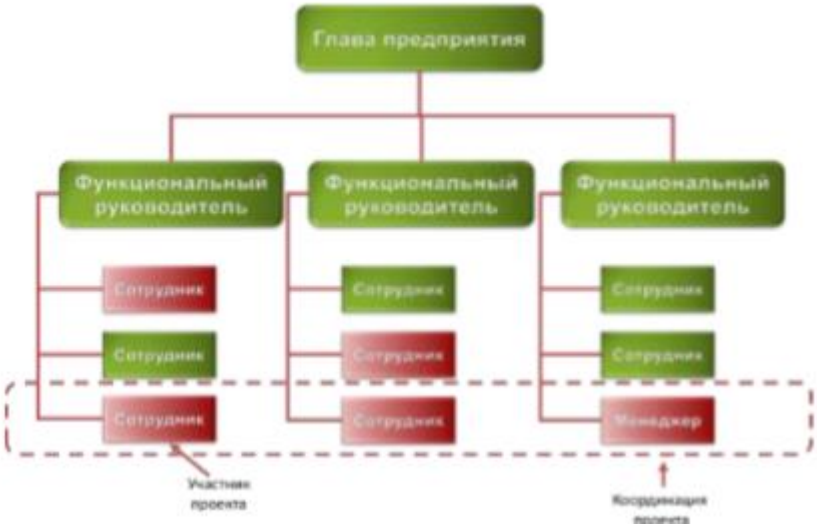

Рисунок 2.5 - Сбалансированная матрица

Сбалансированная матрица характеризуется тем, что появляется менеджер проекта, который реально управляет выделенными на проект ресурсами. Он планирует работы, распределяет задачи среди исполнителей, контролирует сроки и результаты, несет полную ответственность за достижение целей проекта, при соблюдении ограничений.

В сбалансированных матрицах наиболее ярко проявляется проблема двойного подчинения. Руководитель функционального подразделения и менеджер проекта имеют примерно равное влияние на материальный и профессиональный рост разработчиков.

Рисунок2.6 - Сильная матрица

В сильной матрице признается, что проектное управление является самостоятельной областью компетенции, в которой необходимо накапливать экспертизу и использовать общие ресурсы. Поэтому в сильной матрице менеджеры проектов объединяются в самостоятельное функциональное подразделение - офис управления проектами (ОУП). ОУП разрабатывает корпоративные политики и стандарты в области проектного управления, планирует и осуществляет профессиональное развитие менеджеров.

Одной из особенностей матричных структур является то, что они становятся «плоскими», исчезает многоступенчатая иерархия. Предприятие, как правило, делится на

функциональные отделы, в которых работают специалисты разных категорий, напрямую подчиняющиеся начальнику отдела. Начальники лабораторий, секторов, групп упраздняются за ненадобностью. В матричных структурах роль начальника функционального подразделения в производственном процессе заметно снижается, по сравнению с функциональными структурами. В его компетенции остаются вопросы стратегического развития функционального направления, планирование и развитие карьеры сотрудников, вопросы материально-технического обеспечения работ. Следует учитывать, что такое перераспределение полномочий и ответственности от функциональных руководителей к менеджерам проектов часто служит источником конфликтов в компаниях при их переходе от функциональной структуры к матричной.

3 Организация проектной команды

Каждый проект разработки ПО имеет свою организационную структуру, которая определяет распределение ответственности и полномочий среди участников проекта, а также обязанностей и отношений отчетности. Чем меньше проект, тем больше ролей приходится совмещать одному исполнителю.

Роли и ответственности участников типового проекта разработки ПО можно условно разделить на пять групп:

1. Анализ. Извлечение, документирование и сопровождение требований к продукту.
2. Управление. Определение и управление производственными процессами.
3. Производство. Проектирование и разработка ПО.
4. Тестирование. Тестирование ПО.
5. Обеспечение. Производство дополнительных продуктов и услуг.

Группа анализа включает в себя следующие роли:

 Бизнес-аналитик. Построение модели предметной области (онтологии).

 Бизнес-архитектор. Разрабатывает бизнес-концепцию системы. Определяет общее видение продукта, его интерфейсы, поведение и ограничения.

 Системный аналитик. Отвечает за перевод требований к продукту в функциональные требования к ПО.

 Специалист по требованиям. Документирование и сопровождение требований к продукту.

 Менеджер продукта (функциональный заказчик). Представляет в проекте интересы пользователей продукта.

Группа управления состоит из следующих ролей:

 Руководитель проекта. Отвечает за достижение целей проекта при заданных ограничениях (по срокам, бюджету и содержанию), осуществляет операционное управление проектом и выделенными ресурсами.

 Куратор проекта. Оценка планов и исполнения проекта. Выделение ресурсов.

 Системный архитектор. Разработка технической концепции системы. Принятие ключевых проектных решений относительно внутреннего устройства программной системы и её технических интерфейсов.

 Руководитель группы тестирования. Определение целей и стратегии тестирования, управление тестированием.

 Ответственный за управление изменениями, конфигурациями, за сборку и поставку программного продукта.

В производственную группу входят:

 Проектировщик. Проектирование компонентов и подсистем в соответствие с общей архитектурой, разработка архитектурно значимых модулей.

Проектировщик базы данных.

 Проектировщик интерфейса пользователя.

 Разработчик. Проектирование, реализация и отладка отдельных модулей системы.

В большом проекте может быть несколько производственных групп, ответственных за отдельные подсистемы. Как правило, проектировщик выполняет роль лидера группы и управляет своим подпроектом или пакетом работ. Стоит не забывать, что руководитель проекта делегирует полномочия, но не ответственность.

Группа тестирования в проекте состоит из следующих ролей:

 Проектировщик тестов. Разработка тестовых сценариев.

 Разработчик автоматизированных тестов.

 Тестировщик. Тестирование продукта. Анализ и документирование результатов.

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

 Технический писатель.

 Переводчик.

 Дизайнер графического интерфейса.

 Разработчик учебных курсов, тренер.

 Участник рецензирования.

 Продажи и маркетинг.

 Системный администратор.

 Технолог.

 Специалист по инструментальным средствам.

 Другие.

В зависимости от масштаба проекта одну роль могут исполнять несколько человек. Например, разработчики, тестировщики, технические писатели. Некоторые роли всегда должен исполнять только один человек. Например, Руководитель проекта, Системный архитектор. Один человек может исполнять несколько ролей. Возможны следующие совмещения ролей:

 Руководитель проекта + системный аналитик (+ системный архитектор)

 Системный архитектор + разработчик

 Системный аналитик + проектировщик тестов (+ технический писатель)

Системный аналитик + проектировщик интерфейса пользователя

 Ответственный за управление конфигурациями + ответственный за сборку и поставку (+ разработчик)

Крайне нежелательно совмещать следующие роли:

 Разработчик + руководитель проекта

 Разработчик + системный аналитик.

 Разработчик + проектировщик интерфейсов пользователя.

 Разработчик + тестировщик

Не раз приходилось наблюдать, как в критические периоды проекта его менеджер-разработчик с увлечением правит очередные баги, а проектная команда в полном составе стоит у него за спиной и наблюдает за этим процессом. Это плохой пример руководства проектом.

Программисты любят и умеют программировать. Пусть они этим и занимаются. Не стоит загружать программистов несвойственной для них работой. В каждом проекте разработки программного продукта много других работ: бизнес-анализ, проектирование эргономики, графический дизайн, разработка пользовательской документации. Эти работы с программированием не имеют ничего общего. Для них требуются совершенно другая квалификация и другой склад мышления.

При кустарном производстве программ эти задачи, как правило, поручаются программистам, которые это делать не умеют и не любят. Получается обычно плохо, да еще и дорого. В силу своей интроверсии, граничащей с аутизмом, программист просто не в состоянии увидеть свою программу чужими глазами – глазами пользователей. Никто уже не хочет работать с программами с технологической парадигмой навороченного

пользовательского интерфейса - кустарным творением программистов - когда для того чтобы работать с системой, надо обязательно знать, как она устроена. Это типичное творение программиста, которому гораздо важнее видеть, как работает его программа, чем разбираться в том, что она делает для пользователя. Поэтому, необходимо привлекать в проектную команду бизнес-аналитиков, эргономистов, художниковдизайнеров, документалистов. Разделение труда и специализация - залог перехода от кустарного производства к более эффективному промышленному производству.

Из профессиональных программистов получаются отличные тестировщики. Лучшая команда тестирования, которую я встречал, была в Luxoft. Это были маститые программисты из одного академического НИИ с опытом 20-30 лет. Они не осваивали новые программистские технологии, но исключительно эффективно ломали то, что было сделано на их основе. Однако, совмещать одновременно роли программиста и тестировщика – плохая практика. Хороший программист убежден, что он пишет программы правильно и ему психологически тяжело допустить, что где-то в его коде может быть ошибка. А ошибки есть всегда!

Организационная структура проекта обязательно должна включать в себя эффективную систему отчетности, оценки хода выполнения проекта и систему принятия решений. Можно рекомендовать еженедельные собрания по статусу проекта, на которых анализируются риски, оцениваются результаты, достигнутые на предыдущей неделе, и уточняются задачи на новый период.

В модели Scrum рекомендуются ежедневные совещания по состоянию работ – «Stand Up Meeting», но это применимо, скорее, для небольших рабочих групп от 3 до 5 разработчиков. Хотя в критические периоды проекта, приходилось проводить и ежедневные совещания.

Важно помнить, что организационная структура проекта – «живой» организм. Она начинает складываться на стадии планирования и может меняться в ходе проекта. Нестабильность организационной структуры (частые замены исполнителей) – серьезная проблема в управлении сложными программными проектами, поскольку существует время вхождения в контекст проекта, которое может измеряться месяцами.

4 Жизненный цикл проекта. Фазы и продукты

Ранее уже отмечалось, что каждый программный продукт имеет свой жизненный цикл, в который проект разработки очередного релиза входит как одна из фаз. Аналогично, каждый проект разработки ПО имеет свой собственный жизненный цикл, который состоит из четырех фаз (рисунок 2.7).

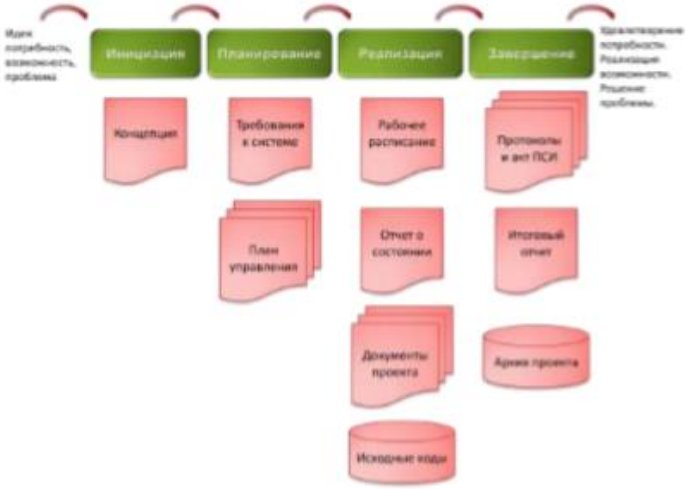

Рисунок 2.7 - Жизненный цикл и основные продукты программного проекта

На фазе инициации проекта необходимо понять, что и зачем мы будем делать – разработать концепцию проекта. Фаза планирования определяет, как мы будем это делать. На фазе реализации происходит материализация наших идей в виде документированного и протестированного программного продукта. И, наконец, на фазе завершения мы должны подтвердить, что мы разработали именно тот продукт, который задумали в концепции проекта, а также провести приемо-сдаточные испытания (ПСИ) продукта на предмет соответствия его свойств, определенным ранее требованиям.

Как правило, редкий проект выполняется в соответствие с первоначальными планами, поэтому важным элементом фазы завершения является «обратная связь»: анализ причин расхождения и усвоение уроков на будущее. Помним, что управляющая система без обратной связи не может быть устойчивой.

Еще одна особенность проекта по сравнению с операционной деятельностью; если в операционной деятельности ресурсы расходуются более-менее равномерно по времени, то в проектном управлении расходование ресурсов в единицу времени имеет явно выраженное колоколообразное распределение (рисунок 2.8).

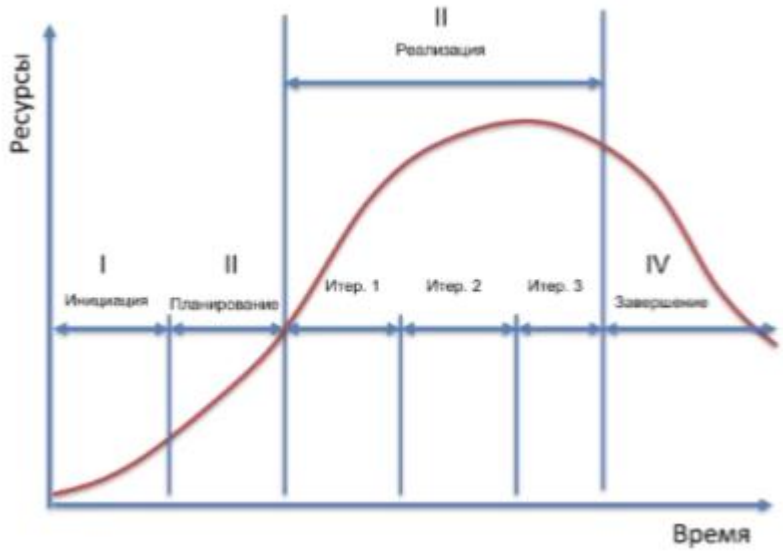

Рисунок 2.8 - Распределение ресурсов по фазам проекта

Проект часто начинается с идеи, которая появляется у одного человека. Постепенно, по мере формулирования, анализа и оценки этой идеи, привлекаются дополнительные специалисты. Еще больше участников требуется на фазе планирования проекта. Пик потребления ресурсов приходится на фазу реализации.

В современных моделях разработки ПО реализация осуществляется на основе сочетания итеративного и инкрементального подходов.

Итеративность предполагает, что требования к системе и ее архитектура прорабатываются не один раз, а постепенно уточняются от итерации к итерации. Это означает, что на каждой итерации происходит полный цикл процессов разработки: уточнение требований, проектирование, кодирование, тестирование и документирование.

Инкрементальность состоит в том, что результатом каждой итерации является версия ПО, которая реализует часть функциональности будущего программного продукта и может быть введена в тестовую или опытную эксплуатацию, а также оценена заказчиком и будущими пользователями. Это означает, что после каждой итерации происходит прирост требуемого функционала, а нереализованных функций будущего продукта остается все меньше.

Сочетание итеративности и инкрементальности обеспечивает эффективность разработки и существенное снижение рисков по ходу проекта. Об этом мы еще будем

говорить.

На последней фазе происходит постепенное высвобождение участников проектной команды. Следует помнить, что проект должен иметь четкое окончание во времени, после которого все работы по проекту закрываются, и на проект перестают тратиться ресурсы. Не должно оставаться «зависших» работ.

Выводы

Проект - этот средство стратегического развития. Цель – описание того, что мы хотим достичь. Стратегия – констатация того, каким образом мы собираемся эти цели достигать. Проекты преобразуют стратегии в действия, а цели в реальность.

Участников типового проекта разработки ПО можно условно разделить на пять групп ролей:

1. Анализ. Извлечение, документирование и сопровождение требований к продукту.
2. Управление. Определение и управление производственными процессами.
3. Производство. Проектирование и разработка ПО.
4. Тестирование. Тестирование ПО.
5. Обеспечение. Производство дополнительных продуктов и услуг.

У программного проекта имеется четыре фактора, которые определяют его успешность:

1. Выполнен в соответствие со спецификациями.
2. Выполнен в срок.
3. Выполнен в пределах бюджета.
4. Каждый участник команды уходил с работы в 18:00 с чувством успеха.

Тема 3 Инициация проекта

1 Управление приоритетами проектов

Эффективные процессы инициации программного проекта минимум наполовину определяют его будущую успешность. Недостаточное внимание именно этой фазе проекта неизбежно приводит к существенным проблемам при планировании, реализации и завершении проекта.

Инициация состоит из процессов, способствующих формальной авторизации начала нового проекта или фазы проекта. Процессы инициации часто выполняются вне рамок проекта и связаны с организационными, программными или портфельными процессами. В ходе процесса инициации уточняются первоначальное описание содержания и ресурсы, которые организация планирует вложить. На этом этапе также выбирается менеджер проекта, если он еще не назначен, и документируются исходные допущения и ограничения. Эта информация заносится в Устав проекта и, если он одобряется, проект официально авторизуется.

Устав проекта - документ, выпущенный инициатором или спонсором проекта, который формально узаконивает существование проекта и предоставляет менеджеру проекта полномочия использовать организационные ресурсы в операциях проекта.

В российской практике данный документ чаще называется Концепция проекта. Концепция (от лат. conceptio — понимание, система), определённый способ понимания, трактовки какого-либо предмета, явления, процесса, основная точка зрения на предмет и др., руководящая идея для их систематического освещения.

В компании, которая принимает решение о старте того или иного проекта разработки ПО, должна существовать единая система критериев для оценки его значимости. Система критериев должна позволять из множества возможных для реализации проектов выбрать наиболее приоритетные для компании.

Приоритет любого проекта должен определяться на основе оценки трех его

характеристик:

 Финансовая ценность.

 Стратегическая ценность.

 Уровень рисков.

Шкала оценки финансовой ценности проекта может выглядеть следующим образом:

 Высокая. Ожидаемая окупаемость до 1 года. Ожидаемые доходы от проекта не менее чем в 1.5 раз превышают расходы. Все допущения при проведении этих оценок четко обоснованны.

 Выше среднего. Ожидаемая окупаемость проекта от 1 года до 3 лет. Ожидаемые доходы от проекта не менее чем в 1.3 раза превышают расходы. Большинство допущений при проведении этих оценок имеют под собой определенные основания.

Средняя. Проект позволяет улучшить эффективность производства в Компании и потенциально может снизить расходы компании не менее чем на 30%. Проект может иметь информационную ценность или помочь лучше контролировать бизнес.

 Низкая. Проект немного снижает расходы компании не менее чем на 10% и дает некоторые улучшения производительности производства.

Например. Финансовая ценность проектов разработки ПО, проектов внедрения или сопровождения, которые выполняются в соответствие с заключенными коммерческими договорами, может быть оценена как высокая. Проект планового развития функциональности продуктов в соответствии с требованиями рынка, инициируемое менеджером продукта на основе анализа предложений отделов маркетинга, консалтинга, продаж и технической поддержки, может получить оценку финансовой ценности выше среднего, а проекты изменения технологических процессов или проекты внутренней автоматизации могут иметь среднюю финансовую ценность.

Одной финансовой ценности для определения приоритета проекта недостаточно. Например, ни одна компания разработчик ПО не возьмется за автоматизацию нелегального оборота наркотиков, если это не соответствует стратегии ее бизнеса. Поэтому, важным показателем приоритета проекта является его соответствие стратегическим целям компании.

Шкала оценки стратегической ценности проекта может иметь следующий вид:

 Высокая. Обеспечивает стратегическое преимущество, дает устойчивое увеличение рынка или позволяет выйти на новый рынок. Решает значительные проблемы, общие для большинства важных клиентов. Повторение конкурентами затруднено или потребует от 1 до 2 лет.

 Выше среднего. Создает временные конкурентные преимущества. Выполнение обязательств перед многими важными клиентами. Конкурентное преимущество может быть удержано в течение 1 года.

 Средняя. Поддерживается доверие рынка к компании. Повышает мнение клиентов о качестве предоставляемых услуг или способствует выполнению обязательств перед несколькими клиентами. Конкуренты уже имеют или способны повторить новые возможности в пределах года.

 Низкая. Стратегическое воздействие отсутствует или незначительно. Влияние на клиентов несущественно. Конкуренты могут легко повторить результаты проекта.

Третьим обязательным показателем приоритета проекта должна быть оценка уровня его риска. Ни один проект, который имеет даже самую высокую оценку финансовой выгодности, не будет запущен в производство, если достижение этой сверхвыгоды имеет минимальные шансы.

Примерная шкала оценки уровня рисков проекта может иметь следующий вид:

 Низкий. Цели проекта и требования хорошо поняты и документированы. Масштаб и рамки проекта заданы четко. Ресурсы требуемой квалификации доступны в полном объеме. Разрабатываемые системы не потребуют новой технологической платформы.

Средний. Цели проекта определены более-менее четко. Хорошее понимание

требований к системе. Масштаб и рамки проекта заданы достаточно хорошо. Ресурсы требуемой квалификации доступны в основном. Системы создаются на новой, но стабильной технологической платформе.

 Выше среднего. Цели проекта недостаточно четки. Задачи системы или бизнес-приложения поняты недостаточно полно. Понимание масштаба и рамок проекта недостаточно. Ресурсы требуемой квалификации сильно ограничены. Системы создаются на новой технологической платформе, сомнения в рыночной стабильности платформы.

 Высокий. Цели проекта нечетки. Основные функциональные компоненты системы не определены. Масштаб и рамки проекта непонятны. Ресурсы требуемой квалификации практически отсутствуют. Системы создаются на новой технологической платформе, в отношении которой крайне мало ясности. Технологии имеют неподтвержденную стабильность.

Если компания уделяет мало внимания управлению приоритетами своих проектов, то это приводит к переизбытку реализуемых проектов, перегруженности исполнителей, постоянным авралам и сверхурочным работам и, как следствие, к низкой эффективности производственной деятельности. При старте нового проекта с высоким приоритетом, компания должна остановить или закрыть менее значимые проекты, чтобы обеспечить новый проект необходимыми ресурсами, а не пытаться сделать все и сразу за счет интенсификации работ, как правило, это не получается.

2 Концепция проекта

У каждого проекта должна быть концепция. Если проект небольшой, то для изложения концепции часто достаточно несколько абзацев. Однако, стартовать проект без концепции, это все равно, что отправлять корабль в плавание, не определив для него пункт назначения.

Концепция проекта разрабатывается на основе анализа потребностей бизнеса. Главная функция документа - подтверждение и согласование единого видения целей, задач и результатов всеми участниками проекта. Концепция определяет что и зачем делается в проекте.

Концепция проекта это ключевой документ, который используется для принятия решений в ходе всего проекта, а также на фазе приемки - для подтверждения результата. Она содержит, как правило, следующие разделы:

Название проекта
Цели проекта
Результаты проекта
Допущения и ограничения
Ключевые участники и заинтересованные стороны
Ресурсы проекта
Сроки
Риски
Критерии приемки
Обоснование полезности проекта

В качестве примера, который позволит иллюстрировать теоретическое изложение основ управления проектами, возьмем реальный проект разработки ПО для автоматизации одного из подразделений крупной производственной компании. Назовем его «Автоматизированная система продажи документации».

Краткая легенда проекта. Заказчик ОАО «XYZ» является одним из ведущих производителей сложных технических изделий. Отдел «123», входящий в ОАО «XYZ», отвечает за продажу дополнительной сопроводительной документации для клиентов ОАО.

Дополнительная документация не входит в стандартную поставку, поскольку владелец этого технического изделия не всегда сам его эксплуатирует, а передает в

эксплуатацию другой компании, которая становится клиентом «XYZ», и закупает у нее эксплуатационную документацию. Ремонт и техобслуживание конкретного изделия может выполнять третья компания, которой уже потребуется детальная техническая документация по ремонту и обслуживанию. Она также становится клиентом «XYZ» и закупает у нее требуемую продукцию.

Основная функция отдела «123» - получение и обработка заказов на дополнительную документацию, согласно ежегодно рассылаемому каталогу. В связи с переездом отдела «123» в новое здание, была поставлена задача на разработку и поставку системы, автоматизирующей основную деятельность отдела «123».

Текст документа Концепция проекта, который будет приводиться в качестве примера, будем выделять цветом фона.

3 Цели и результаты проекта

Цели проекта должны отвечать на вопрос, зачем данный проект нужен. Цели проекта должны описывать бизнес-потребности и задачи, которые решаются в результате исполнения проекта. Целями проекта могут быть:

 Изменения в Компании. Например, автоматизация ряда бизнеспроцессов для повышения эффективности основной производственной деятельности

 Реализация стратегических планов. Например, завоевание значительной доли растущего рынка за счет вывода на него нового продукта.

 Выполнение контрактов. Например, разработка программного обеспечения по заказу.

 Разрешение специфических проблем. Например, доработка программного продукта в целях приведения его в соответствие с изменениями в законодательстве.

Цели должны быть значимыми (направленными на достижение стратегических целей Компании), конкретными (специфичными для данного проекта) измеримыми (т.е иметь проверяемые количественные оценки), реальными (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. Что конкретно будет произведено по окончании проекта.

 Высокоуровневые требования. 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.

Соответствующий раздел документа концепция проекта создания «Автоматизированной системы продажи документации» будет выглядеть следующим образом.

1. Цели и результаты проекта

1.1. Целью проекта является повышение эффективности основной производственной деятельности отдела «123».

1.2. Дополнительными целями проекта являются:

1.2.1. Установление долгосрочных отношений с важным заказчиком ОАО «XYZ».

1.2.2. Выход на новый перспективный рынок современных B2C систем.

2. Результаты проекта должны обеспечить:

2.1. Снижение затрат на обработку заявок.

2.2. Снижение сроков обработки заявок.

2.3. Повышение оперативности доступа к информации о наличии продукции.

2.4. Повышение оперативности доступа к информации о прохождении заявок.

2.5. Повышение надежности и полноты хранения информации о поступивших заявках и результатах их обработки.

3. Продуктами проекта являются:

3.1. Прикладное ПО и документация пользователей.

3.2. Базовое ПО.

3.3. Оборудование ЛВС, рабочие станции, сервера и операционно-системное ПО.

3.4. Проведение пуско-наладочных работ и ввод в опытную эксплуатацию.

3.5. Обучение пользователей и администраторов системы.

3.6. Сопровождение системы на этапе опытной эксплуатации.

3.7. Передача системы в промышленную эксплуатацию.

4. Система должна автоматизировать следующие функции:

4.1. Авторизация и аутентификация пользователей.

4.2. Просмотр каталога продуктов.

4.3. Поиск продуктов по каталогу.

4.4. Заказ выбранных продуктов.

4.5. Просмотр информации о статусе заказа.

4.6. Информирование клиента об изменении статуса заказа.

4.7. Просмотр и обработка заказов исполнителями из службы продаж.

4.8. Просмотр статистики поступления и обработки заказов за период.

4.9. Подготовка и сопровождение каталога продукции

4 Допущения и ограничения

Допущения, как правило, тесно связаны с управлением рисками, о котором мы будем говорить далее. В разработке ПО часто приходится формулировать риски в виде допущений, тем самым передавая его заказчику. Например, оценивая проект разработки и внедрения по схеме с фиксированной ценой, мы должны записать в допущения предположение о том, что стоимость лицензий на стороннее ПО не изменится, до завершения проекта.

Ограничения, как правило, сокращают возможности проектной команды в выборе решений. В частности они могут содержать:

 Специфические нормативные требования. Например, обязательная сертификация продукта, услуги на соответствие определенным стандартам.

 Специфические технические требования. Например, разработка под заданную программно-аппаратную платформу.

 Специфические требования к защите информации.

В этом разделе также уместно сформулировать те требования к системе, которые могут ожидаться заказчиком по умолчанию, но не включаются в рамки данного проекта. Например, в данный раздел может быть включен пункт о том, что разработка программного интерфейса (API) для будущей интеграции с другими системами заказчика не входит в задачи данного проекта.

Содержание этого раздела для нашего проекта-примера выглядит следующим образом.

5. Допущения и ограничения

5.1. Проектирование прикладного ПО выполняется с использованием UML1.

5.2. Средством разработки ПО является Symantec Visual Cafe for Java2.

5.3. В качестве промежуточного ПО сопровождения и поддержки каталога используется ОО БД «Poet»3.

5.4. Нагрузка на систему не должна быть более 100 одновременно работающих пользователей.

5.5. В рамки проекта не входят:

5.5.1. Защита системы от преднамеренного взлома.

5.5.2. Разработка B2B API и интеграция с другими системами.

5 Ключевые участники и заинтересованные стороны

Одна из задач фазы инициации проекта это выявить и описать всех его участников. К участникам проекта относятся все заинтересованные стороны (stakeholders), лица и организации, например заказчики, спонсоры, исполняющая организация, которые активно участвуют в проекте или чьи интересы могут быть затронуты при исполнении или завершении проекта. Участники также могут влиять на проект и его результаты поставки.

К ключевым участникам программного проекта, как правило, относятся:

 Спонсор проекта - лицо или группа лиц, предоставляющая финансовые ресурсы для проекта в любом виде.

 Заказчик проекта - лицо или организация, которые будут использовать продукт, услугу или результат проекта. Следует учитывать, что заказчик и спонсор проекта не всегда совпадают.

 Пользователи результатов проекта.

 Куратор проекта - представитель исполнителя, уполномоченный принимать решение о выделении ресурсов и изменениях в проекте.

 Руководитель проекта - представитель исполнителя, ответственный за реализацию проекта в срок, в пределах бюджета и с заданным качеством.

 Соисполнители проекта. Субподрядчики и поставщики.

Содержание этого раздела в концепции-примере будет иметь вид.

6. Ключевые участники и заинтересованные стороны

6.1. Спонсор проекта - директор Департамента информатизации ОАО «XYZ» В.Васильев.

6.2. Заказчик – начальник Отдела «123» Ф.Федотов

6.3. Пользователи автоматизированной системы:

6.4. Клиенты ОАО «XYZ» (поиск и заказ документации).

6.5. Руководство ОАО «XYZ» (анализ деятельности Отдела «123»).

6.6. Сотрудники производственных департаментов ОАО «XYZ» (сопровождение каталога).

6.7. Сотрудники Отдела «123» (обработка заявок и поставка документации).

6.8. Сотрудники департамента информатизации ОАО «XYZ» (администрирование системы).

6.9. Куратор проекта - начальник отдела заказных разработок И.Иванов.

6.10. Руководитель проекта - ведущий специалист отдела заказных разработок МП П.Петров.

7. Соисполнители:

7.1. Поставщик оборудования и операционно-системного ПО - ООО «Альфа».

7.2. Поставщик базового ПО - ООО «Бета».

6 Ресурсы. Сроки. Риски. Критерии приемки

Для того чтобы понять, сколько будет стоить реализация программного проекта, требуется определить и оценить ресурсы необходимые для его выполнения:

 Людские ресурсы и требования к квалификации персонала.

 Оборудование, услуги, расходные материалы, лицензии на ПО, критические компьютерные ресурсы.

 Бюджет проекта. План расходов и, при необходимости, предполагаемых доходов проекта с разбивкой по статьям и фазам/этапам проекта.

Специфика программного проекта заключается в том, что людские ресурсы вносят основной вклад в его стоимость. Все остальные затраты, как правило, незначительны, по сравнению с этим расходами. О том, как следует подходить к оценкам трудозатрат на реализацию проекта разработки ПО, мы будем подробно говорить в следующих лекциях. На фазе инициации хорошей считается оценка трудозатрат с точностью от -50% до +100%.

Необходимо помнить, что помимо непосредственно программирования в проекте разработки ПО есть много других процессов, которые требуют ресурсы соответствующей квалификации, а само программирование составляет лишь четверть всех затрат. Распределение трудозатрат по основным производственным процессам при современном процессе разработки ПО выглядит в среднем следующим образом (риснок 3.1).

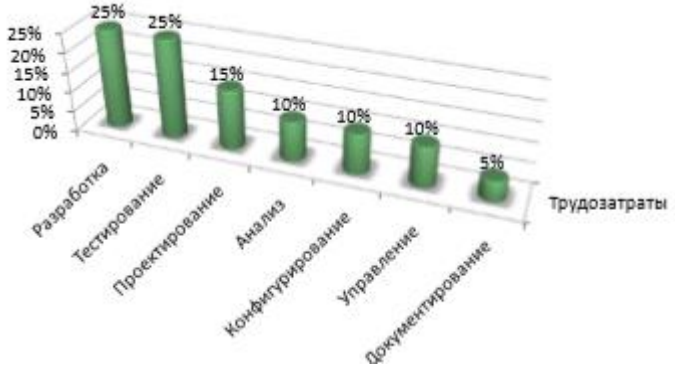

Рисунок 3.1 Распределение трудозатрат по основным производственным процессам при разработке ПО

Поэтому, если по вашей оценки для реализации требуемой функциональности в проекте необходимо написать 10 KSLOC (тысяч строк исходного программного кода), а ваши программисты пишут в среднем по 100 SLOC в день, то общие трудозатраты на проект будут не 100 чел.*дней, а не менее чем 400 чел.*дней. Остальные ресурсы потребуются на анализ и уточнение требований, проектирование, документирование, тестирование и другие проектные работы.

Прежде, чем определять численность и состав проектной команды для нашего примера, нам необходимо сделать оценку трудоемкости разработки ПО. В нашем случае такая экспертная оценка составила с учетом затрат на гарантийное сопровождение на этапе опытной эксплуатации 9000 чел.*час. Исходя из эмпирической кривой Б. Боэма (Рисунок 15), численность команды, близкая к оптимальной, составила 10 человек, из них

8. Ресурсы проекта

8.1. Требования к персоналу

8.1.1. 1 - руководитель проекта,

8.1.2. 1 - технический лидер (архитектура, проектирование),

8.1.3. 1 - системный аналитик (требования, тест-дизайн, документирование),

Трудозатраты 8.1.4. 4 - программисты (с учетом работ по конфигурационному управлению),

8.1.5. 3 - тестировщика.

8.2. Материальные и другие ресурсы

8.2.1. Сервер управления конфигурациями и поддержки системы контроля версий

8.2.2. 2 серверных комплекса (для разработки и тестирования):

8.2.3. Сервер приложений с установленным BEA Weblogic AS

8.2.4. Сервер оперативной БД с установленной Oracle RDBMS

8.2.5. Сервер каталога с установленной OODB “Poet”

8.3. Лицензии на средства разработки и тестирования:

8.3.1. Oracle Designer – 1 лицензия

8.3.2. Symantec Visual Cafe for Java - 5 лицензий.

8.3.3. IBM Rational Test Robot (1 лицензия разработчика + неограниченная лицензия на клиент).

8.4. Расходная часть бюджета проекта4

8.4.1. Разработка и сопровождение прикладного ПО:

8.4.1.1. 9000 чел.*час. * $40 = $360 000

8.4.2. Поставка оборудования и операционно-системного ПО:

8.4.2.1. 3 сервера * $10 000 = $30 000

8.4.3. Поставка базового ПО:

8.4.3.1. BEA Weblogic AS $20 000

8.4.3.2. Oracle RDBMS $20 000

Итого: $430 000

Сроки

Ф. Брукс писал: «Чтобы родить ребенка требуется девять месяцев независимо от того, сколько женщин привлечено к решению данной задачи. Многие задачи программирования относятся к этому типу, поскольку отладка по своей сути носит последовательный характер».

Там же Брукс приводит исключительно полезную, но почему-то редко применяемую, эмпирическую формулу оценки срока проекта по его трудоемкости. Формула была выведена Барии Боэмом (Barry Boehm) на основе анализа результатов 63 проектов разработки ПО, в основном в аэрокосмической области. Согласно этой формуле, для проекта, общая трудоемкость которого составляет N ч.*м. (человеко-месяцев), можно утверждать что:

 Существует оптимальное, с точки зрения затрат, время выполнения графика для первой поставки: T = 2,5 (N ч.*м.)^1/3. То есть оптимальное время в месяцах пропорционально кубическому корню предполагаемого объема работ в человеко-месяцах. Следствием является кривая, дающая оптимальную численность проектной команды (рисунок 3.2).

 Кривая стоимости медленно растет, если запланированный график длиннее оптимального. Работа занимает все отведенное для нее время.

 Кривая стоимости резко растет, если запланированный график короче оптимального. Практически ни один проект невозможно завершить быстрее, чем за ¾ расчетного оптимального графика вне зависимости от количества занятых в нем! (Рисунок 3.3)

Этот примечательный результат дает менеджеру программного проекта солидное подкрепление, когда высшее руководство требует принятия невозможного графика.

Рисунок 3.2 - Закон Б.Боэма

Для сколь-нибудь серьезного программного проекта недостаточно определить только срок его завершения. Необходимо еще определить его этапы – контрольные точки, в которых будет происходить переоценка проекта на основе реально достигнутых показателей.

Контрольная точка - важный момент или событие в расписании проекта, отмечающее достижение заданного результата и/или начало / завершение определенного объема работы. Каждая контрольная точка характеризуется датой и объективными критериями ее достижения.

Современный проект разработки ПО должен реализовываться с применением инкрементального процесса. В этом случае контрольные точки должны соответствовать выпуску каждой промежуточной версии ПО, в которой будет реализована и протестирована определенная часть конечной функциональности программного продукта. В зависимости от сложности и масштаба проекта продолжительность одной итерации может составлять от 2 до 8 недель.

Рисунок 3.3 - Следствия закона Б.Боэма

Соответствующий раздел концепции нашего проекта-примера будет иметь

следующий вид.

9. Сроки проекта

9.1. 03.03 старт

9.2. 28.11 завершение

9.3. Контрольные точки:

9.3.1. 15.04 ТЗ утверждено

9.3.2. 30.04 1-я итерация завершена. Подсистема заказа документации передана в тестовую эксплуатацию (на серверах разработчика).

9.3.3. 15.05 Монтаж оборудования у заказчика завершен .

9.3.4. 30.05 Базовое ПО установлено у заказчика.

9.3.5. 15.06 2-я итерация завершена. Подсистема обработки заказов передана в тестовую эксплуатацию на оборудовании Заказчика

9.3.6. 02.09 3-я итерация завершена. Акт передачи системы в опытную эксплуатацию утвержден

9.3.7. 28.11 Система передана в промышленную эксплуатацию.

Риски

Риск - неопределенное событие или условие, наступление которого отрицательно или положительно сказывается на целях проекта.

Как правило, в случае возникновения негативного риска, почти всегда стоимость проекта увеличивается и происходит задержка в выполнении мероприятий, предусмотренных расписанием проекта. Управлению рисками проекта будет посвящена отдельная лекция.

На этапе инициации, когда нет необходимых данных для проведения детального анализа, часто приходится ограничиваться качественной оценкой общего уровня рисков: низкий, средний, высокий.

В случае нашего проекта-примера раздел «риски» будет выглядеть следующим образом.

10. Риски проекта

10.1. Задачи системы поняты недостаточно полно. Понимание масштаба и рамок проекта недостаточно. Системы создаются на новой технологической платформе, сомнения в рыночной стабильности платформы. Суммарный уровень рисков следует оценить выше среднего.

Критерии приемки

Критерии приемки должны определять числовые значения характеристик системы, которые должны быть продемонстрированы по результатам приемосдаточных испытаний или опытной эксплуатации и однозначно свидетельствовать о достижении целей проекта.

В рассматриваемом примере раздел «Критерии приемки» будет выглядеть следующим образом:

11. Критерии приемки. По итогам опытной эксплуатации система должна продемонстрировать следующие показатели:

11.1. Средние затраты сотрудников Отдела «123» на регламентную обработку одного заказа не превышают 4 чел.*час.

11.2. Срок регламентной обработки 1-го заказа не более 2 недель.

11.3. Время поиска и предоставления информации о наличии дополнительной документации не более 1 мин.

11.4. Время предоставления информации о сделанных заказах и истории их обработки не более 1 мин.

11.5. Система хранит всю информацию о сделанных заказах и истории их обработки.

11.6. Показатель доступности системы 98%.

7 Обоснование полезности проекта

Этот раздел концепции должен содержать краткое технико-экономическое обоснование проекта:

 Для кого предназначены результаты проекта.

 Описание текущей ситуации «As Is». Какие у потенциального заказчика существуют проблемы.

 Каким образом результаты проекта решают эти проблемы («To Be»).

 Насколько значимо для клиента решение данных проблем (оценка экономического эффекта).

 Какие преимущества в итоге из этого может извлечь компанияисполнитель проекта.

Соответствующий раздел в концепции проекта-примера будет иметь следующий вид.

12. Обоснование полезности проекта

12.1. Для Заказчика:

12.1.1. Повышение производительности обработки заказов в 2 раза.

12.1.1.1. “As Is”: 2500 заказов/год по 8 чел.*час.

12.1.1.2. “To Be”: 2500 заказов/год по 4 чел.*час.

12.1.1.3. Экономия: 2500 * 4 * $50 = $500 000 в год.

12.1.2. Повышение оперативности контроля

12.1.2.1. “As Is”: Ежемесячная отчетность.

12.1.2.2. “To Be”: Отчетность on-line.

12.1.3. Повышение удовлетворенности клиентов:

12.1.3.1. Сокращение срока обработки заказа в 2 раза.

12.1.3.2. Сокращение времени на поиск необходимой документации в 10 раз

12.1.3.3. Повышение оперативности обновления каталога 10 раз.

12.2. Для компании-исполнителя:

12.2.1. Высокая стратегическая ценность. Дает устойчивое увеличение рынка и завоевание нового рынка.

12.2.2. Финансовая ценность выше среднего. Ожидаемые доходы от проекта не менее чем в 1.3 раза превышают расходы.

Выводы

Эффективные процессы инициации программного проекта во многом определяют его будущую успешность. Недостаточное внимание этой фазе проекта неизбежно приводит к существенным проблемам при планировании, реализации и завершении.

Концепция проекта это ключевой документ, который используется для принятия решений в ходе всего проекта, а также на фазе приемки - для подтверждения результата.

Приоритет проекта определяется на основе оценки трех показателей:

 Финансовая ценность.

 Стратегическая ценность.

 Уровень рисков.

Тема 4 Планирование проекта

1 Уточнение содержания и состава работ

«Если не получается проглотить слона целиком, то его надо порезать на отбивные». Человечество пока не придумало ничего более эффективного для решения сложной задачи, чем анализ и ее декомпозиция на боле простые подзадачи, которые, в свою очередь, могут быть разделены на еще боле простые подзадачи и так далее. Получается некоторая

иерархическая структура, дерево, в корне которого находится проект, а на листьях элементарные задачи или работы, которые надо выполнить, чтобы завершить проект в условиях заданных ограничений.

Иерархическая структура работ (ИСР) (Work Breakdown Structure, WBS) - ориентированная на результат иерархическая декомпозиция работ, выполняемых командой проекта для достижения целей проекта и необходимых результатов. С ее помощью структурируется и определяется все содержание проекта. Каждый следующий уровень иерархии отражает более детальное определение элементов проекта.

Основой для разработки ИСР служит концепция проекта, которая определяет продукты проекта и их основные характеристики. ИСР обеспечивает выявление всех работ, необходимых для достижения целей проекта. Многие проекты проваливаются не от того, что у них нет плана, а от того что в этом плане забыты важные работы, например, тестирование и исправление ошибок, и продукты проекта, например пользовательская документация. Поэтому, если ИСР составлена корректно, то любая работа, которая в нее не вошла не может считаться работой по проекту.

ИСР делит проект на подпроекты, пакеты работ, подпакеты. Каждый следующий уровень декомпозиции обеспечивает последовательную детализацию содержания проекта, что позволяет производить оценку сроков и объемов работ. ИСР должна включать все промежуточные и конечные продукты.

Выполнять декомпозицию работ проекта можно по-разному. Например, ГОСТ 19.102-77 предусматривает каскадный подход и определяет следующие стадии разработки программной системы:

1. Техническое задание
2. Эскизный проект
3. Технический проект
4. Рабочий проект
5. Внедрение

Если следовать этому стандарту, то на первом уровне ИСР должны находиться именно эти проектные продукты. Если бы пришлось разрабатывать АСУ для управления ядерным реактором или пилотируемым космическим аппаратом, то именно так и следовало поступать. Однако в коммерческой разработке ПО такой подход не эффективен. Как мы уже говорили, современный процесс разработки коммерческого ПО должен быть инкрементальным. Это означает, что на верхнем уровне декомпозиции нашего проекта должны находиться продукты проекта, а на следующем уровне - компоненты, из которых эти продукты состоят. Компоненты далее могут быть декомпозированы на «фичи» - функции, которые они должны реализовывать.

Выделение компонентов, составляющих программный продукт, это элемент высокоуровневого проектирования, которое мы должны выполнить на фазе планирования проекта, не дожидаясь проработки всех функциональных требований к разрабатываемому ПО. Компонентами могут быть как прикладные подсистемы, так и инфраструктурные или ядерные, например, подсистема логирования, безопасности, библиотека визуальных компонентов GUI.

При составлении базового плана работ не стоит стремиться максимально детализировать все работы. ИСР не должна содержать слишком много уровней, достаточно 3-5. Например, ИСР нашего проекта-примера разработки «Автоматизированной системы продажи документации» может выглядеть следующим образом.

Пример: Иерархическая структура работ проекта разработки «Автоматизированной системы продажи документации» (курсивом выделены контрольные точки проекта).

1. Проект разработки «Автоматизированной системы продажи документации»

1.1. Подготовка технического задания на автоматизацию

1.1.1.1. Проведение аналитического обследования

1.1.1.2. Разработка функциональных требований
1.1.1.3. Разработка требований базовому ПО
1.1.1.4. Разработка требований к оборудованию и к операционносистемному ПО
1.1.1.5. Согласование и утверждение ТЗ
1.1.1.6. ТЗ утверждено
1.2. Поставка и монтаж оборудования
1.2.1. Разработка спецификации на оборудование
1.2.2. Закупка и поставка оборудования
1.2.3. Монтаж оборудования
1.2.4. Установка и настройка опреационно-системного ПО
1.2.5. Монтаж оборудования завершен
1.3. Поставка и установка базового ПО
1.3.1. Разработка спецификаций на базовое ПО
1.3.2. Закупка базового ПО
1.3.3. Развертывание и настройка базового ПО
1.3.4. Базовое ПО установлено у заказчика
1.4. Разработка и тестирование прикладного ПО
1.4.1. Разработка спецификаций на прикладное ПО
1.4.2. Установка и конфигурирование рабочей среды
1.4.3. Проектирование и разработка ПО
1.4.3.1. Авторизация и аутентификация пользователей.
1.4.3.2. Разработка подсистемы заказа документации
1.4.3.2.1. Просмотр каталога продуктов.
1.4.3.2.2. Поиск продуктов по каталогу.
1.4.3.2.3. Заказ выбранных продуктов.
1.4.3.2.4. Просмотр информации о статусе заказа.
1.4.3.2.5. Информирование клиента об изменении статуса заказа.
1.4.3.2.6. Подсистема заказа документации передана в тестовую эксплуатацию (на серверах разработчика).
1.4.3.3. Разработка подсистемы обработки заказов
1.4.3.3.1. Просмотр и обработка заказов исполнителями из службы продаж.
1.4.3.3.2. Просмотр статистики поступления и обработки заказов за период.
1.4.3.3.3. Подсистема обработки заказов передана в тестовую эксплуатацию на оборудовании Заказчика
1.4.3.4. Разработка подсистемы сопровождения каталога
1.4.3.4.1. Подготовка и сопровождение каталога продукции.
1.4.3.5. Исправление ошибок
1.4.4. Тестирование ПО
1.4.4.1. Раунд
1 1.4.4.2. Раунд 2
1.4.4.3. Раунд 3
1.4.4.4. Выходное тестирование
1.4.5. Документирование прикладного ПО
1.5. Обучение пользователей
1.5.1. Подготовка учебных курсов
1.5.2. Обучение сотрудников Отдела 123
1.5.3. Обучение руководства ОАО XYZ
1.5.4. Обучение администраторов системы
1.6. Ввод в опытную эксплуатацию
1.6.1. Развертывание и настройка прикладного ПО
1.6.2. Проведение приемо-сдаточных испытаний
1.6.3. Акт передачи системы в опытную эксплуатацию утвержден

1.7. Сопровождение системы в период опытной эксплуатации
1.8. Система передана в промышленную эксплуатацию

Должна быть установлена персональная ответственность за все части проекта (подпроекты и пакеты работ). Для каждого пакета работ должен быть четко определен результат на выходе. Работы и оценки проекта должны быть согласованы с ключевыми участниками команды, руководством компанииисполнителя и, при необходимости, с заказчиком. В результате согласования члены команды принимают на себя обязательства по реализации проекта, а руководство принимает на себя обязательства по обеспечению проекта необходимыми ресурсами.

ИСР является одним из основных инструментов (средств) в механизме управления проектом, с помощью которого измеряется степень достижения результатов проекта. Важнейшая ее функция это обеспечить консистентное представление всех у частников проекта относительно того, как будет делаться проект. В последующем базовый план будет служить ориентиром для сравнения с текущим исполнением проекта и выявления отклонений для целей управления.

2 Планирование управления содержанием

Одна из распространенных «болезней» программных проектов называется «ползучий фичеризм». Это, когда к изначально спроектированной будке для любимой собаки сначала пристраивают сарайчик для хранения садового инвентаря, а потом и домик в несколько этажей для ее хозяина. И все это пытаются построить на одном и том же фундаменте и из тех же самых материалов. Эта болезнь стала причиной летального исхода многих проектов разработки ПО.

Поэтому, сразу, как только удалось стабилизировать и согласовать ИСР, необходимо разработать план управления содержанием проекта. Для этого следует:

 Определить источники запросов на изменение.
 Установить порядок анализа, оценки и утверждения/отклонения изменения содержания.
 Определить порядок документирования изменений содержания.
 Определить порядок информирования об изменении содержания.

Первая задача, которую необходимо решить при анализе запроса на изменения - выявить объекты изменений: требования, архитектура, структуры данных, исходные коды, сценарии тестирования, пользовательская документация, проч. Затем требуется спроектировать и детально описать изменения во всех выявленных объектах. И наконец, следует оценить затраты на внесение изменений, тестирование изменений и регрессионное тестирование продукта и их влияние на сроки проекта.

Эта работа, которая потребует затрат рабочего времени и порой значительных разных специалистов: аналитиков, проектировщиков, разработчиков, тестировщиков, наконец, менеджера проекта. Поэтому эта работа должна обязательно быть учтена в плане.

3 Планирование организационной структуры

Организационная структура - это согласованное и утвержденное распределение ролей, обязанностей и целей деятельности ключевых участников проекта. Она в обязательном порядке должна включать в себя систему рабочих взаимоотношений между рабочими группами проекта, систему отчетности, оценки хода выполнения проекта и систему принятия решений. Следует помнить, что организационная структура проекта – «живой» организм. Она начинает складываться на стадии планирования и должна меняться по ходу проекта.

Нестабильность организационной структуры – частая смена исполнителей - может стать серьезной проблемой в управлении проектом, поскольку, существует цена замены, которая определяется временем вхождения нового участника в контекст проекта.

Планирование управления конфигурациям

Конфигурационное управление один из важных процессов производства программного обеспечения. Об этой области знаний написана не одна книга. Мы будем говорить только о том, что эта работа должна быть спланирована.

План проекта должен включать в себя работы по обеспечению единого хранилища всей проектной документации и разрабатываемого программного кода, обеспечению сохранности и восстановление проектной информации после сбоя. Работы по настройке рабочих станций и серверов, используемых участниками проектной команды, тоже должны войти в план. Кроме этого в плане должны содержаться работы, необходимые для организации сборки промежуточных выпусков системы, а также ее конечного варианта.

Эти работы, как правило, выполняет один человек – инженер по конфигурациям. Если проект небольшой, то эта роль может быть дополнительной для одного из программистов. «Размазывать» эту работу на всех участников проекта, во-первых, неэффективно. Установка и конфигурирование среды разработки, например, баз данных и серверов приложений, требует определенных компетенций и знаний особенностей конкретных версий продуктов. Если эти навыки придется осваивать всем разработчикам, то на это уйдет слишком много рабочего времени. Во-вторых, «размазывание» работ по управлению конфигурациями может привести к коллективной безответственности, когда никто не знает, от чего не собирается проект и как откатиться к консистентной версии.

Управление конфигурациями может многократно усложниться, если проектной команде параллельно с разработкой новой функциональности продукта приходится поддерживать несколько релизов этого продукта, которые были установлены ранее у разных клиентов. Все эти работы должны быть учтены в плане проекта.

4 Планирование управления качеством

Обеспечение качества еще одна из базовых областей знаний в программной инженерии. Относительно того, что такое качество ПО и как его эффективно обеспечивать, можно рассуждать очень и очень долго. В нашем курсе мы ограничимся утверждением о том, что обеспечение качества - это важная работа, которая должна быть спланирована заранее и выполняться по ходу всего программного проекта, а не только во время приемо-сдаточных испытаний.

При планировании этой работы необходимо понимать, что продукт проекта не должен обладать наивысшим возможным качеством, которое недостижимо за конечное время. Необходимое качество продукта определяется требованиями к нему. И еще. Основная задача обеспечения качества - это не поиск ошибок в готовом продукте (выходной контроль) а их предупреждение в процессе производства. Для примера, гладкость обработки детали на токарном станке только случайно может оказаться соответствующей требуемому качеству в 1 микрон, если шпиндель, в котором крепится деталь, плохо центрован.

План управления качеством должен включать в себя следующие работы:

 Объективную проверку соответствия программных продуктов и технологических операций применяемым стандартам, процедурам и требованиям.

 Определение отклонений по качеству, выявление их причин, применение мер по их устранению, а также контроль исполнения принятых мер и их эффективности.

 Представление высшему руководству независимой информации о несоответствиях, не устраняемых на уровне проекта.

Помимо перечисленных разделов план проекта должен включать еще:

 План управления рисками

 Оценку трудоемкости и сроков работ

5 Базовое расписание проекта

После определения трудоемкости работ необходимо определить график их выполнения и общие сроки реализации проекта – составить расписание работ по проекту. Базовое расписание - утвержденный план-график с указанными временными фазами проекта, контрольными точками и элементами иерархической структуры работ.

Базовое расписание может быть наиболее наглядно представлено диаграммой Ганта. В этой диаграмме плановые операции или элементы иерархической структуры работ перечислены с левой стороны, даты отображаются сверху, а длительность операций показана горизонтальными полосками от даты начала до даты завершения.

Базовое расписание это, как правило, элемент контракта с заказчиком. Контрольные точки (вехи) должны служить точками анализа состояния проекта и принятия решения «GO/NOT GO», поэтому они должны зримо демонстрировать статус проекта. Контрольная точка «Проектирование завершено» - плохо. Наиболее эффективный подход – метод последовательных поставок: контрольная точка «Завершено тестирование требований 1, 3, 5, 7»

Если работы не связаны между собой, то любую из них мы можем начинать и завершать, когда нам удобно. Все работы можно делать параллельно и в этом случае минимальная длительность проекта равна длительности самой долгой работы. Однако, на практике между работами существуют зависимости, которые могут быть «жесткими», например, анализ - проектирование – кодирование – тестирование и документирование конкретной функции; или «нежесткими», которые могут пересматриваться или смягчаться. Например, последовательное выполнение задач конкретным исполнителем (можно перепланировать на другого исполнителя) или разработка базового ПО, которая должна предшествовать разработке прикладного ПО. В этом случае можно создавать «заглушки» эмулирующие работу базового ПО. Таким образом, диаграмма Ганта для расписания проекта выглядит как гамак, составленный из множества цепочек взаимосвязанных работ с единой точкой начала и завершения.

Критический путь проекта (Critical path)– самая длинная цепочка работ в проекте. Увеличение длительности любой работы в этой цепочки приводит к увеличению длительности всего проекта.

В проекте всегда существует хотя бы один критический путь, но их может быть несколько. Критический путь может меняться во время исполнения проекта. При исполнении проекта руководитель должен обращать внимание на исполнение задач на критическом пути в первую очередь и следить за появлением других критических путей. Практическая рекомендация: на критическом пути должны стоять работы с нежесткими связями, которые всегда можно перепланировать, если возникает угроза срыва сроков.

Выводы

На верхнем уровне ИСР должны находиться не процессы, а продукты проекта, на следующем уровне - компоненты из которых эти продукты состоят. Выделение компонентов, составляющих программный продукт, это элемент высокоуровневого проектирования, которое мы должны выполнить на фазе планирования проекта, не дожидаясь проработки всех функциональных требований к разрабатываемому ПО.

Помимо работ, непосредственно направленных на создание программного обеспечения, в плане проекта должны быть предусмотрены необходимые ресурсы для обеспечения работ по следующим процессам:

- управление содержанием;
- управление конфигурациями,
- управление качеством,
- управление рисками,

 управление проектом.

В проекте всегда существует хотя бы один критический путь, но их может быть несколько. Критический путь может меняться во время исполнения проекта. При исполнении проекта руководитель должен обращать внимание на исполнение задач на критическом пути в первую очередь и следить за появлением других критических путей

Тема 5 Управление рисками проекта

1 Основные понятия

Согласно Том Демарко: «Проект без риска – удел неудачников. Риски и выгода всегда ходят рука об руку». В первой лекции мы уже говорили о том, что, в силу специфики отрасли, программная инженерия остается и, в ближайшем будущем, будет оставаться производством с высоким уровнем рисков. Если задуматься, то все, что мы делаем, управляя проектом разработки ПО, направлено на борьбу с рисками не уложиться в срок, перерасходовать ресурсы, разработать не тот продукт, который требуется. Определение риска было дано в предыдущей лекции.

Риск - это проблема, которая еще не возникла, а проблема – это риск, который материализовался. Риск характеризуется следующими характеристиками (рисунок 5.1).

 Причина или источник. Явление, обстоятельство обусловливающее наступление риска.

 Симптомы риска, указание на то, что событие риска произошло или вотвот произойдет. Первопричина нам может быть не наблюдаема, например, заразились гриппом. Мы наблюдаем некоторые симптомы – поднялась температура.

 Последствия риска. Проблема или возможность, которая может реализоваться в проекте в результате произошедшего риска.

 Влияние риска. Влияние реализовавшегося риска на возможность достижения целей проекта. Воздействие обычно касается стоимости, графика и технических характеристик разрабатываемого продукта. Многие риски происходят частично и оказывают соразмерное отрицательное или положительное воздействие на проект.

Риск это всегда вероятность и последствия. Например, всегда есть вероятность того, что метеорит упадет на офис центра программных разработок, и это будет иметь катастрофические последствия для проекта. Однако вероятность наступления этого события настолько мала, что мы в большинстве проектов принимаем это риск и не пытаемся им управлять.

Майк Ньюэлл, вице-президент компании PSM Consulting, рассказывал, как он объясняет аудитории на своих лекциях, что такое риск. Он предлагает сыграть в кости на таких условиях, если на кубике выпадает шестерка, то выигрывает он. Если - любое другое число, то выигрывает слушатель. Ставка по 1 доллару. Обычно, большая часть аудитории соглашается сыграть на таких условиях. Майк поднимает ставки: $10, $100, $1000. Постепенно количество желающих поиграть становится все меньше и меньше. При ставке $1000, как правило, желающих рисковать не остается.

Принято выделять две категории рисков:

 «Известные неизвестные». Это те риски, которые можно идентифицировать и подвергнуть анализу. В отношении таких рисков можно спланировать ответные действия.

 «Неизвестные неизвестные». Риски, которые невозможно идентифицировать и, следовательно, спланировать ответные действия.

Рисунок 5.1 - Пример характеристик риска

Неизвестные риски это непредвиденные обстоятельства. Единственное, что мы можем в этом случае предпринять, это создать управленческий резерв бюджета проекта на случай незапланированных, но потенциально возможных изменений. На расходование этого резерва менеджер проекта, как правило, обязан получать одобрение вышестоящего руководства. Управленческие 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.

Девиз разработчиков ПО из Microsoft:«Мы не боремся с рисками — мы ими управляем». Цели управления рисками проекта – снижение вероятности возникновения и/или значимости воздействия неблагоприятных для проекта событий. Адекватное управление рисками в компании – признак зрелости производственных процессов. Том Демарко пишет: «Рассматривать только благоприятные сценарии и встраивать их в план проекта – настоящее ребячество. И все же мы постоянно так поступаем. …Если тех, кто говорит о возможных проблемах до открытия проекта, называют troublemakers, а тех, кто сдает проект спустя 2 месяца после обещанного срока, работая при этом по 6080 часов в неделю, – героями, то у вас плохая команда».

Отказываться от управления проектными рисками это все равно, что в кинотеатре не иметь огнетушителей и плана эвакуации на случай пожара.

2 Планирование управления рисками

Управление рисками это определенная деятельность, которая выполняется в проекте от его начала до завершения. Как и любая другая работа в проекте управление рисками требует времени и затрат ресурсов. Поэтому эта работа обязательно должна планироваться. Планирование управления рисками – это процесс определения подходов и планирования операций по управлению рисками проекта. Тщательное и подробное планирование управления рисками позволяет:

 выделить достаточное количество времени и ресурсов для выполнения операций по управлению рисками,

 определить общие основания для оценки рисков,

 повысить вероятность успешного достижения результатов проекта.

Планирование управления рисками должен быть завершено на ранней стадии

планирования проекта, поскольку оно крайне важно для успешного выполнения других процессов.

Исходными данными для планирования управления рисками служат:

 Отношение к риску и толерантность к риску организаций и лиц, участвующих в проекте, оказывает влияние на план управления проектом. Оно должно быть зафиксировано в изложении основных принципов и подходов к управлению рисками.

 Стандарты организации. Организации могут иметь заранее разработанные подходы к управлению рисками, например категории рисков, общие определение понятий и терминов, стандартные шаблоны, схемы распределения ролей и ответственности, а также определенные уровни полномочий для принятия решений.

 Описание содержания проекта подробно описывает результаты поставки проекта и работы, необходимые для создания этих результатов поставки.

 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 в себя следующие элементы:

 Определение подходов, инструментов и источников данных, которые могут использоваться для управления рисками в данном проекте.

 Распределение ролей и ответственности. Список позиций выполнения, поддержки и управления рисками для каждого вида операций, включенных в план управления рисками, назначение сотрудников на эти позиции и разъяснение их ответственности.

 Выделение ресурсов и оценка стоимости мероприятий, необходимых для управления рисками. Эти данные включаются в базовый план по стоимости проекта.

 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 расписание проекта.

 Категории рисков. Структура, на основании которой производится систематическая и всесторонняя идентификация рисков с нужной степенью детализации. Такую структуру можно разработать с помощью составления иерархической структуры рисков (рисунок 5.2).

 Общие подходы для определения уровней вероятности, шкалы воздействия и близости рисков на проект.

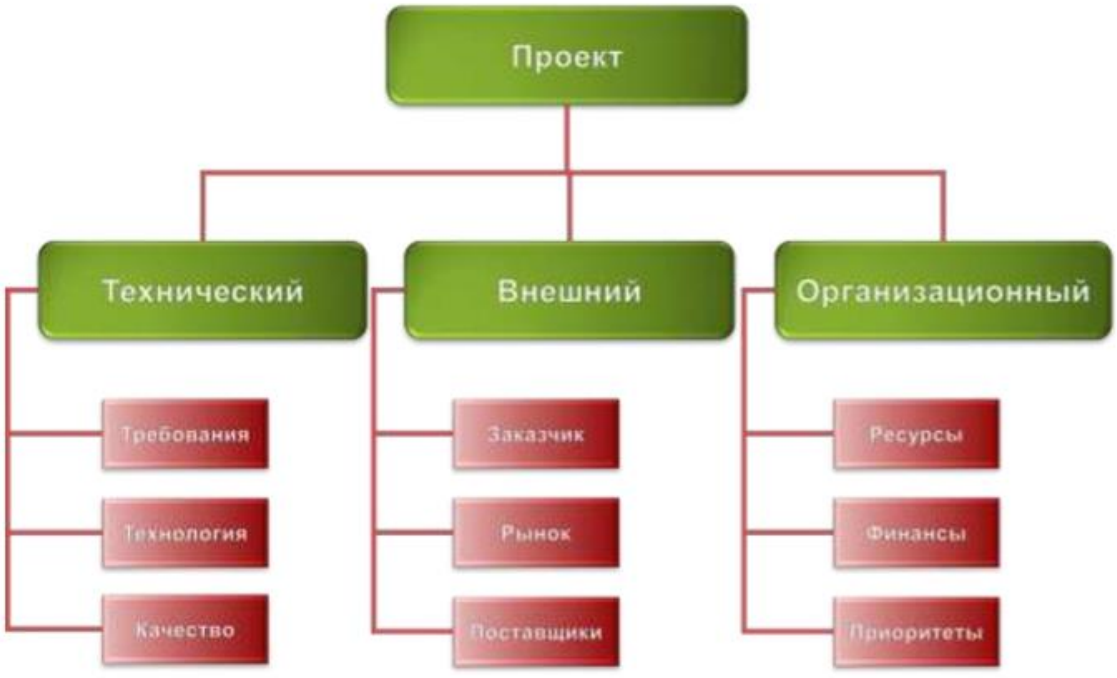

Рисунок 5.2 - Пример иерархической структуры рисков проекта

Шкала оценки воздействия отражает значимость риска (таблица 5.1) в случае его возникновения. Шкала оценки воздействия может различаться в зависимости от потенциально затронутой риском цели, типа и размера проекта, принятыми в организации стратегиями и его финансовым состоянием, а также от чувствительности организации к конкретному виду воздействий.

Таблица 5.1 - Пример шкалы оценки воздействия рисков

Вес	**Значение**	**Критерий**
3	Катастрофические	Потери более $100K
2	Критичные	Потери от $10K до $100K
1	Умеренные	Потери менее $10K

Хотя риск может воздействовать и на сроки проекта, и на качество получаемого продукта, но все эти отклонения могут быть оценены в денежном эквиваленте. Например, последствия задержка по срокам для заказной разработки может быть выражена в сумме денежных санкций, определенных в контракте.

Похожая шкала может быть применена для оценки вероятности наступления риска (таблица 5.2).

Таблица 5.2 - Пример шкалы оценки вероятности осуществления риска

Вес	**Значение**	**Критерий**
3	Очень вероятно	Шансы наступления весьма велики
2	Возможно	Шансы равны
1	Мало вероятно	Наступление события весьма сомнительно

Еще одной важной характеристикой 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. Для шкалы оценки близости риска может быть применена, например, следующая градация: очень скоро, не очень скоро, очень нескоро.

3 Идентификация рисков

Идентификация рисков – это выявление рисков, способных повлиять на проект, и документальное оформление их характеристик. Это итеративный процесс, который периодически повторяется на всем протяжении проекта, поскольку в рамках его жизненного цикла могут обнаруживаться новые риски.

Исходные данные для выявления и описания характеристик рисков могут браться из разных источников.

В первую очередь это база знаний организации. Информация о выполнении прежних проектов может быть доступна в архивах предыдущих 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. Наконец, многие форумы по программированию могут дать бесценную информацию о возникших ранее проблемах в похожих проектах.

Каждый проект задумывается и разрабатывается на основании ряда гипотез,

сценариев и допущений. Как правило, в описании содержания проекта перечисляются принятые допущения - факторы, которые для целей планирования считаются верными, реальными или определенными без привлечения доказательств. Неопределенность в допущениях проекта следует также обязательно рассматривать в качестве потенциального источника возникновения рисков проекта. Анализ допущения позволяет идентифицировать риски проекта, происходящие от неточности, несовместимости или неполноты допущений.

Для сбора информации о рисках могут применяться различные подходы. Среди этих подходов наиболее распространены:

- Опрос экспертов
- Мозговой штурм
- Метод Дельфи
- Карточки Кроуфорда

Цель опроса экспертов – идентифицировать и оценить риски путем интервью подходящих квалифицированных специалистов. Специалисты высказывают своё мнение о рисках и дают им оценку, исходя из своих знаний, опыта и имеющейся информации. Этот метод может помочь избежать повторного наступления на одни и те же грабли.

Перед опросом эксперт должен получить всю необходимую вводную информацию. Деятельность экспертов необходимо направлять, задавая вопросы. Во время опроса вся информация, выдаваемая экспертом, должна записываться и сохраняться. При работе с несколькими экспертами выходная информация обобщается и доводится до сведения всех задействованных экспертов.

К участию в мозговом штурме привлекаются квалифицированные специалисты, которым дают «домашнее задание» - подготовить свои суждения по определенной категории рисков. Затем проводится общее собрание, на котором специалисты по очереди высказывают свои мнения о рисках. Важно: споры и замечания не допускаются. Все риски записываются, группируются по типам и характеристикам, каждому риску дается определение. Цель – составить первичный перечень возможных рисков для последующего отбора и анализа.

Метод Дельфи во многом похож на метод мозгового штурма. Однако есть важные отличия. Во-первых, при применении этого метода эксперты участвуют в опросе анонимно. Поэтому результат характеризуется меньшей субъективностью, меньшей предвзятостью и меньшим влиянием отдельных экспертов. Во-вторых, опрос экспертов проводится в несколько этапов. На каждом этапе модератор рассылает анкеты, собирает и обрабатывает ответы. Результаты опроса рассылаются экспертам снова для уточнения их мнений и оценок. Такой подход позволяет достичь некоего общего мнения специалистов о рисках.

Для быстрого выявления рисков можно воспользоваться еще одной из методик социометрии является известной как "Карточки Кроуфорда" .

Суть этой методики в следующем. Собирается группа экспертов 7-10 человек. Каждому участнику мини-исследования раздается по десять карточек (для этого вполне подойдет обычная бумага для записок). Ведущий задает вопрос: "Какой риск является наиболее важным в этом проекте?" Все респонденты должны записать наиболее, по их мнению, важный риск в данном проекте. При этом никакого обмена мнениями не должно быть. Ведущий делает небольшую паузу, после чего вопрос повторяется. Участник не может повторять в ответе один и тот же риск.

После того как вопрос прозвучит десять раз, в распоряжении ведущего появятся от 70 до 100 карточек с ответами. Если группа подобрана хорошо (в том смысле, что в нее входят люди с различными точками зрения), вероятность того, что участники эксперимента укажут большинство значимых для проекта рисков, весьма высока. Остается составить список названных рисков и раздать его участникам для внесения изменений и дополнений.

В качестве источника информации при выявлении рисков могут служить различные доступные контрольные списки рисков проектов разработки ПО, которые следует проанализировать на применимость к данному конкретному проекту.

Например, Барии Боэм приводит список 10 наиболее распространенных рисков программного проекта:

1. Дефицит специалистов.
2. Нереалистичные сроки и бюджет.
3. Реализация несоответствующей функциональности.
4. Разработка неправильного пользовательского интерфейса.
5. “Золотая сервировка”, перфекционизм, ненужная оптимизация и оттачивание деталей.
6. Непрекращающийся поток изменений.
7. Нехватка информации о внешних компонентах, определяющих окружение системы или вовлеченных в интеграцию.
8. Недостатки в работах, выполняемых внешними (по отношению к проекту) ресурсами.
9. Недостаточная производительность получаемой системы.
10. “Разрыв” в квалификации специалистов разных областей знаний.

Демарко и Листер приводят свой список из пяти наиболее важных источников рисков любого проекта разработки ПО:

1. Изъяны календарного планирования
2. Текучесть кадров
3. Раздувание требований
4. Нарушение спецификаций
5. Низкая производительность

Не существует исчерпывающих контрольных списков рисков программного проекта, поэтому необходимо внимательно анализировать особенности каждого конкретного проекта.

Результатом идентификации рисков должен стать список рисков с описанием их основных характеристик: причины, условия, последствий и ущерба.

Если вернуться к примеру проекта создания «Автоматизированной системы продажи документации», который мы рассматривали в предыдущих лекциях, то список главных выявленных рисков может выглядеть следующим образом

Таблица 5.3 - Список рисков проекта создания «Автоматизированной системы продажи документации»

Причина	**Условия**	**Последствия**	**Ущерб**
Требования не ясны.	Отсутствие описания сценариев использования системы.	Задержка начала разработки прикладного ПО. 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

За процессом идентификации рисков следует процесс их качественного анализа.

4 Качественный анализ рисков

Качественный анализ рисков включает в себя расстановку рангов для идентифицированных рисков. При анализе вероятности и влияния предполагается, что никаких мер по предупреждению рисков не производится.

Качественный анализ рисков включает:

- Определение вероятности реализации рисков.
- Определение тяжести последствий реализации рисков.
- Определения ранга риска по матрице «вероятность - последствия».
- Определение близости наступления риска.
- Оценка качества использованной информации.

Для качественной оценки вероятности реализации риска и определения тяжести последствий его реализации применяется, как правило, общепринятые в организации шкалы, примеры которых мы приводили ранее.

Для определения ранга риска используется матрица вероятностей и последствий (рисунок 5.2). Ранг риска определяется произведением веса вероятности и значимости последствий.

Рисунок 5.2 - Ранг риска и матрица вероятностей и последствий

Могут, конечно, существовать и более сложные шкал для оценок вероятностей, значимости последствий и ранга рисков. Встречались шкалы, которые содержали до 10 градаций. Но, на мой взгляд, наиболее прагматичный подход – это использовать трехуровневое ранжирование.

Продолжая рассмотрение примера проекта создания «Автоматизированной системы продажи документации», матрица рангов главных выявленных рисков может выглядеть следующим образом (таблица 5.4).

Таблица 5.4 - Матрица рангов главных выявленных рисков проекта создания «Автоматизированной системы продажи документации»

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

Для оценки рисков необходима точная и адекватная информация. Использование

неточной информации ведет к ошибкам в оценке. Неверная оценка риска также является риском.

Критерии оценки качества используемой при анализе информации выглядят следующим образом:

- Степень понимания риска.
- Доступность и полнота информации о риске.
- Надежность, целостность и достоверность источников данных.

Результатом качественного анализа рисков является их подробное описание (таблица 5.5).

Таблица 5.5 - 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: Критичная.
Близость : Очень скоро.	Ранг: 6.
Исходные данные: «Содержание проекта», «План обеспечения ресурсами», Протоколы совещаний №21 от 01.06.2008, №27 от 25.06.2008.	

Результаты качественного анализа используются в ходе последующего количественного анализа рисков и планирования реагирования на риски.

5 Количественный анализ рисков

Количественный анализ производится в отношении тех рисков, которые в процессе качественного анализа были квалифицированы как имеющие высокий и средний ранг.

Для количественного анализа рисков могут быть использованы следующие методы:

- Анализ чувствительности.
- Анализ дерева решений.
- Моделирование и имитация.

Анализ чувствительности помогает определить, какие риски обладают наибольшим потенциальным влиянием на проект. В процессе анализа устанавливается, в какой степени неопределенность каждого элемента проекта отражается на исследуемой цели проекта, если остальные неопределенные элементы принимают базовые значения. Результаты представляются, как правило, в виде диаграммы «торнадо». На рисунке 5.3 представлен пример диаграммы, которая отражает влияние на проектные трудозатраты различных факторов профессионализма разработчиков ПО.

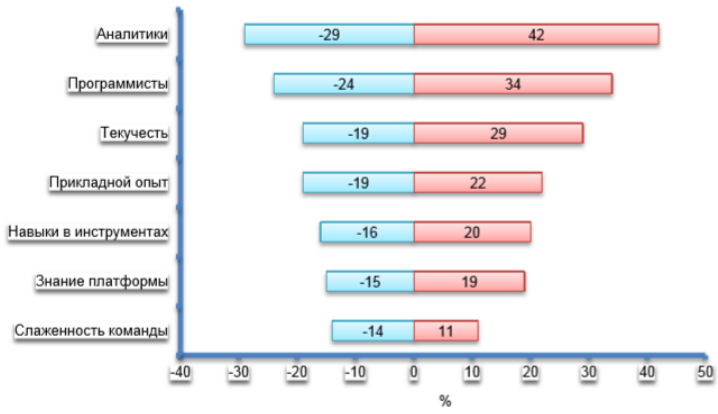

Рисунок 5.3 - Влияние факторов профессионализма разработчиков ПО на трудозатраты по проекту.

Анализ последствий возможных решений проводится на основе изучения диаграммы дерева решений, которая описывает рассматриваемую ситуацию с учетом каждой из имеющихся возможностей выбора и возможного сценария. Рисунок 5.4 представляет пример диаграммы дерева решений на дугах которой проставлены вероятности и затраты при развитии событий по тому или иному сценарию. Критерием для принятия решения служит математическое ожидание потерь от его принятия.

Рисунок 5.4 - Пример анализ дерева решений при выборе покупать или производить необходимую для проекта библиотеку визуальных компонентов (VCL).

При моделировании рисков проекта используется модель для определения последствий от воздействия подробно описанных неопределенностей на результаты проекта в целом. Моделирование обычно проводится с помощью метода Монте-Карло.

Пример подобной модели - система Riskology от Демарко и Листера, который иллюстрирует применение метода Монте-Карло для получения информации о том, какой запас времени будет необходим для того, чтобы преодолеть влияние всех неуправляемых рисков проекта. Модель позволяет учесть пять основных (рисунок 5.5) и пять дополнительных рисков проекта.

Таблица 5.6 - Пять основных факторов риска программного проекта, учитываемые в модели Riskology

НАЗВАНИЕ РИСКА	*ОПИСАНИЕ*	*СТАТУС*
КАЛЕНД. ПЛАН	Изъяны календарного планирования	ВКЛ
ТЕКУЧКА	Текучесть кадров	ВКЛ
РАЗДУВАНИЕ	Раздувание требований	ВКЛ
СПЕЦИФИКАЦИИ	Нарушение спецификаций	ВКЛ
	Низкая производительность	ВКЛ

Характеристики предопределенных в системе Riskology рисков пользователь может изменить, задав значения минимальной, максимальной и наиболее вероятной задержки сроков сдачи проекта из-за влияния данного риска. Можно включить в модель дополнительные собственные риски. Результат моделирования по методу Монте-Карло будет представлен в виде гистограммы распределения срока завершения оцениваемого проекта (рисунок 5.5).

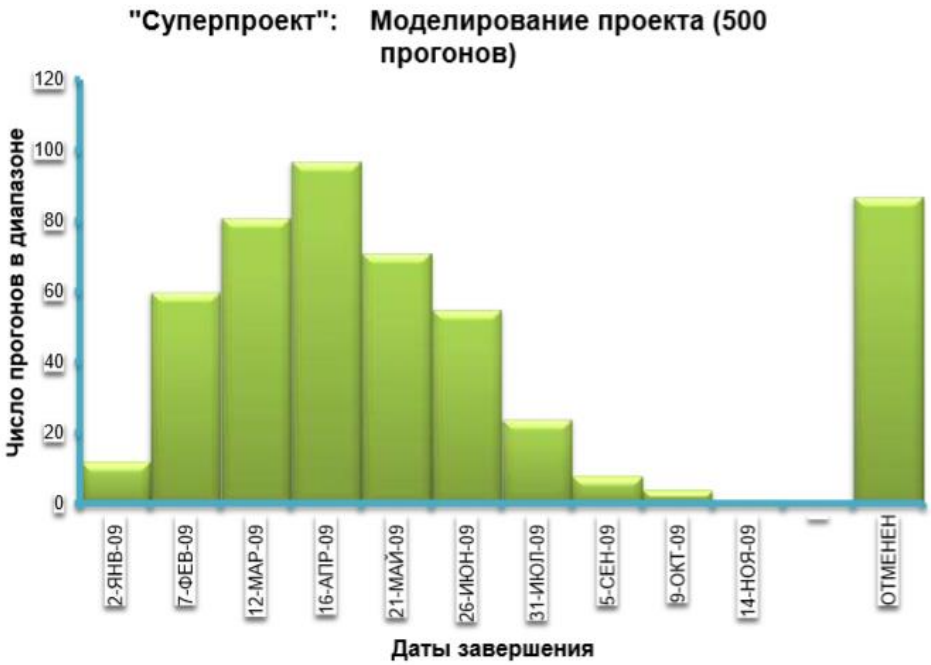

Рисунок 5.5 - Гистограмма распределения возможного срока завершения проекта, рассчитанная по результатам моделирования методом Монте-Карло

На диаграмме также приведено количество случаев, примерно 80 из 500 прогонов, в которых проект, согласно результатам моделирования, был отменен до своего завершения.

6 Планирование реагирования на риски

Планирование реагирования на риски – это процесс разработки путей и определения действий по увеличению возможностей и снижению угроз для целей проекта. Данный процесс начинается после проведения качественного и количественного анализа рисков.

Запланированные операции по реагированию на риски должны соответствовать серьезности риска, быть экономически эффективными в решении проблемы, своевременными, реалистичными в контексте проекта и согласованными со всеми участниками.

Возможны четыре метода реагирования на риски:

- Уклонение от риска (risk avoidance).
- Передача риска (risk transference).
- Снижение рисков (risk mitigation).
- Принятие риска (risk acceptance).

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). Некоторые риски, возникающие на ранних стадиях проекта, можно избежать при помощи уточнения требований, получения дополнительной информации или проведения экспертизы. Например, уклониться от риска можно, если 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. Передача риска просто переносит ответственность за его управление другой стороне, но риск при этом никуда не девается. Передача риска практически всегда предполагает выплату премии за риск стороне,

принимающей на себя риск. Например, заказ на стороне разработки рискованного компонента по фиксированной цене. В IT часто приходится формулировать риски в виде допущений, тем самым передавая его заказчику. Например, оценивая проект внедрения, мы можем записать допущение о том, что производитель не изменит стоимость лицензий на базовое ПО.

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 оказываются более эффективными, нежели усилия по устранению негативных последствий, предпринимаемые после наступления события риска. Например, раннее разрешение архитектурных рисков снижает потери при досрочном закрытии проекта. Или регулярная ревизия поставок заказчиком может снизить вероятность риска его неудовлетворенности конечным результатом. Если в проектной команде высока вероятность увольнения сотрудников, то введение на начальной стадии в проект дополнительных (избыточных) людских ресурсов снижает потери при увольнении членов команды, поскольку не будет затрат на «въезд» в проектный контекст новых участников.

И, наконец, принятие риска означает, что команда проекта осознанно приняла решение не изменять план управления проектом в связи с риском или не нашла подходящей стратегии реагирования. Мы вынуждены принимать все «неизвестные риски».

Принятие - это то, что всегда происходит, когда мы вообще не управляем рисками. Если же мы управляем рисками, то мы можем страховать риски, закладывая резерв в оценки срока завершения и/или трудозатрат. Проактивное отношение к принятым рискам может состоять в разработке план реагирования на риски. Этот план может быть введен в действие только при заранее определенных условиях, если есть уверенность и достаточное количество признаков того, что данный план будет успешно выполнен.

Важно помнить о вторичных рисках (Secondary Risks), возникающих в результате применения реагирования на риски, которые тоже должны быть идентифицированы, проанализированы и при необходимости включены в список управляемых рисков.

7 Главные риски программных проектов и способы реагирования

Список из пяти главных причин провала программных проектов - следующий:

- Требования заказчика отсутствуют / не полны / подвержены частым изменениям.
- Отсутствие необходимых ресурсов и опыта.
- Отсутствие рабочего взаимодействия с заказчиком.
- Неполнота планирования. «Забытые работы».
- Ошибки в оценках трудоемкостей и сроков работ.

Это звучит банально, но сколько бы раз об этом не твердили ранее, попрежнему, приходится сталкиваться с программными проектами, в которых отсутствуют какие-либо определенные цели и требования. Цитата из жизни: «Была бы разработана хорошая программа, а какой процесс автоматизировать с ее помощью, мы найдем». К этому можно добавить только одно: «Когда человек не знает, к какой пристани он держит путь, для него никакой ветер не будет попутным» (Сенека Луций Аней, философ, 65-3 до н.э.)

К часто упускаемым требованиям можно отнести:

- Функциональные
  - Программы установки, настройки, конфигурации.
  - Миграция данных.
  - Интерфейсы с внешними системами.
  - Справочная система.
- Общесистемные
  - Производительность.
  - Надежность.

o Открытость.
o Масштабируемость.
o Безопасность.
o Кросплатформенность.
o Эргономичность.

Как правило, эти требования «всплывают» при подготовке и проведении приемо-сдаточных испытаний и могут сильно задержать проект по времени и увеличить трудозатраты на его реализацию. Чтобы этого не происходило, следует достигать соглашения с заказчиком по всем перечисленным пунктам предпочтительнее еще на стадии инициации проекта. Например, если требования портируемости продукта на разные аппаратно-программные платформы нет, то это целесообразно включить в раздел концепции с допущениями проекта.

Если вероятность изменений требований проекта высока, то возможны следующие подходы для реагирования на данный риск:

 Переоценка проекта каждый раз, когда требования добавляются / изменяются (уклонение).

 Итерационная разработка. Контракт с компенсацией затрат на основе «Time & Materials» (передача риска Заказчику).

 Учет в оценках трудоемкости и сроков возможности роста требований, например, на 50% (резервирование риска).

И еще, при сборе требований следует соблюдать принцип минимализма Вольтера: «Рассказ закончен не тогда, когда в него нечего добавить, а тогда, когда из него нечего больше выкинуть». Для большинства программных продуктов применим принцип Парето: 80% ценности продукта заключены лишь в 20% требований к нему.

Если у нас в проекте недостаточно квалифицированных специалистов, то мы можем снизить последствия этого риска, применив следующие действия:

 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.

 Учесть в оценках «время разгона» для новых сотрудников.

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

 Постоянное взаимодействие.

 Согласование пользовательских интерфейсов и разработка прототипа продукта.

 Периодические поставки тестовых версий конечным пользователям для их оценки.

При планировании работ по проекту часто «забывают»:

 Обучение.

 Координация работ.

 Уточнение требований.

 Управление конфигурациями.

 Разработка и поддержка скриптов автосборки.

 Разработка автотестов.

 Создание тестовых данных.

 Обработка запросов на изменения.

И еще. Не стоит надеяться, что участники проекта будут каждую неделю по 40 часов работать именно над вашим проектом. Есть множество причин, по которым они не смогут работать по проекту 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.

Рекомендация, планировать, что разработчики, которые назначены в ваш проект на 100% будут реально работать над вашими задачами в среднем от 24 до 32 часов в неделю.

8 Управление проектом, направленное на снижение рисков

На стадии инициации проекта оценка его трудоемкости имеет погрешность от 50% до +100%. Это, если оценка хорошая! А если плохая, то неопределенность, а, следовательно, и риски сорвать сроки и превысить плановую трудоемкость, могут быть в разы больше. Если не прилагать специальных усилий этот «дамоклов меч» неопределенности будет висеть над проектом на всем его протяжении (рисунок 5.6).

Рисунок 5.6 - Неопределенность не уменьшается, если управление не направлено на раннее разрешение рисков

Проектом следует управлять так, чтобы риски несвоевременной сдачи и перерасхода ресурсов постоянно снижались.

Ранее мы уже говорили о том, что 80% ценности разработки обусловлена лишь 20% требований к продукту, без реализации которых продукт для заказчика становится просто ненужным. Остальные требования, как правило, так называемые «украшательства», от части которых заказчик, как правило, может отказаться, чтобы получить проект в срок. Поэтому следует в первую очередь реализовывать ключевые функциональные требования.

Но есть и еще архитектурные риски. Известно, что закон Парето применим и к потреблению вычислительных ресурсов: 80% потребления ресурсов (время и память) приходится на 20% компонентов. Поэтому, необходимо реализовывать архитектурно-значимые требования так же в первую очередь, создавая «представительный» прототип будущей системы, который «простреливает» весь стек, применяемых технологий. Прототип позволит измерить и оценить общесистемные свойства будущего продукта: доступность, быстродействие, надежность, масштабируемость и проч. (рисунок 5.7)

Ошибка - реализовывать сначала легкие требования, чтобы продемонстрировать быстрый прогресс проекта.

Рисунок 5.7 - Определение приоритетов требований на первые итерации проекта

Управление, нацеленное на снижение рисков, позволяет существенно снизить неопределенность на ранних стадиях проекта (рисунок 5.8).

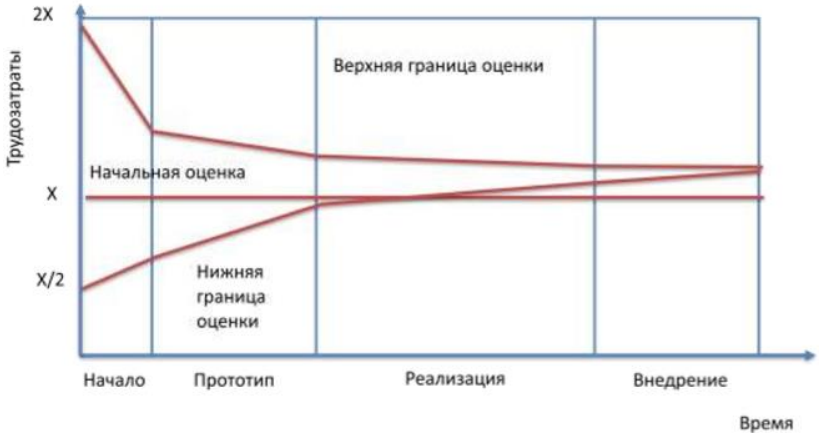

Рисунок 5.8 - Управление, нацеленное на снижение рисков, позволяет уменьшать неопределенность

Проработка ключевых функциональных требований и детальное планирование их реализации позволяет уменьшить разброс начальных оценок, примерно, в 2 раза: от -30% до +50%. Детальное проектирование и разработка прототипа будущей системы позволит получить еще более точные оценки общей трудоемкости: от -10% до +15%.

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 2 раза превысит первоначальную оценку трудоемкости.

Если с заказчиком не удается найти взаимоприемлемое решение при первоначальной оценке проекта, то разумно попытаться договориться о выполнении проекта в 2 этапа с самостоятельным финансированием:

1. Исследование. Бизнес-анализ, уточнение требований, проектирование и прототипироание решения, уточнение суммарных оценок трудозатрат. Эта работа, как правило, требует 10 % общих трудозатрат и 20% времени всего проекта.

2. Непосредственно реализация. Если уточненные оценки трудозатрат окажутся

приемлемыми для заказчика.

С вменяемыми заказчиками это часто удается.

8 Мониторинг и контроль рисков

Управление рисками должно осуществляться на протяжении всего проекта. Не вести мониторинг рисков в ходе проекта – все равно, что не следить за уровнем топлива при поездке на автомобиле.

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.

В процессе мониторинга и управления рисками используются различные методики, например, анализ трендов и отклонений, для выполнения которых необходимы количественные данные об исполнении, собранные в процессе выполнения проекта.

Мониторинг и управление рисками включает 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. Неплохо начинать каждый статус митинг с вопроса: «Ну и какие еще неприятности нас ожидают?» Идентификация новых рисков, и пересмотр известных рисков происходит с использованием процессов, описанных ранее.

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

Тренды в процессе выполнения проекта подлежат проверке с использованием данных о выполнении. Для мониторинга выполнения всего проекта могут использоваться анализ освоенного объема и другие методы анализа отклонений проекта и трендов (см. Лекция 8. Реализация проекта). На основании выходов этих анализов можно прогнозировать потенциальные отклонения проекта на момент его завершения по показателям стоимости и расписания. Отклонения от базового плана могут указывать на последствия, вызванные как угрозами, так и благоприятными возможностями.

Выводы

Отказываться от управления проектными рисками это все равно, что в кинотеатре не иметь огнетушителей и плана эвакуации на случай пожара.

Все, что мы делаем, управляя проектом разработки ПО, должно быть направлено на борьбу с рисками не уложиться в срок, перерасходовать ресурсы, разработать не тот продукт, который требуется.

Цели управления рисками проекта – снижение вероятности возникновения и/или значимости воздействия неблагоприятных для проекта событий.

Главные причины провала программных проектов:

- Требования заказчика отсутствуют / не полны / подвержены частым изменениям.
- Отсутствие необходимых ресурсов и опыта.
- Отсутствие рабочего взаимодействия с заказчиком
- Неполнота планирования. «Забытые работы».
- Ошибки в оценках трудоемкостей и сроков работ.

Тема 6 Оценка трудоемкости и сроков разработки ПО

1 Оценка - вероятностное утверждение

Первое, что необходимо понимать при оценке проекта, это то, что любая оценка - это всегда вероятностное утверждение. Если мы просто скажем, что трудоемкость данного пакета работ составляет М чел.*мес. (Рисунок 34), то это будет плохой оценкой потому, что одно единственное число ничего не скажет нам о вероятности того, что на реализацию этого пакета потребуется не более, чем М чел.*мес. Вряд ли мы можем считать себя «предсказамусами», которые точно знают, что произойдет в будущем и сколько потребуется затрат на реализацию этого пакета работ.

Для того, чтобы понять, откуда берется неопределенность, рассмотрим простейший пример, попытаемся оценить трудоемкость добавления поля ввода телефонного номера клиента к уже существующей форме. Менеджер, наблюдающий работу программистов только со стороны, скажет, что эта работа потребует не больше 15 минут рабочего времени. Человек, умудренный программистским опытом, скажет, что эта работа может занять от 2 до 200 часов, и чтобы дать более точную оценку ему надо получить ответы на ряд вопросов:

- Может ли вводиться несколько номеров?
- Должна ли быть проверка номеров на действительность?
- Простая или сложная проверка?
- Если реализуем простую проверку, то не захочет ли клиент заменить ее на более сложную?
- Должна ли проверка работать для иностранных номеров?
- Можно ли воспользоваться готовым решением?
- Каково должно быть качество реализации? Вероятность ошибки после поставки?
- Сколько времени потребуется на реализацию и отладку? (зависит от конкретного исполнителя).

Называя такую «размытую» оценку опытный программист резервирует все риски разработки, связанные с перечисленными неопределенностями данного требования, которые он вынужден принимать на себя, не имея в данный момент необходимой уточняющей информации.

То, что наша оценка должна быть вероятностным утверждением, означает, что для нее существует некоторое распределение вероятности (рисунок 6.1), которое может быть очень широким (высокая неопределенность) или достаточно узким (низкая неопределенность).

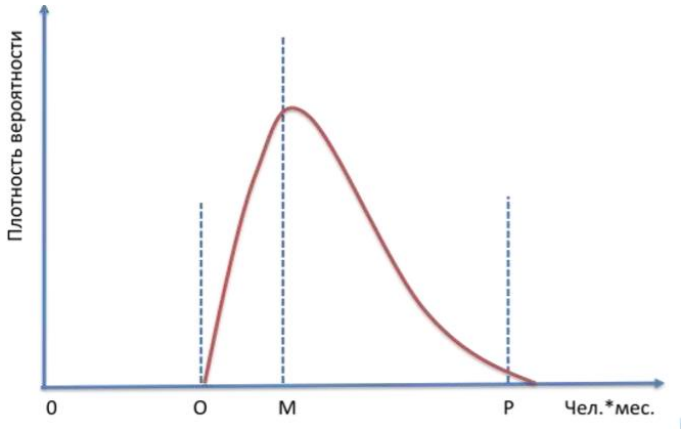

Рисунок 6.1 - Оценка – всегда вероятностная величина

Если М – наиболее вероятное значение, то это не означает что это хорошая оценка, поскольку вероятность того, что фактическая трудоемкость превысит эту оценку, составляет более 50%.

Какая оценка может считаться хорошей? Стив Макконнелл утверждает: «Хорошей

считается оценка, которая обеспечивает достаточно ясное представление реального состояния проекта и позволяет руководителю проекта принимать хорошие решения относительно того, как управлять проектом для достижения целей».

2 Негативные последствия «агрессивного» расписания

В программостроении уже стало банальностью то, что разработчики без достаточного основания называют слишком оптимистичные сроки. Среди руководителей даже распространено неписаное правило: умножать на 2 оценку трудоемкости, которую сделал программист. Это пессимистичный подход. Реалисты умножают на $\pi = 3.14$.

Действительно, так иногда приходится поступать, если это программист, который только вчера отладил свою первую программу «Hello world!». Но если помочь молодым специалистам научится анализировать задачу, проектировать решение, составлять план работы, эффективно его реализовывать и анализировать полученные результаты, то можно будет не вспоминать, чему равно число π.

Еще один распространенный источник занижения сроков - необоснованные ожидания на применение новых технологий и средств разработки. Эти ожидания, как правило, не оправдываются. Согласно статистике, приведенной Демарко, средняя производительность в программном производстве растет всего лишь на 3-5% в год.

Часто «агрессивное» расписание проекта появляется из-за того, что руководство и/или заказчик боятся переоценить проект, полагая, что согласно закону Паркинсона, работы по проекту займут все отведенное для него время. Следствием подобных опасений является, как правило, директивное занижение сроков реализации проекта.

Не реалистичность оценок один из серьезнейших демотивирующих факторов для участников. Недооценка приводит к ошибкам планирования и неэффективному взаимодействию. Например, было запланировано тестирование, а релиз еще не готов. Следствие - простой тестировщиков увеличение трудозатрат.

Если расписание излишне агрессивное, то с целью сэкономить время, недостаточно внимания уделяется анализу требований и проектированию. Исправление ошибок, допущенных на этих этапах, приведет к существенным дополнительным затратам.

Половина всех ошибок программирования возникают из-за стресса, вызванного излишним давлением фактора сроков. Ошибки исправляются наспех, обходными путями. В результате будет получен большой проблемный код и постоянно растущие затраты на исправление ошибок и внесение изменений. Позднее обнаружение ошибок приводит к тому, что затраты на их исправление увеличиваются в 50-100 раз.

3 Прагматичный подход. Метод PERT

Использование собственного опыта или опыта коллег, полученного в похожих проектах, это наиболее прагматичный подход, который позволяет получить достаточно реалистичные оценки трудоемкости и срока реализации программного проекта, быстро и без больших затрат.

Инженерный метод оценки трудоемкости проекта PERT (Program / Project Evaluation and Review Technique) был разработан в 1958 году в ходе проекта по созданию баллистических ракет морского базирования «Поларис». Входом для данного метода оценки служит список элементарных пакетов работ. Для инженерного подхода нет необходимости точно знать закон распределения нашей оценки трудоемкости каждого такого элементарного пакета. Диапазон неопределенности достаточно охарактеризовать тремя оценками:

M_i – наиболее вероятная оценка трудозатрат.

O_i – минимально возможные трудозатраты на реализацию пакета работ. Ни один риск не реализовался. Быстрее точно не сделаем. Вероятность такого, что мы уложимся в эти

затраты, равна 0.

P_i – пессимистическая оценка трудозатрат. Все риски реализовались.

Оценку средней трудоемкости по каждому элементарному пакету можно определит по формуле:

$$E_i = (P_i + 4M_i + O_i)/6.$$

Для расчета среднеквадратичного отклонения используется формула:

$$СКО_i = (P_i - O_i)/6.$$

Если наши оценки трудоемкости элементарных пакетов работ статистически независимы, а не испорчены, например, необоснованным оптимизмом то, согласно центральной предельной теореме теории вероятностей суммарная трудоемкость проекта может быть рассчитана по формуле:

$$E = \Sigma E_i$$

А среднеквадратичное отклонение для *оценки* суммарной трудоемкости будет составлять:

$$СКО=(\Sigma СКО_i^2)\wedge(1/2)$$

Тогда для оценки суммарной трудоемкости проекта, которую мы не превысим с вероятностью 95%, можно применить формулу:

$$E_{95\%} = E + 2 * СКО.$$

Это значит, что вероятность того, что проект превысит данную оценку трудоемкости составляет всего 5%. А это уже вполне приемлемая оценка, под которой может расписаться профессиональный менеджер.

Список элементарных пакетов работ, который используется при оценке трудоемкости, как правило, берется из нижнего уровня ИСР проекта. Но может быть использован и накопленный опыт аналогичных разработок. Проиллюстрирую данный подход на примере реального проекта. В Ассоциации CBOSS задачей проекта, который нам с коллегами посчастливилось реализовывать, была разработка на основе стандартов J2EE общесистемного ПО для перевода рабочих мест CBOSS на новую трехзвенную архитектуру. Был разработан набор стандартных компонентов и сервисов, из которых как из конструктора можно эффективно и качественно собирать прикладные подсистемы. Высокоуровневая архитектура реализовывала стандартный паттерн MVC (рисунок 6.2), каждый из компонентов которого имел «точки расширения» для прикладной разработки, которые на рисунке выделены красным светом.

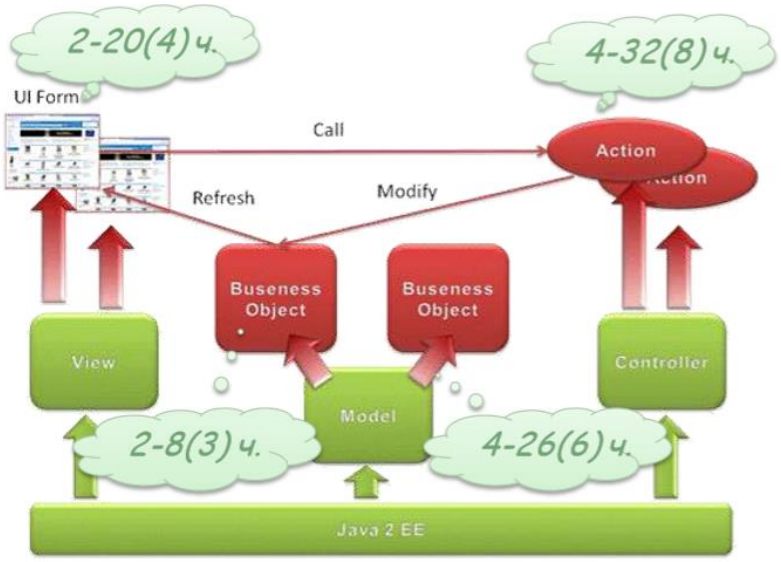

Рисунок 6.2 - Высокоуровневая архитектура J2EE фреймворка для разработки приложений.

Такими точками расширения являлись:

 Пользовательский экран (UI Form), который собирался из готовых визуальных компонентов.

 Обработчики (Action), которые обрабатывали на сервере приложений события от активных визуальных компонентов, входящих в состав экрана.

 Объекты (Business Obj), которые моделировали прикладную область, и к которым обращались обработчики событий.

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

Согласно этой статистике, разработка и отладка требовала у программиста:

 для одного экрана - от 2 до 20 часов (наиболее вероятно – 4 часа);

 для одного обработчика событий - от 4 до 32 часов (наиболее вероятно – 8 часов);

 для нового бизнес-объекта - от 2 до 8 часов (наиболее вероятно – 3 часа);

 для добавление нового бизнес-метода - от 2 до 26 часов (наиболее вероятно – 6 часов).

Весь проект прикладной разработки измерялся в «попугаях»:

 KUI – количество пользовательских экранов.

 KAct – количество обработчиков событий.

 KBO – количество новых бизнес-объектов.

 KBM – количество новых или модифицируемых бизнес-методов.

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

EUI = (2 + 4*4 + 20) / 6 = 6.7 чел.*час., СКОUI = (20 - 2) / 6 = 3 чел.*час

EAct = (4 + 4*8 + 32) / 6 = 11.3 чел.*час., СКОAtct = (32 - 4) / 6 = 4.7 чел.*час

EBO = (2 +4*3 + 8) / 6 = 3.7 чел.*час., СКОBO = (8 - 2) / 6 = 1 чел.*час

EBM = (2 + 4*6 + 26) / 6 = 8.7 чел.*час., СКОBM = (26 - 2) / 6 = 4 чел.*час

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

$$E = 20 * 6.7 + 60 * 11.3 + 16 * 3.7 + 40 * 8.7 \approx 1220 \text{ чел.*час.}$$

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

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

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

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

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

составит, приблизительно, 5200 чел.*час.

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

Тогда согласно формуле Б.Боэма оптимальная продолжительность проекта составит:

T = 2.5 * (40) ^ 1/3 = 8.5 месяцев,

а средняя численность команды – 5 человек.

Потребление ресурсов в проекте неравномерно, поэтому начинать проект должны 1-3 человека, а на стадии реализации начальная численность команды может быть увеличена в несколько раз.

Если же собственный опыт аналогичных проектов отсутствует, а коллеги-эксперты недоступны, то нам не остается ничего другого, как использовать формальные методики, основанные на обобщенном отраслевом опыте. Среди них наибольшее распространение получили два подхода:

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

5 Обзор метода функциональных точек

Анализ функциональных точек - стандартный метод измерения размера программного продукта с точки зрения пользователей системы. Метод разработан Аланом Альбрехтом (Alan Albrecht) в середине 70-х. Метод был впервые опубликован в 1979 году. В 1986 году была сформирована Международная Ассоциация Пользователей Функциональных Точек (International Function Point User Group – IFPUG), которая опубликовала несколько ревизий метода.

Метод предназначен для оценки на основе логической модели объема программного продукта количеством функционала, востребованного заказчиком и поставляемого разработчиком. Несомненным достоинством метода является то, что измерения не зависят от технологической платформы, на которой будет разрабатываться продукт, и он обеспечивает единообразный подход к оценке всех проектов в компании.

При анализе методом функциональных точек надо выполнить следующую последовательность шагов (рисунок 6.3):

1. Определение типа оценки.
2. Определение области оценки и границ продукта.
3. Подсчет функциональных точек, связанных с данными.
4. Подсчет функциональных точек, связанных с транзакциями.
5. Определение суммарного количества не выровненных функциональных точек (UFP).
6. Определение значения фактора выравнивания (FAV).
7. Расчет количества выровненных функциональных точек (AFP).

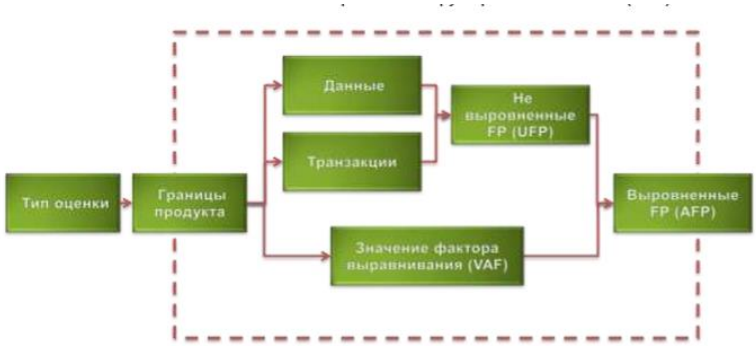

Рисунок 6.3 - Процедура анализа по методу функциональных точек

Определение типа оценки

Первое, что необходимо сделать, это определить тип выполняемой оценки. Метод предусматривает оценки трех типов:

1. Проект разработки. Оценивается количество функциональности поставляемой пользователям в первом релизе продукта.

2. Проект развития. Оценивается в функциональных точках проект доработки: добавление, изменение и удаление функционала.

3. Продукт. Оценивается объем уже существующего и установленного продукта.

Определение области оценки и границ продукта

Второй шаг – это определение области оценки и границ продукта. В зависимости от типа область оценки может включать:

 Все разрабатываемые функции (для проекта разработки)

 Все добавляемые, изменяемые и удаляемые функции (для проектов поддержки)

 Только функции, реально используемые, или все функции (при оценке продукта и/или продуктов).

Третий шаг. Границы продукта (Рисунок 38) определяют:

 Что является «внешним» по отношению к оцениваемому продукту.

 Где располагается «граница системы», через которую проходят транзакции передаваемые или принимаемые продуктом, с точки зрения пользователя.

 Какие данные поддерживаются приложением, а какие – внешние.

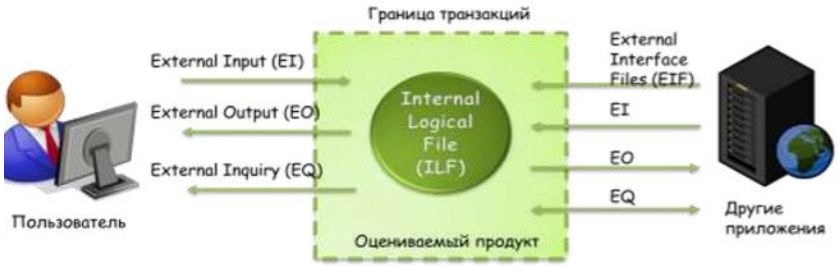

Рисунок 6.4 - Границы продукта в методе функциональных точек

К логическим данным системы относятся:

 Внутренние логические файлы (ILFs) – выделяемые пользователем логически связанные группы данных или блоки управляющей информации, которые поддерживаются внутри продукта.

 Внешние интерфейсные файлы (EIFs) - выделяемые пользователем логически связанные группы данных или блоки управляющей информации, на которые ссылается

продукт, но которые поддерживаются вне продукта.

Примером логических данных (информационных объектов) могут служить: клиент, счет, тарифный план, услуга.

Подсчет функциональных точек, связанных с данными

Третий шаг - подсчет функциональных точек, связанных с данными. Сначала определяется сложность данных по следующим показателям:

 DET (data element type) – неповторяемое уникальное поле данных, например, Имя Клиента – 1 DET; Адрес Клиента (индекс, страна, область, район, город, улица, дом, корпус, квартира) – 9 DET’s

 RET (record element type) – логическая группа данных, например, адрес, паспорт, телефонный номер.

Оценка количества не выровненных функциональных точек, зависит от сложности данных, которая определяется на основании матрицы сложности (таблица 6.1).

Таблица 6.1 Матрица сложности данных

	1-19 DET	20-50 DET	50+ DET
1 RET	Low	Low	Average
2-5 RET	Low	Average	High
6+ RET	Average	High	High

Оценка данных в не выровненных функциональных точках (UFP) подсчитывается по-разному для внутренних логических файлов (ILFs) и для внешних интерфейсных файлов (EIFs) (таблица 6.2) в зависимости от их сложности.

Таблица 6.2 - Оценка данных в не выровненных функциональных точках (UFP) для внутренних логических файлов (ILFs) и внешних интерфейсных файлов (EIFs)

Сложность данных	Количество UFP (ILF)	Количество UFP (EIF)
Low	7	5
Average	10	7
High	15	10

Для иллюстрации рассмотрим пример оценки в не выровненных функциональных точках объекта данных «Клиент» (рисунок 6.5).

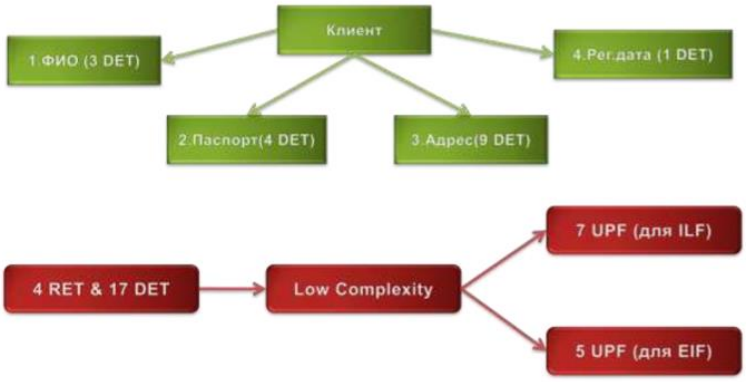

Рисунок 6.5 - Пример оценки в не выровненных функциональных точках объекта данных «Клиент».

Объект «Клиент» содержит четыре логических группы данных, которые в совокупности состоят из 15 неповторяемых уникальное полей данных. Согласно матрице (таблица 6.1), следует оценить сложность этого объекта данных, как «Low». Теперь, если оцениваемый объект относится к внутренним логическим файлам, то согласно таблице 6.2 его сложность будет 7 не выровненных функциональных точек (UPF). Если же объект является внешним интерфейсным файлом, то его сложность составит 5 UPF.

Подсчет функциональных точек, связанных с транзакциями

Подсчет функциональных точек, связанных с транзакциями – это четвертый шаг анализа по методу функциональных точек.

Транзакция – это элементарный неделимый замкнутый процесс, представляющий значение для пользователя и переводящий продукт из одного консистентного состояния в другое.

В методе различаются следующие типы транзакций (таблица 6.3):

 EI (external inputs) – внешние входные транзакции, элементарная операция по обработке данных или управляющей информации, поступающих в систему из вне.

 EO (external outputs) – внешние выходные транзакции, элементарная операция по генерации данных или управляющей информации, которые выходят за пределы системы. Предполагает определенную логику обработки или вычислений информации из одного или более ILF.

 EQ (external inquiries) – внешние запросы, элементарная операция, которая в ответ на внешний запрос извлекает данные или управляющую информацию из ILF или EIF.

Таблица 6.3 - Основные отличия между типами транзакций. Легенда: О – основная; Д – дополнительная; NA – не применима

Функция	Тип транзакции		
	EI	EO	EQ
Изменяет поведение системы	О	д	NA
Поддержка одного или более ILF	О	д	NA
Представление информации пользователю	д	О	О

Оценка сложности транзакции основывается на следующих ее характеристиках:

 FTR (file type referenced) – позволяет подсчитать количество различных файлов (информационных объектов) типа ILF и/или EIF модифицируемых или считываемых в транзакции.

 DET (data element type) – неповторяемое уникальное поле данных. Примеры. EI: поле ввода, кнопка. EO: поле данных отчета, сообщение об ошибке. EQ: поле ввода для поиска, поле вывода результата поиска.

Для оценки сложности транзакций служат матрицы, которые представлены в таблицах 6.4 и 6.5.

Таблица 6.4 -Матрица сложности внешних входных транзакций (EI)

El	1-4 DET	5-15 DET	16+ DET
0-1 FTR	Low	Low	Average
2 FTR	Low	Average	High

3+ FTR	Average	High	High

Таблица 6.5 - Матрица сложности внешних выходных транзакций и внешних запросов (EO & EQ)

EO&EQ	1-5 DET	6-19 DET	20+ DET
0-1 FTR	Low	Low	Average
2-3 FTR	Low	Average	High
4+ FTR	Average	High	High

Оценка транзакций в не выровненных функциональных точках (UFP) может быть получена из матрицы (таблица 6.6)

Таблица 6.6 - Сложность транзакций в не выровненных функциональных точках (UFP)

Сложность транзакций	Количество UFP (El & EQ)	Количество UFP (EO)
Low	3	4
Average	4	5
High	6	7

В качестве примера, рассмотрим оценку управляющей транзакции (EI) для диалогового окна, задающего параметры проверки орфографии в MS Office Outlook (рисунок 6.7).

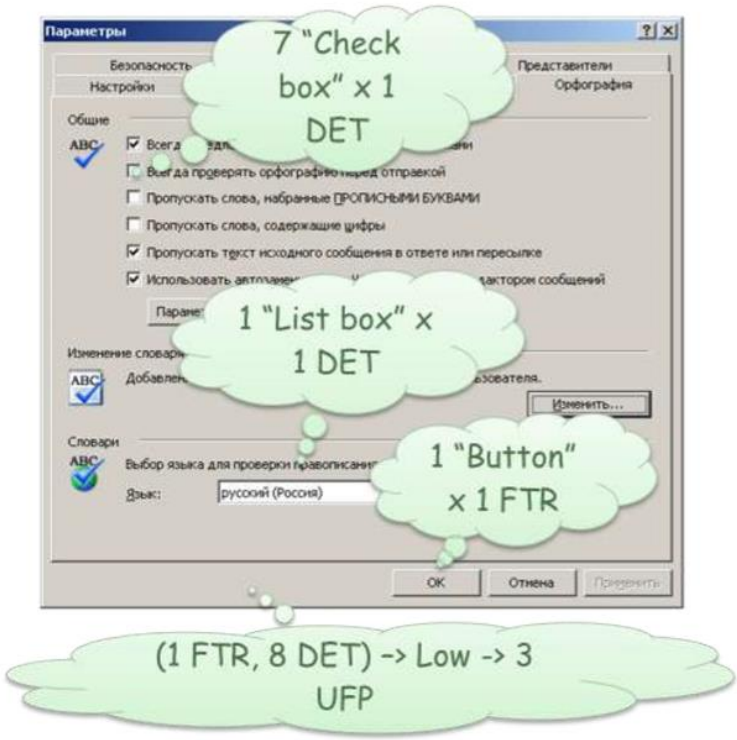

Рисунок 6.7 - Диалоговое окно, управляющее проверкой орфографии в MS Office Outlook

Каждый “Check box” оценивается, как 1 DET. Выпадающий список - 1 DET. Каждая

управляющая кнопка должна рассматриваться как отдельная транзакция. Например, если оценивать управляющую транзакцию по кнопке «OK», то, для данной транзакции мы имеем 1 FTR и 8 DET. Поэтому, согласно матрице (Таблица 10), мы можем оценить сложность транзакции, как Low. И, наконец, в соответствие с матрицей (Таблица 12), данная транзакция должна быть оценена в 3 не выровненных функциональных точек (UFP).

Определение суммарного количества не выровненных функциональных точек (UFP)

Общий объем продукта в не выровненных функциональных точках (UFP) определяется путем суммирования по всем информационным объектам (ILF, EIF) и элементарным операциям (транзакциям EI, EO, EQ).

$$UFP = \sum_{ILF} UFPi + \sum_{EIF} UFPi + \sum_{EI} UFPi + \sum_{EO} UFPi + \sum_{EQ} UFPi$$

Определение значения фактора выравнивания (FAV)

Помимо функциональных требований на продукт накладываются общесистемные требования, которые ограничивают разработчиков в выборе решения и увеличивают сложность разработки. Для учета этой сложности применяется фактор выравнивания (VAF). Значение фактора VAF зависит от 14 параметров, которые определяют системные характеристики продукта:

1. Обмен данными (0 – продукт представляет собой автономное приложение; 5 – продукт обменивается данными по более, чем одному телекоммуникационному протоколу).
2. Распределенная обработка данных (0 – продукт не перемещает данные; 5 – распределенная обработка данных выполняется несколькими компонентами системы).
3. Производительность (0 – пользовательские требования по производительности не установлены; 5 – время отклика сильно ограничено критично для всех бизнес-операций, для удовлетворения требованиям необходимы специальные проектные решения и инструменты анализа.
4. Ограничения по аппаратным ресурсам (0 – нет ограничений; 5 – продукт целиком должен функционировать на определенном процессоре и не может быть распределен).
5. Транзакционная нагрузка (0 – транзакций не много, без пиков; 5 – число транзакций велико и неравномерно, требуются специальные решения и инструменты).
6. Интенсивность взаимодействия с пользователем (0 – все транзакции обрабатываются в пакетном режиме; 5 – более 30% транзакций – интерактивные).
7. Эргономика (эффективность работы конечных пользователей) (0 – нет специальных требований; 5 – требования по эффективности очень жесткие).
8. Интенсивность изменения данных (ILF) пользователями (0 – не требуются; 5 – изменения интенсивные, жесткие требования по восстановлению).
9. Сложность обработки (0 – обработка минимальна; 5 – требования безопасности, логическая и математическая сложность, многопоточность).
10. Повторное использование (0 – не требуется; 5 – продукт разрабатывается как стандартный многоразовый компонент).
11. Удобство инсталляции (0 – нет требований; 5 – установка и обновление ПО производится автоматически).
12. Удобство администрирования (0 – не требуется; 5 – система автоматически самовосстанавливается).
13. Портируемость (0 – продукт имеет только 1 инсталляцию на единственном процессоре; 5 – система является распределенной и предполагает установку на различные «железо» и ОС).
14. Гибкость (0 – не требуется; 5 – гибкая система запросов и построение произвольных отчетов, модель данных изменяется пользователем в интерактивном режиме).

14 системных параметров (degree of influence, DI) оцениваются по шкале от 0 до 5. Расчет суммарного эффекта 14 системных характеристик (total degree of influence, TDI) осуществляется простым суммированием:

$$TDI = \Sigma DI$$

Расчет значения фактора выравнивания производится по формуле

$$VAF = (TDI * 0.01) + 0.65$$

Например, если, каждый из 14 системных параметров получил оценку 3, то их суммарный эффект составит TDI = 3 * 14 = 42. В этом случае значение фактора выравнивания будет: VAF = (42 * 0.01) + 0.65 = 1.07

Расчет количества выровненных функциональных точек (AFP)

Дальнейшая оценка в выровненных функциональных точках зависит от типа оценки. Начальное оценка количества выровненных функциональных точек для программного приложения определяется по следующей формуле:

$$AFP = UFP * VAF.$$

Она учитывает только новую функциональностсть, которая реализуется в продукте. Проект разработки продукта оценивается в DFP (development functional point) по формуле:

$$DFP = (UFP + CFP) * VAF,$$

где CFP (conversion functional point) – функциональные точки, подсчитанные для дополнительной функциональности, которая потребуется при установке продукта, например, миграции данных.

Проект доработки и совершенствования продукта оценивается в EFP (enhancement functional point) по формуле:

$$EFP = (ADD + CHGA + CFP) * VAFA + (DEL* VAFB),$$

где

ADD - функциональные точки для добавленной функциональности;

CHGA - функциональные точки для измененных функций, рассчитанные после модификации;

VAFA – величина фактора выравнивания рассчитанного после завершения проекта;

DEL – объем удаленной функциональности;

VAFB - величина фактора выравнивания рассчитанного до начала проекта.

Суммарное влияние процедуры выравнивания лежит в пределах +/- 35% относительно объема рассчитанного в UFP.

Метод анализа функциональных точек ничего не говорит о трудоемкости разработки оцененного продукта. Вопрос решается просто, если компания разработчик имеет собственную статистику трудозатрат на реализацию функциональных точек. Если такой статистики нет, то для оценки трудоемкости и сроков проекта можно использовать метод COCOMO II.

6 Основы методики COCOMO II

Методика COCOMO позволяет оценить трудоемкость и время разработки программного продукта. Впервые была опубликована Бари Боэмом [3] в 1981 году в виде результат анализа 63 проектов компании «TRW Aerospace». В 1997 методика была усовершенствована и получила название COCOMO II. Калибровка параметров производилась по 161 проекту разработки. В модели используется формула регрессии с параметрами, определяемыми на основе отраслевых данных и характеристик конкретного проекта.

Различаются две стадии оценки проекта: предварительная оценка на начальной фазе и детальная оценка после проработки архитектуры.

Формула оценки трудоемкости проекта в чел.*мес. имеет вид:

$$PM = A \times SIZE^E \times \prod_{i=1}^{n} EM_i$$

$$A = 2{,}94$$

$$E = B + 0{,}01 \times \sum_{j=1}^{5} SF_j$$

$$B = 0{,}91$$

где
SIZE- размер продукта в KSLOC
EM i - множители трудоемкости
SF_j - факторы масштаба
n =7 - для предварительной оценки
n =17 - для детальной оценки

Главной особенностью методики является то, что для того, чтобы оценить трудоемкость, необходимо знать размер программного продукта в тысячах строках исходного кода (KSLOC, Kilo Source Lines Of Code). Размер программного продукта может быть, например, оценен экспертами с применением метода PERT.

Если мы провели анализ продукта методом функциональных точек, то его размер может быть рассчитан с использованием собственных статистических данных или с использованием статистики по отрасли (таблица 6.7).

Таблица 6.7 - Оценка количества строк, необходимых на реализацию одной не выровненной функциональной точки для некоторых распространенных языков программирования.

Язык программирования	Оценка количества строк		
	Наиболее вероятная	Оптимистичная	Пессимистичная
Assembler	172	86	320
C	148	9	704
C++	60	29	178
C#	59	51	66
J2EE	61	50	100
JavaScript	56	44	65
PL/SQL	46	14	110
Visual Basic	50	14	276

Факторы масштаба

В методике используются пять факторов масштаба SFj, которые определяются следующими характеристиками проекта:

1. PREC – прецедентность, наличие опыт аналогичных разработок (Very Low – опыт в продукте и платформе отсутствует; Extra High – продукт и платформа полностью знакомы)
2. FLEX – гибкость процесса разработки (Very Low – процесс строго детерминирован; Extra High – определены только общие цели).
3. RESL – архитектура и разрешение рисков (Very Low – риски неизвестны/не проанализированы; Extra High – риски разрешены на 100%)
4. TEAM – сработанность команды (Very Low – формальные взаимодействия; Extra High – полное доверие, взаимозаменяемость и взаимопомощь).
5. PMAT – зрелость процессов (Very Low – CMM Level 1; Extra High – CMM Level 5)

Значение фактора масштаб, в зависимости от оценки его уровня, приведены в таблице 6.8.

Таблица 6.8 - Значение фактора масштаба, в зависимости от оценки его уровня

Фактор маштаба	Оценка уровня фактора					
	Very Low	Low	Nominal	High	Very High	Extra High
PREC	6.20	4.96	3.72	2.48	1.24	0
FLEX	5.07	4.05	3.04	2.03	1.01	0
RESL	7.07	5.65	4.24	2.83	1.41	0
TEAM	5.48	4.38	3.29	2.19	1.10	0
PMAT	7.80	6.24	4.68	3.12	1.56	0

Множители трудоемкости

В нашу задачу не входит детальное описание метода COCOMO II, поэтому мы рассмотрим только случай предварительной оценки трудоемкости программного проекта. Для этой оценки необходимо оценить для проекта уровень семи множителей трудоемкости Mi:

1. PERS – квалификация персонала (Extra Low – аналитики и программисты имеют низшую квалификацию, текучесть больше 45%; Extra High - аналитики и программисты имеют высшую квалификацию, текучесть меньше 4%)

2. RCPX – сложность и надежность продукта (Extra Low – продукт простой, специальных требований по надежности нет, БД маленькая, документация не требуется; Extra High - продукт очень сложный, требования по надежности жесткие, БД сверхбольшая, документация требуется в полном объеме)

3. RUSE – разработка для повторного использования (Low – не требуется; Extra High – требуется переиспользование в других продуктах)

4. PDIF – сложность платформы разработки (Extra Low – специальные ограничения по памяти и быстродействию отсутствуют, платформа стабильна; Extra High – жесткие ограничения по памяти и быстродействию, платформа нестабильна)

5. PREX – опыт персонала (Extra Low – новое приложение, инструменты и платформа; Extra High - приложение, инструменты и платформа хорошо известны)

6. FCIL – оборудование (Extra Low – инструменты простейшие, коммуникации затруднены; Extra High – интегрированные средства поддержки жизненного цикла, интерактивные мультимедиа коммуникации)

7. SCED – сжатие расписания (Very Low – 75% от номинальной длительности; Very High – 160% от номинальной длительности)

Влияние множителей трудоемкости в зависимости от их уровня определяется их числовыми значениями, которые представлены в матрице, приведенной в таблице 6.8.

Таблица 6.8 - Значения множителей трудоемкости, в зависимости от оценки их уровня

EM_i	Оценка уровня фактора						
	Extra Low	Very Low	Low	Nominal	High	Very High	Extra High
PREC	2.12	1.62	1.26	1.00	0.83	0.63	0.5
RCPX	0.49	0.60	0.83	1.00	1.33	1.91	2.72
RUSE	n/a	n/a	0.95	1.00	1.07	1.15	1.24
PDIF	n/a	n/a	0.87	1.00	1.29	1.81	2.61
PREX	1.59	1.33	1.22	1.00	0.87	0.74	0.62
FOIL	1.43	1.30	1.10	1.0	0.87	0.73	0.62
SCED	n/a	1.43	1.14	1.00	1.00	1.00	n/a

Из этой таблицы, в частности, следует, если в нашем проекте низкая квалификация аналитиков, то его трудоемкость возрастет примерно в 4 раза по сравнению с проектом, в котором участвуют аналитики экстра-класса.

Оценка многокомпонентного продукта

Для того чтобы адекватно спланировать проект и оценить его трудоемкость, необходимо выполнить предварительное проектирование программного продукта. В результате декомпозиции мы получаем некоторое количество компонентов (N), которые составляют программный продукт.

Следует понимать, что суммарная трудоемкость проекта не равна простой сумме трудоемкостей разработки каждого из компонентов:

$$PM \neq \sum_{k=1}^{N} PM_k$$

Простая сумма не учитывает взаимосвязи компонентов и трудозатраты на их интеграцию.

Методика COCOMO II определяет следующую последовательность вычисления трудоемкости проекта при многокомпонентной разработке.

1. Суммарный размер продукта рассчитывается, как сумма размеров его компонентов:

$$SIZE^A = \sum_{k=1}^{N} SIZE_k$$

2. Базовая трудоемкость проекта рассчитывается по формуле:

$$PM^B = A \times (SIZE^A)^E \times SCED$$

3. Затем рассчитывается базовая трудоемкость каждого компонента:

$$PM_k^B = PM^B \times \frac{SIZE_k}{SIZE^A}$$

4. На следующем шаге рассчитывается оценка трудоемкости компонентов с учетом всех множителей трудоемкости, кроме множителя SCED.

$$PM_k' = PM_k^B \times \prod_{i=1}^{6} EM_i$$

5. Итоговая трудоемкость проекта определятся по формуле:

$$PM = \sum_{k=1}^{N} PM_k'$$

Оценка длительности проекта

Длительность проекта в методике COCOMO II рассчитывается по формуле:

$$TDEV = C \times (PM_{NS})^{D+0,2\times 0,01\times \sum_{j=1}^{5} SF_j} \times \frac{SCED}{100},$$

где

C=3,67; D=0,28

PM_{NS} – трудоемкость проекта без учета множителя SCED, определяющего сжатие расписания.

Выводы

Оценка трудоемкости должна быть вероятностным утверждением. Это означает, что для нее существует некоторое распределение вероятности, которое может быть очень широким, если неопределенность высокая, или достаточно узким, если неопределенность низкая.

Использование собственного опыта или опыта коллег, полученного в похожих проектах, это наиболее прагматичный подход, который позволяет получить достаточно реалистичные оценки трудоемкости и срока реализации программного проекта, быстро и без больших затрат.

Если собственный опыт аналогичных проектов отсутствует, а коллеги-эксперты недоступны, то необходимо использовать формальные методики, основанные на обобщенном отраслевом опыте. Среди них наибольшее распространение получили два подхода:

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

Не реалистичность оценок один из серьезнейших демотивирующих факторов для участников проектной команды. Недооценка приводит к ошибкам планирования и неэффективному взаимодействию. Агрессивные сроки, постоянное давление, сверхурочные, авралы служат причиной того, что затраты на проект растут экспоненциально и неограниченно.

Тема 7 Формирование команды

1 Лидерство и управление

В работе руководителя проекта есть две стороны: управление и лидерство, которые одинаково важны и не могут существовать в отрыве друг от друга. Нельзя быть лидером материальных ресурсов, денежных потоков, планов, графиков и рисков. Ими необходимо управлять. Потому что у вещей нет права и свободы выбора, присущих только человеку.

Интеллектуальными людьми невозможно управлять. Творческие команды можно только направлять и вести. «Высокопроизводительное управление в отсутствие эффективного лидерства подобно упорядочению расстановки стульев на палубе тонущего «Титаника». Никакой успех в управлении не компенсирует провала в лидерстве».

Эффективные команды не образуются сами по себе, они кристаллизуются вокруг признанного лидера. Как не бывает лидеров без последователей, так и не бывает команд без лидеров. Поэтому первый шаг руководителя при создании эффективной команды - это стать лидером, вокруг которого сможет сплотиться рабочий коллектив. Лидера нельзя назначить.

Лидерство, это в первую очередь, это умение управлять своей собственной жизнью и только потом другими людьми. Сверхвысокое значение коэффициента интеллекта IQ, к сожалению, в этом не поможет. Личная эффективность человека на 80% определяется его коэффициентом эмоционального интеллекта EQ (Emotional Intelligence) [3] – способностью

понимать и эффективно взаимодействовать с другими людьми. Хорошая новость. В отличие от IQ, который формируется в ранней молодости и затем практически не меняется, EQ можно повышать на протяжении всей жизни. Если, конечно, прилагать к этому усилия.

Лидер должен получить признание команды. Для того этого необходимо:

 Признание командой профессиональной компетентности и превосходства лидера.

 Полное доверие команды к действиям и решениям лидера, признание его человеческих качеств, убежденность в его честности, порядочности, вера в его искренность и добросовестность.

Если руководитель не смог стать лидером, он будет вынужден применять в своей практике управленческие антипаттерны. Для такого руководителя характерны чрезмерная настороженность, скрытность, неспособность делегировать полномочия. Он исходит из предпосылок индустриальной эпохи Генри Форда: «Работники ленивы, поэтому им необходимы внешние стимулы для работы. У людей нет честолюбия, и они стараются избавиться от ответственности. Чтобы заставить людей трудиться, необходимо использовать принуждение, контроль и угрозу наказания». Манфред Кетс де Врис называет данное отклонение «параноидальным управлением».

Бесполезно пытаться мотивировать участников команды на успех проекта, не исключив из своего руководящего арсенала практики демотивации – антипаттерны, применение которых в управлении творческими коллективами не приносит ничего, кроме вреда. Вместо мотивирования сотрудников на успех, применение антипаттернов мотивирует их на избежание риска и негативных для себя последствий, подавляет свободу, самостоятельность, творчество и инициативу. Это приводит к деструктивному подчинению, когда все работают строго по инструкции и только в соответствие с указаниями руководства, и полному отсутствию личной ответственности исполнителей, «А какие ко мне претензии? Как сказали, так я и сделал!» В результате - низкая эффективность и качество работы, ухудшение морального климата. Вместо доверия и сотрудничества в коллективе процветают подозрительность и формальное взаимодействие. А вы никогда не видели, как тимлид беседует с программистом только «под протокол» и с подписями на каждом листе? Наконец, применение антипаттернов - это стрессы, усталость участников, личные проблемы, увольнение наиболее профессиональных сотрудников и провал проекта.

Эффективный лидер обязан обладать следующими компетенциями:

 Видение целей и стратегии их достижения.

 Глубокий анализ проблем и поиск новых возможностей

 Нацеленность на успех, стремление получить наилучшие результаты.

 Способность сочувствия, понимания состояния участников команды.

 Искренность и открытость в общении.

 Навыки в разрешении конфликтов.

 Умение создавать творческую атмосферу и положительный микроклимат.

 Терпимость, умение принимать людей какие они есть, принятие их права на собственное мнение и на ошибку.

 Умение мотивировать правильное профессиональное поведение членов команды.

 Стремление выявлять и реализовывать индивидуальные возможности для профессионального роста каждого.

 Способность активно "обеспечивать", "доставать", "выбивать" и т.д.

Не существует одной лучшей стратегии руководства. В зависимости от готовности участников рабочей группы выполнять задания руководителя, он должен использовать одну из 4-х стратегий:

1. «Директивное управление». Руководитель говорит, указывает, направляет, устанавливает. Жесткое назначение работ, строгий контроль сроков и результатов.

2. «Объяснения». Лидер "продает", объясняет, проясняет, убеждает. Сочетание директивного и коллективного управления. Объяснение своих решений.

3. «Участие». Лидер участвует, поощряет, сотрудничает, проявляет преданность. Приоритетное коллективное принятие решений, обмен идеями, поддержка инициативы подчиненных.

4. «Делегирование». Лидер делегирует, наблюдает, обслуживает. «Не мешать» - пассивное управление сформировавшегося лидера.

Применение этих стратегий можно проиллюстрировать на примерах (рисунок 7.1).

1. Вас назначили руководителем в новый коллектив. Вы еще не получили признания, а дело делать надо. Стратегия: «Директивное управление».

2. Вы были участником команды. Вас назначили руководителем этой команды. Доверие есть, а уверенности в правильности ваших действий нет. Стратегия: «Объяснения».

3. Вас назначили руководителем в новый коллектив. Все знают о ваших прежних сложных и успешных проектах. Все признают ваше превосходство, но доверия к вам нет. Никто не знает, какой ценой были достигнуты ваши победы. Стратегия: «Участие».

4. Между вами и участниками установлено взаимное доверие. Все достаточно мотивированы на успех проекта. Каждый сам себе может быть руководителем. Стратегия: «Делегирование».

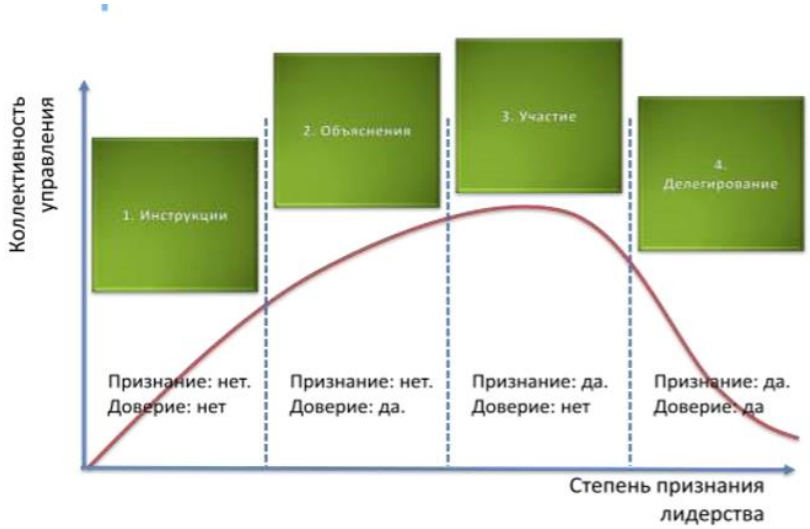

Рисунок 7.1 - Ситуационное лидерство.

Основные усилия руководителя, если он стремится получить наивысшую производительность рабочей группы, должны быть направлены на изучение и изменение объекта управления: людей и их взаимодействия. Следовательно, задачу адаптивного управления мы можем разделить на две подзадачи:

1. Обеспечить эффективность каждого участника рабочей группы.
2. Обеспечить эффективные процессы взаимодействия.

2 Правильные люди

Рональд Рейган говорил: «Окружите себя самыми лучшими людьми, которых вы только сможете найти, передайте им в руки власть и не мешайте им».

Эффективный командный игрок:

 Занимает активную позицию, стремится расширить свою ответственность и увеличить личный вклад в общее дело.

 Постоянно приобретет новые профессиональные знания и опыт, выдвигает новые идеи, направленные на повышение эффективности достижения общих целей, добивается распространения своих знаний, опыта и идей среди коллег.

 Получает удовольствие от своей работы, гордится ее результатами и стремится, чтобы эти же чувства испытывали все коллеги.

 Четко осознает свои личные и общие цели, понимает их взаимообусловленность, настойчиво стремится к их достижению.

 Уверен в себе и в своих коллегах, объективно оценивает их достижения и успехи, внимательно относится к их интересам и мнениям, активно ищет взаимовыгодное решение в конфликтах.

 Является оптимистом, при этом твердо знает, что окружающий мир несовершенен; воспринимает каждую новую проблему, как дополнительную возможность подтвердить собственный профессионализм в своих глазах и во мнении коллег.

Вместе с тем, встречаются патологии поведения, которые, на мой взгляд, неприемлемы в команде:

Вместе с тем, встречаются патологии поведения, которые, на мой взгляд, неприемлемы в команде:

 Непорядочность. Лживость, отсутствие совести и чувства справедливости, способность на низкие поступки.

 Синдром острого дефицита эмпатии. Эгоцентризм. Неуважение и невнимание к партнерам. Склонность к отрицательным оценкам других. Грубость. «Каждый сам за себя! – никто тебе не поможет!» «Человек человеку волк!»

 «Звезданутость». Завышенная самооценка. Подчеркивание собственного превосходства. Умничанье. Человек сильно переоценивает свой личный вклад в общее дело и поэтому считает, что он должен работать меньше, чем его «менее способные» коллеги.

 Вульгарный анархизм. Вольница – это полная безответственность, свобода от каких либо обязательств перед другими, ничем не сдерживаемые проявления чувств, действия или поступки. «Произвольничать, поступать самовольно, в обиду другим, нагло, дерзко» (с) В.Даль. Не путать со «свободой»!

 «Социальный паразитизм». Стремление прожить вольготно за чужой счет там, где ответственность размыта, а личный вклад трудно четко выделить.

Четыре необходимых и достаточных условия для того, чтобы сотрудник эффективно решил поставленную задачу. Это

1. Понимание целей работы.
2. Умение ее делать.
3. Возможность ее сделать.
4. Желание ее сделать.

Для того чтобы обеспечить выполнение этих условий, руководитель должен уметь эффективно выполнять четыре функции:

1. Направлять. Если сотрудник не понимает что делать, задача руководителя - обеспечить общее видение целей и стратегии их достижения.

2. Обучать. Если сотрудник не умеет, задача руководителя – «обучать», быть наставником и образцом для подражания.

3. Помогать. Если у сотрудника не может выполнить работу, задача руководителя – «помогать», обеспечить исполнителя всем необходимым, убрать препятствия с его пути.

4. Вдохновлять. Если у сотрудника не достаточно желание выполнить работу, задача руководителя – «вдохновить», обеспечить адекватную мотивацию участника на протяжении всего проекта.

3 Мотивация

Известно, что ни одна задача не будет решена за любое, отведенное на это время, если человек не захочет ее сделать. Он всегда найдет для оправдания этого 100 «объективных» причин, вместо того, чтобы найти хотя бы один способ решения задачи. У

каждого участника рабочей группы должна быть личная цель (внутренняя мотивация), которую он сможет достичь, продвигая проект к успеху.

Начните с себя! Вам нужно четко понимать, в чем состоит ваш выигрыш в случае успешного завершения проекта. Добиться от участников приверженности проекту больше, чем имеете вы сами, вам не удастся. Если у участника нет такой значимой личной цели, избавьтесь от него. Иначе вам придется потратить все свое время на «промывание его мозгов» и попытки мотивировать его на эффективную работу.

Мотивация должна начинаться с подбора сотрудников в команду. В старой экономике людей нанимали за умения и обучали нужному отношению к делу. В новой экономике необходимо поступать с точностью до наоборот: нанимать за нужное отношение к делу и учить необходимым умениям.

Люди не рождаются победителями, они ими становятся. Кандидата стоит нанимать только в случае, если вы можете предложить ему возможность стать победителем. Настоящий лидер предлагает не работу, а возможности.

Все люди разные, а ситуаций, в которых они могут находиться в ходе проекта, бесчисленное множество. Бойтесь стереотипов. Если вы не учитываете индивидуальные особенности конкретной личности, то эффективность ваших взаимодействий сильно снижается. Модель объекта управления нам неизвестна, следовательно, не может существовать исчерпывающий набор правил, типа «если…, то…», по которым смог бы действовать руководитель. Поэтому, сколько людей и ситуаций, столько и вариантов решений должен иметь эффективный руководитель в своем запасе. «Если у руководителя в руках только молоток, то все вокруг будут похожи на гвозди».

Руководитель при поиске решения опирается на свой багаж знаний и умений. Он пытается понять каждого участника, классифицировать состояние, найти в своем опыте похожую ситуацию и адаптировать ранее использованное успешное решение применительно к данному конкретному случаю. Таким образом, руководитель стремится помочь человеку (объекту управления) перейти в новое более эффективное с точки зрения целей проекта состояние.

Затем руководитель должен наблюдать за результатами своего воздействия – это и есть дополнительный контур обратной связи. Необходимо помнить, что понять человека можно, только слушая и слыша, что он говорит. Руководитель, который в течение недели не пообщался индивидуально с каждым из своих прямых подчиненных, зря получает зарплату. И совсем не обязательно разговор должен идти о статусе проектных работ. Порой, достаточно поговорить о погоде, кино или футболе. После этого руководитель анализирует полученные результаты и аккумулирует новый опыт (положительный или отрицательный) в своей «базе знаний».

Чем опытней руководитель, тем точнее он может распознать и классифицировать сложившуюся ситуацию, тем больше в его «базе знаний» прецедентов, используя которые, он может синтезировать решение для данного конкретного случая. Именно поэтому в управлении программными проектами в первую очередь ценится опыт руководителя и только потом, возможно, его звания и знания.

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

1. Телу необходимы деньги и безопасность.
2. Сердцу - любовь и признание.
3. Разуму – развитие и самосовершенствование.
4. Душе – самореализация.

Предоставьте все это вашим сотрудникам, и эффективность их труда возрастет многократно. В противном случае люди, которые хотят побеждать, найдут все это в другой команде, а в вашей останутся только неудачники.

Эффективное взаимодействие

Ранее мы уже говорили, что процесс производства программного обеспечения, применяемый в проекте, должен основываться на итеративности, инкрементальности,

самоуправляемости команды и адаптивности. Главный принцип: не люди должны строиться под выбранную модель процесса, а модель процесса должна подстраиваться под конкретную команду, чтобы обеспечить ее наивысшую производительность.

В программной инженерии многие уже признали, что наиболее эффективные производственные процессы складываются в самоуправляемых и самоорганизующихся рабочих командах. Идеи командного менеджмента на Западе зародилась в начале 80-х годов. Эффективность команд в новых экономических условиях одними из первых оценили такие гиганты, как Procter & Gamble и Boeing. Доктрина командного менеджмента предполагает ясность общих ценностей и целей, самоорганизацию и самоуправление совместной деятельностью, взаимный контроль, взаимопомощь и взаимозаменяемость, коллективную ответственность за результаты труда, всемерное развитие и использование индивидуального и группового потенциалов.

Если на Западе идеи командного менеджмента только зарождаются, то для России они имеют давние традиции. С XIII века в России существовали производственные артели - различные формы добровольных объединений людей с целью осуществления общей хозяйственной деятельности. Артель была добровольным товариществом совершенно равноправных работников, призванных на основе взаимопомощи и взаимовыручки решать практически любые хозяйственные и производственные задачи. Известный российский писатель и революционер, А.И. Герцен видел в коллективизме, в уникальности существующей сельской общины, в городской артели и в самобытной воинской организации казачества характерную черту жизни и психологии русского народа. Позднее во времена СССР широкое распространение на производстве получили рабочие бригады, а в научно-исследовательских институтах - временные научные творческие коллективы, которые объединялись на основе общности цели, взаимопомощи и коллективной ответственности за результат.

Истинный российский коллективизм (соборность) не имеет ничего общего с вульгарным коллективизмом: со стадностью, подавлением личности («я нет, есть только мы!», «я – последняя буква алфавита!»), уступчивостью, групповой психологией и слепым подчинением меньшинства большинству. Суть российского коллективизма в том, что человек ощущает себя элементом органичной системы, где он выполняет свою функцию, свою задачу, совершенно особенную, которую не выполняет никто другой и выполнить не может. Эту задачу он выполняет совершенно сознательно, стремясь к максимальной реализации своих целей.

Руководителю недостаточно стать лидером, надо еще суметь сплотить команду. Эксперты в области командного менеджмента выделяют 4 обязательные последовательные стадии, через которые должна пройти рабочая группа прежде, чем она станет эффективной командой.

1. Forming. Формирование. Характеризуется избытком энтузиазма, связанного с новизной. Люди должны преодолеть внутренние противоречия, переболеть конфликтами прежде, чем сформируется действительно спаянный коллектив. На этом этапе многое зависит от руководителя. Он должен четко поставить цели членам команды, верно определить роль каждого в проекте.

2. Storming. Разногласия и конфликты. Самый сложный и опасный период. Мотивация новизны уже исчезла, а сильные и глубокие стимулы у команды еще не появились. Неизбежные сложности или неудачи порождают конфликты и «поиск виновных». Участники команды методом проб и ошибок вырабатывают наиболее эффективные процессы взаимодействия. Руководителю на этом этапе важно обеспечить открытую коммуникацию в команде. Конфликты не следует прятать или разрубать. Споры необходимо разруливать спокойно, терпеливо и тщательно.

3. Norming. Становление. В команде растет доверие, люди начинают замечать в коллегах не только проблемные, но и сильные стороны. Закрепляются и оттачиваются наиболее эффективные процессы взаимодействия. На смену битве амбиций приходит

продуктивное сотрудничество. Четче становится разделение труда, исчезает дублирование функций. Руководитель перестает находиться в состоянии постоянного аврала, работа по построению команды на этом этапе – уже не тушение пожара, а скрупулезный труд по отработке общих норм и правил.

4. Performing. Отдача. Команда работает эффективно, высок командный дух, люди хорошо знают друг друга и умеют использовать сильные стороны коллег. Все стремятся придерживаться выработанных общих процессов. Высок уровень доверия. Это лучший период для раскрытия индивидуальных талантов.

Часто случается, что рабочая группа вязнет на одной из стадий и никогда не достигает плато наивысшей производительности.

Если команда прошла все стадии формирования и вышла на фазу «Performing», не стоит полагать, что менеджер проекта может делегировать все свои полномочия и отправиться в отпуск. Задача менеджера на этом этапе - «точить пилу». Это значит поддерживать требуемый уровень мотивации, быть штурманом, искать новые пути и открывать новые возможности. Постоянно наблюдать и оценивать эффективность всех процессов, применяемых в проекте. Искать ответ на вопросы: «Что угрожает проекту?» «Что лишнее мы делаем?» «Что можно делать проще?» Работать на сокращение ненужных усилий вместо того, чтобы «стремиться к новым героическим подвигам».

Если руководитель не будет прилагать дополнительные усилия команда, рано или поздно, начнет «сползать» с плато наивысшей эффективности в состояние застоя и стагнации (рисунок 7.2). Помните, что окружение и команда изменяются по ходу проекта. Прежняя мотивация ослабевает или перестает действовать. Изменяйте правила и процессы. Отказывайтесь от того, что перестало действовать или стало работать неэффективно. «Встряхивайте» (Reforming) и возвращайте команду в стадию «Forming». Это позволит ей снова, пройдя через все этапы становления, выйти на новый более высокий уровень производительности. Разумеется, делать это следует, после сдачи очередного релиза программного продукта, ну и, возможно, в случае глубокого кризиса проекта.

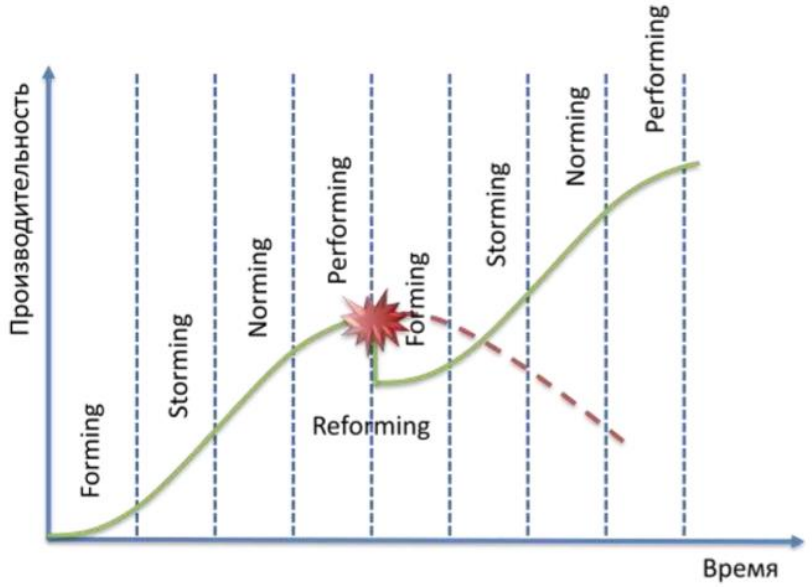

Рисунок 7.2 - Reforming. «Встряхивание» и перевод команды проекта на новый, более высокий, уровень производительности.

Выводы

Эффективные команды не образуются сами по себе, они кристаллизуются вокруг признанного лидера. Для того чтобы стать лидером, необходимо:

 Признание командой профессиональной компетентности и превосходства лидера.

 Полное доверие команды к действиям и решениям лидера, признание его человеческих качеств, убежденность в его честности, порядочности, вера в его искренность и добросовестность.

Мотивация должна начинаться с подбора сотрудников в команду. В старой экономике людей нанимали за умения и обучали нужному отношению к делу. В новой экономике необходимо поступать с точностью до наоборот: нанимать за нужное отношение к делу и учить необходимым умениям.

Рабочая группа прежде, чем она станет эффективной командой, должна пройти четыре обязательных последовательных стадии: 1) Forming, 2) Storming, 3) Norming, 4) Performing.

Если руководитель не будет прилагать дополнительные усилия команда, рано или поздно, начнет «сползать» с плато наивысшей эффективности в состояние застоя и стагнации. Задача менеджера на этом этапе - «точить пилу»: поддерживать требуемый уровень мотивации, быть штурманом, искать новые пути и открывать новые возможности. Четыре стадии развития команды должны циклически повторяться, чтобы обеспечить непрерывный рост производительности.

Тема 8 Реализация проекта

1 Рабочее планирование

Управление - это расчленение, анализ, определение последовательности действий, конкретная реализация. Управление фокусируется на нижнем уровне: как мне сделать это наилучшим образом? Эта компетенция руководителя определяет эффективность движения по выбранному пути. Как правило, менеджеры, вышедшие из программистов, без особого труда овладевают необходимыми управленческими навыками.

Базовое расписание, составленное на этапе планирования проекта, служит ориентиром для мониторинга состояния дел на макроуровне. Для оперативного управления проектом используется рабочий план. Рабочее планирование рекомендуется выполнять методом «набегающей волны»: работа, которую надо будет выполнить в ближайшей перспективе, подробно планируется на низшем уровне ИСР, а далеко отстоящая работа планируется на сравнительно высоком уровне ИСР.

Элементарная работа, как правило, представляет собой отдельное функциональное требование к программному продукту или запрос на изменение, над которым последовательно работают: бизнес-аналитик, проектировщик, разработчик, тестировщик и документалист. Трудоемкость элементарной работы каждого из исполнителей должна быть от 4 до 20 чел.*час. Если трудоемкость задачи не укладывается в эти пределы, следует провести декомпозицию работы.

Для рабочего планирования целесообразно использовать систему управления задачами или багтрекинга, поскольку она позволяет задавать последовательность переходов задачи от исполнителя к исполнителю, управлять приоритетами работ и адекватно отслеживать их статус: анализ, проектирование, кодирование, тестирование, документирование. Работа должна считаться законченной только тогда, когда реализация требования протестирована и документирована.

В зависимости от уровня профессионализма и зрелости команды проекта распределение работ может осуществляться либо директивно с жесткой постановкой срока и контролем исполнения каждой задачи, либо эти полномочия делегируются исполнителям. В этом случае они сами выбирают задачи последовательно в соответствие с приоритетами, а их выполнение анализируется периодически на статус митинге. Можно рекомендовать еженедельные собрания по статусу проекта всей команды или, если проект достаточно большой, то ключевых его частников: руководителей подпроектов и лидеров команд. Хорошее время для этого утро понедельника, поскольку участники проекта, особенно

студенты, которые совмещают учебу и работу, часто работают в выходные, но, разумеется, не потому, что аврал, а потому что им так удобнее. Обсуждаются, как правило, всего три вопроса:

 Угрозы и проблемы.

 Анализ результатов за неделю.

 Уточнение приоритетов задач на новую неделю.

Как правило, нет смысла оценивать процент реализации работы в промежуточном состоянии, поскольку, если задача передана в тестирование, то это вовсе не означает, что 70% работы сделаны. На этапе тестирования может быть выявлена ошибка проектирования и вся работа начнется заново. Рекомендация - использовать правило «50/100». Если работа по задаче начата, то следует учитывать ее, как выполненную на 50%. А 100% поучает только протестированная и документированная работа.

2 Принципы количественного управления

«Тем, что нельзя измерить, нельзя управлять». Измерения по проекту необходимо выполнять регулярно, не реже одного раза в 1-2 недели. Для каждого измеримого показателя должны быть определены его плановые значения. Для каждого планового значения должны быть определены три области критичности отклонений:

 Допустимые отклонения. Предполагается, что никаких управляющих воздействий не требуется.

 Критичные отклонения. Требуется тщательный анализ причин отклонения и при необходимости применение корректирующих действий.

 Недопустимые отклонения. Требуется срочный анализ причин отклонения и обязательное применение корректирующих действий.

Измерения необходимо производить регулярно. Цель – выявить причины наступивших или возможных критичных и недопустимых отклонений. Результатом анализа должны стать планирование корректирующих действий по компенсации недопустимых отклонений, их реализация и мониторинг результативности применения этих корректирующих действий.

Все измерения необходимо сохранять в репозитарии проекта. Измерения, накопленные в ходе проекте, являются наиболее достоверной основой при детальной оценке и планировании работ на следующих итерациях проекта.

Поскольку главная задача менеджера удержать проект в пределах «железного» треугольника, то, в первую очередь, необходимо анализировать отклонения проекта по срокам и затратам. Делается это при помощи метода освоенного объема [1]. Приходилось сталкиваться с мнением, что этот метод не применим в управлении программными проектами. Это действительно так, если мы используем «водопадную» модель процесса разработки. Но если ИСР проекта, ориентирована на инкрементальную разработку, то это означает, что на верхних уровнях декомпозиции находятся компоненты проектного продукта и их функционал, а не производственные процессы. Следовательно, если в проекте реализованы, протестированы и документированы 50 % функциональных требований, то есть все основания полагать, что осталась приблизительно половина проектных работ.

Суть метода оценки проекта по освоенному объему заключается в следующем. Сначала оценивается отклонение от графика SV (Shedule Variance) в денежных единицах:

$$SV = EV – PV,$$

где

EV (Earned Value) - освоенный объем. Плановая стоимость выполненных работ. Объем выполненных работ, выраженный в терминах одобренного бюджета,

выделенного на эти работы для плановой операции и элемента иерархической структуры работ;

PV (Planned Value) - плановый объем. Плановая стоимость запланированных работ.

Утвержденный бюджет, выделенный на плановые работы, выполняемые в рамках плановой операции или элемента иерархической структуры работ.

Например. Пусть мы на текущий момент реализовали (протестировали и документировали) 20 функциональных требований, на каждое из которых было запланировано затратить по 40 чел.*час. по 1000 руб, то освоенный объем будет

$$EV = 20 * 40 * 1000 = 800\ 000 \text{ руб.}$$

Если же на текущий момент планировалось реализовать только 15 требований, то плановый объем будет

$$PV = 15 * 40 * 1000 = 600\ 000 \text{ руб.}$$

Следовательно, мы опережаем график (отклонение от графика положительное) на величину

$$SV = EV – PV = 800\ 000– 600\ 000 = 200\ 000 \text{ руб.}$$

Как пересчитывается отклонение от графика, выраженное в денежных единицах, в сокращение сроков проекта иллюстрируется на Рисунок 43.

Если мы опережаем график, то это не обязательно означает что проект идет успешно. Хорошо это или плохо зависит от значения другого показателя метода освоенного объема: CV (Cost Variance) - отклонения по затратам, которое оценивается по формуле:

$$CV = EV – AC$$

где AC, (Actual Cost) - фактические затраты. Фактическая стоимость выполненных работ. Фактические затраты на выполнение работ за определенный период в рамках плановой операции или элемента иерархической структуры работ.

Например, если мы для того что сократить время работ по проекту работали 25% времени сверхурочно и в выходные дни с двойной оплатой, то фактические трудозатраты составили:

$$AC = 20 * (30 * 1000 + 10 * 2000) = 1\ 000\ 000 \text{ руб.}$$

Поэтому отклонения по затратам в нашем случае будет

$$CV = EV – AC\ PV = 800\ 000 – 1\ 000\ 000 = -\ 200\ 000 \text{ руб.}$$

Отрицательное значение отклонения по затратам означает, что мы превысили бюджет, что, в общем случае, не очень хорошо. Но если срок завершения проекта для нас имеет высший приоритет, и наши прогнозируемые затраты по завершению проекта не превышают плановых с учетом управленческого резерва (рисунок 8.1), то в этом случае можно считать, что проект выполняется успешно.

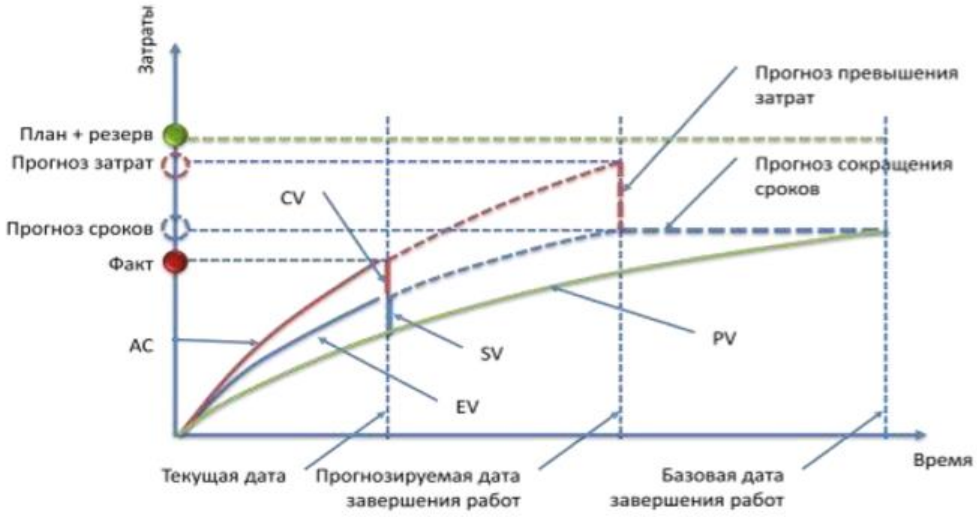

Рисунок 8.1 - Оценка и прогноз показателей по методу освоенного объема

Отклонение от бюджета и по срокам в абсолютных денежных единицах недостаточно для характеристики проектов разных масштабов. Более наглядны относительные

показатели: индекс выполнения сроков SPI (Schedule Performance Index)

$$SPI = EV / PV$$

и индекс выполнения стоимости CPI (Cost Performance Index)

$$CPI = EV / AC,$$

которые характеризуют проект независимо от его размера. Если значения обоих индексов больше 1, то это свидетельствуют о благополучном состоянии в проекте.

Какие еще измеримые показатели целесообразно применять в управлении программным проектом?

В первую очередь это показатель прогресса проекта, доля реализованных и проверенных высокоуровневых требований к проекту, например отношение числа завершенных сценариев использования продукта к их общему числу.

Другой показатель - стабильность проекта, общее количество принятых (утвержденных спонсором или заказчиком) изменений в плане управления проектом. Чем выше нестабильность в проекте, тем больше сложность в управлении работами и ниже производительность участников.

Если кто-то думает что код это решение проблемы, то это не так. Код это новый источник проблем. Поэтому всегда следует измерять текущий размер проекта - количество строк исходного кода, добавленных, измененных и удаленных в ходе выполнения проекта разработки ПО. Чем больше объем исходного кода, тем больше времени потребуется на внесение изменений и исправление ошибок.

При увеличении объема проектного продукта трудозатраты на каждую новую строку исходного кода увеличиваются. Если за номинал взять производительность проектной команды при производстве продукта в 10 KSLOC, то та же команда на проекте в 100 KSLOC покажет производительность в 1.3 – 1.7 раз меньшую, а на проекте в 1000 KSLOC следует ожидать, что производительность снизится в 1.6 – 3.0 раза.

Большой объем кода так же потребует большего количества людей на его сопровождение. Поскольку, даже если будет выявляться только несколько критических ошибок в год, то для того чтобы их исправить в приемлемые сроки, например, за 24 часа, в продукте общим объемом в 1000 KSLOC то один программист с этим не справится. Это связано с тем, что для того чтобы исправить ошибку в ограниченные сроки необходимо оперативно выявить и устранить ее причину, а для этого надо хорошо знать архитектуру и код программного продукта. Чтобы эффективно сопровождать продукт подобного объема необходимо иметь в «горячем» резерве примерно 20 разработчиков, потому что 50 KSLOC, на мой взгляд, это предельный объем кода, который может удерживать в голове и эффективно сопровождать один человек. И еще проблема: чем этих людей занимать в свободное от исправлений ошибок время, если нет новых проектов развития продукта.

Следующий важный показатель состояния проекта – это средняя производительность, отношение текущего размера проекта к фактическим затратам по проекту. С. Макконнелл2] приводит следующие показатели (минимальное, максимальное и среднее значение) производительности в KSLOC на один чел.*мес. фактических затрат для стандартных типов проектов объемом в 100 KSLOC:

- 300-7000 (800) - интранет система.
- 200-7000 (600) - бизнес система.
- 100-2000 (300) - Интернет система.
- 50-600 (100) - системное ПО, телекоммуникации.
- 20-300 (50) - системы реального времени.

Высокая производительность в проекте – это далеко не всегда хороший признак. Приходилось встречаться с проектами, в которых вследствие активного применения метода «copy+past», средняя производительность в разработке бизнес системы достигала 2000 SLOC/чел.*мес. Однако для реализации требуемого функционала было написано в 3 – 4 раза больше кода, чем это могло бы потребоваться при адекватной проработке архитектуры.

Еще одна группа количественных показателей, которые следует наблюдать в ходе

реализации проекта, характеризует качество программного продукта:

 Дефектность продукта – количество выявленных дефектов на единицу объема продукта (например, KSLOC).

 Доля не устраненных дефектов - отношение количества незакрытых максимально критичных и критичных дефектов к количеству выявленных несоответствий.

 Средние затраты на сопровождение – средние трудозатраты на исправление одного дефекта. Высокое значение этого показателя может свидетельствовать о некачественной архитектуре программного продукта.

 Документированность кода - определяет процент строк исходного кода с комментарии по отношению к общему количеству строк.

Следует подчеркнуть, что наблюдать надо за средними по проекту значениями показателей, и ни в коем случае не пытаться измерять индивидуальные характеристики производительности и качества. Главные причины, почему это не следует делать, заключаются в том, что, во-первых, в этом случае вместо слаженной командной работы мы получим личную конкуренцию, а, во-вторых, наиболее «продвинутые» разработчики станут работать на формальные показатели, а не на достижение целей проекта.

Если команда действительно состоялась, то для нее характерна коллективная ответственность за достижение общих целей. И, как пишет, Т.Демарко [3], «менеджер проекта должен занимать очередь, чтобы покритиковать сотрудника, не выполняющего свои обещания», поскольку в правильной команде для этого всегда найдется масса желающих.

3 Завершение проекта

Главная цель этой фазы – проверить и передать заказчику результат проекта. Для этого необходимо выполнить приемо-сдаточные работы в соответствии с процедурой приемки, которая должна быть определена заранее на самой ранней стадии проекта.

Результаты проекта должны быть переданы во внедрение или сопровождение, или должным образом законсервированы для дальнейшего использования. Не должно оставаться «зависших» работ по проекту. Все линейные руководители всех участников должны быть извещены о завершении работ по проекту, и освобождении сотрудников.

Важная задача, которая должна быть решена на данной фазе, это реализация обратной связи по проекту. Цель – сохранить результаты, знания и опыт, полученные в проекте, для более эффективного и качественного выполнения аналогичных проектов в будущем. Необходимо архивировать все результаты, документировать опыт, уроки по проекту и предложения по улучшению технологии выполнения работ и управления проектами.

Все проекты и в особенности провальные проекты должны завершаться итоговым отчетом, если компания не хочет «наступать на одни и те же грабли». Помним о том, что «вчерашние проблемы, это сегодняшние риски».

Итоговый отчет должен содержать следующую информацию:

 Итоги проекта:

o Достижение целей проекта

o Дополнительные полезные результаты

o Фактические сроки

o Фактические расходы

o Обоснование отклонения от целей

o Отклонения результатов от требований

 Уроки проекта

o Проблемы проекта и способы их решения

o Материалы программные компоненты для последующего использования

o Предложения по изменению технологий или стандартов компании

На фазе завершения желательно реализовать и план мотивации участников проектной команды, поскольку отложенное вознаграждение мотивирует существенно слабее.

Выводы

Для оперативного управления проектом используется рабочий план. Элементарная работа, как правило, представляет собой отдельное функциональное требование к программному продукту или запрос на изменение, над которым последовательно работают: бизнес-аналитик, проектировщик, разработчик, тестировщик и документалист.

Измерения по проекту необходимо выполнять регулярно, не реже одного раза в 1-2 недели. Для каждого измеримого показателя должны быть определены его плановые значения и допустимые отклонения.

В состав измеряемых показателей должны входить следующие характеристики проекта:

- Освоенный и плановый объемы работ и фактические затраты по проекту.
- Показатели прогресса и стабильности проекта.
- Размер продукта.
- Производительность.
- Показатели качества программного продукта.

По результатам проекта обязательно должна быть реализована обратная связь. Цель – сохранить результаты, знания и опыт, полученные в проекте, для более эффективного и качественного выполнения аналогичных проектов в будущем.

Тема 9 Программное обеспечение в области управления проектами

1 Microsoft Project для управления проектами

MS Project из настольного приложения для менеджера трансформировался в программное обеспечение серверного типа для автоматизированного управления деятельностью команды проектировщиков в рамках корпоративной системы управления проектами.

Линейка продуктов MS Project состоит из четырех продуктов: Microsoft Project Standard (стандартная редакция MS Project), Microsoft Project Ptofessional (профессиональная редакция MS Project), Microsoft Project Server и Microsoft Project Portfolio Server. Продукт Microsoft Project Portfolio Server применяют для управления портфелями проекто.

Приложение MS Project является специфическим программным обеспечением, предназначенным исключительно для управления проектами, в отличие от других приложений семейства Microsoft Office, ориентированных на максимально широкую область применения. Поэтому без знания теоретических основ управления проектами эффективно работать с MS Project невозможно.

2 Составление плана работ

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

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

задачи.

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

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

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

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

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

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

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

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

Ограничения определяют способ привязки задач к датам (Проект|Сведения о

задаче|Дополнительно|Тип ограничения):

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

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

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

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

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

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

3 Планирование ресурсов

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

Независимо от типа ресурс может быть бюджетным или внебюджетным (управляется

с помощью флага Проект|Сведения о ресурсе|Общие|Бюджет).

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

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

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

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

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

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

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

4 Учет вероятности выполнения работ

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

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

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

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

Анализ плана проекта методом критического пути позволяет найти те задачи, изменение длительности которых приведет к изменению сроков проекта. Критический путь – это множество задач, определяющие дату окончания проекта. MS Project может относить задачи к критическим не только, если ее временной резерв меньше, либо равен нулю, но и близок к нулю (Сервис| Параметры| Расчеты| Считать критическими задачи, имеющие резерв не более … дней). Для отображения критического пути можно воспользоваться мастером диаграмм Ганта (Формат| Мастер диаграмм Ганта). На втором шаге нужно выбрать «Критический путь». После этого критический путь будет выделен красным. Изменение длительностей задач, может исключить их из критического пути, может сделать критическими другие задачи.

Помимо общей стоимости проекта («Общие затраты» для суммарной задачи) при анализе структуры затрат обычно рассматриваются:

- Распределение затрат по фазам проекта. Чтобы видеть процент затрат на фазу, удобно ввести дополнительное поле в таблицу (Вставка| Столбец), которое будет вычисляться по формуле (Сервис| Настройка| Поля| Формула).

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

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

5 Учет стоимости работ

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

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

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

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

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

Для планирования динамики освоения бюджета следует определить не только стоимость работ, но и порядок оплаты: предоплата, оплата по факту завершения или оплата по мере выполнения. Способ оплаты задается и для ресурсов (Проект| Сведения о ресурсе| Затраты| Начисление затрат), и для фиксированных затрат (Вид| Таблица| Затраты| Начисление фиксированных затрат). Необязательно порядок оплаты должен совпадать с

договорным. Например, если по договору оплата производится по окончании работ, но неизвестно, когда они завершатся, то имеет смысл выбрать порядок оплаты в начале работ, чтобы средства были запланированы, и можно было расплатиться в любой момент.

Если труд сотрудников оплачивается не по сдельной схеме, а в форме оклада, то это можно учесть следующим образом. В отдельной таблице норм затрат для каждого сотрудника зафиксировать зарплату в поле «затраты на использование». Далее на определенный день запланировать задачу «выдача зарплаты» и назначить на нее всех сотрудников, получающих зарплату.

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

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

3 ПРАКТИЧЕСКИЙ РАЗДЕЛ

1. Управление проектами.
2. Инициация проекта.
3. Планирование проекта.
4. Управление рисками проекта.
5. Оценка трудоемкости и сроков разработки ПО.
6. Формирование команды.
7. Реализация проекта.
8. Программное обеспечение в области управления проектами.

Практическое занятие 1 -2 Управление проектами

Задание 1. Дать собственное определение понятиям проект и управление проектом на основе обобщения существующих. Например:

Проект – это уникальное и временное начинание, с определенным началом и концом с целью создать или модифицировать определенный продукт или услугу.

Управление проектом – это область менеджмента, охватывающая те сферы, в которых создание продукта реализуется как уникальный комплекс работ при определенных требованиях к срокам, бюджету и характеристикам результата.

Задание 2. Определить какая деятельность является проектом, а какая – нет.

Организация вечеринки; внедрение новой процедуры подбора персонала компании; уборка квартиры; замена информационной системы по учету труда и заработной платы компании; покраска крупного моста; возведение монумента на площади; повторяющиеся (рутинные) операции предприятия; организация олимпиады в Токио в 2020 г., постройка офисного здания; апгрейд планшета производителем; разработка и вывод на рынок инновационного продукта; замена аппаратного (ПК) и программного обеспечения учебной аудитории ВУЗа; составление ежегодных финансовых отчетов предприятия; осуществление изменений в оргструктуре и кадровом составе организации, строительство Титаника.

Задание 3. Задание по кейсу: продумайте Ваш предпринимательский проект в условиях города? Зафиксируйте данную проектную инициативу в следующем документе:

МОДЕЛЬ ПРОЕКТА «_____»

Отразите:

1. Сущность проекта
2. Какую проблему решает проект?
3. Основные цели, результаты (продукты проекта) и требования к ним
4. Состав работ проекта (описать конкретные действия в ходе реализации проекта)

5. Риски проекта

6. Оценить доход от проекта

Концепция проекта должна отражать, что Вы хотите сделать в проекте, зачем и как Вы это сделаете. Каждая группа должна представить концепцию своего проекта в презентации Power Point.

Практические занятия 2-3. Инициация и планирование проекта

По выбранному проекту предоставить (в презентации Power Point) следующую информацию:

Задание 1. Общая информация о проекте (аннотация):

- Наименование проекта
- Менеджер проекта
- Даты начала и окончания, длительность проекта
- Причины инициации проекта (обоснование)
- Цели
- Продукты/результаты проекта и требования к ним
- Оценка бюджета проекта
- Список заинтересованных сторон

Задание 2. Состав работ проекта

Представить иерархическое разбиение всей работы, которую необходимо выполнить для достижения целей проекта, как показано на рисунке 1. В упрощённом варианте пакетов работ может не быть, если Вы используете только три уровня иерархии.

Задание 3. Расписание проекта

Используя составленную иерархическую структуру работ по проекту, составить упрощённое расписание проекта в таблице MS Excel, как показано в таблице 1.

Таблица 1 – Структура работ

ID	Наименование	Предшественник	Начало (дата)	Конец (дата)	Длительность (дней)
	Проект				
1	Создание результата 1.				
1.1	Пакет работ 1.1.- ...				
1.1.1					
1.1.2					
1.2					
2.	Создание результата 2.				
2.1					
2.2					
2.2.1					

Задание 4. Исполнение и контроль исполнения проекта

Выполнить запланированные работы и осуществить контроль исполнения проекта, ответив на вопросы:

- Все ли работы выполнены?
- Получены ли запланированные продукты/результаты проекта?

- Соответствуют ли продукты/результаты проекта требованиям к ним?
- Соблюдено ли расписание и бюджет проекта?

Практическое занятие 4 Управление рисками проекта

Задание 1 Оцените основные риски проекта, составив матрицу рисков.

Таблица 1 - Матрица оценки риска

Событие	Вероятность	Степень серъезности	Трудность обнаружения	Время

Задание 2 Составить план управления рисками проекта:

1. Иерархическая структура рисков - перечисляет источники рисков либо категории и подкатегории, в рамках которых могут возникать риски

2. Методом мозгового штурма получить список идентифицированных рисков и предполагаемые даты их наступления

3. Ранжирование рисков по значимости с использованием Матрицы вероятностей и влияний (последствий) рисков. В её ячейках - ранги рисков, определяющие их значимость для дальнейшего планирования реагирования

Влияние (последствия)/ Вероятность	Очень высокое (4)	Высокое (3)	Среднее (2)	Низкое (1)
Очень высокая (4)	Очень высокий (4*4=16)	Очень высокий (4*3=12)	Высокий (4*2=8)	Высокий (4*1=4)
Высокая (3)	Очень высокий (3*4=12)	Высокий 3*3=9)	Высокий 3*2=6)	Средний (3*1=3)
Средняя (2)	Высокий (2*4=8)	Высокий (2*3=6)	Средний (2*2=4)	Средний (2*1=2)
Низкая (1)	Высокий (1*4=4)	Средний (1*3=3)	Средний(1*2=2)	Низкий (1*1=1)

Рисунок 1 – Матрица вероятностей и влияний (последствий)

4. Количественный анализ наиболее значимых рисков

5. Упреждающие мероприятия (реакции) по снижению вероятностей наступления наиболее значимых рисков и их последствий/влияний в случае наступления

6. Мероприятия (реакции) по уменьшению последствий на случай возникновения наиболее значимых рисков

7. Реестр рисков.

Таблица 2 - – Шаблон реестра рисков проекта

Риск	Категория (Источник)	Вероятность наступления (% и по шкале от 1 до 4)	Степень влияния последствия но шкале от 1 до 4	Рейтинг	Упреждающие мероприятия по снижению вероятности наступления
Уход из проекта ключевою члена проектной команды	Организационный риск: потеря ресурса	60%, 3-высокая	3 - высокое	9(=3*3)-высокий	Увеличить его оплату труда
Несвоевременное финансирование	Организационный: финансирование	20%, 1 - низкая	3 - высокое	3(-1*3)-среднип	отсутствуют

Практическое занятие 5 Оценка трудоемкости и сроков разработки ПО

Задание

1 Составьте смету бюджета проекта.

2 Оцените трудоемкость проекта.

3 Оцените сроки разработки ПО.

Практическое занятие 6 Формирование команды

Задание 1 Выявите заинтересованные стороны (шаблон - рисунок 1) и составить матрицу ответственности RACI (шаблон - таблица 1) по проекту

Рисунок 1 – Заинтересованные стороны проекта

Таблица 1 - Матрица ответственности RACI

Пакеты или этапы работ по проекту	Роли заинтересованных сторон (ФИО или должность)			
	Спонсор/куратор Менеджер проекта проекта (начальник (стажёр FPT подразделения ГГ в FPT) Marge)		Заказчик	
…	R		I	
Разработка бюджета проекта	А, Б, С			

R – Responsible (Исполнитель), ответственный за выполнение, должен быть минимум один по каждому пакету или этапу работ;

A – Accountable (Утверждает), перед ним производится отчет в полученном результате (принимает работу у исполнителя), должен быть один на пакет/этап работ;

C – Consult before doing (Согласует) - с ним необходимо предварительное согласование/консультация перед началом выполнения;

I – Inform after doing (Информируемый) оповещается после исполнения.

ПРИМЕРНЫЙ СПИСОК РАБОТ ПО СОЗДАНИЮ ВЕБСАЙТА И ВИДЕОУРОКОВ

При составлении списка работ матрицы RACI необходимо учесть, что работу по созданию и поддержке вебсайта можно разделить на следующие этапы:

— Подготовительный (создание Технического Задания на разработку);
— Разработка дизайн-макета;
— Верстка;
— Программирование;
— Наполнение контентом (информацией);
— Расположение сайта в сети Интернет;
— Тестирование сайта
— Раскрутка сайта
— Администрирование (поддержка) сайта

Задание 2 Исходя из того, что проектная команда уже сформирована и не нуждается в развитии, описать каким образом будет осуществляться управление её членами, в т.ч. составить план по контролю и мотивации

Практическое занятие 7 Реализация проекта

Задание 1 Оцените освоенный объем проекта. Сравните с плановым объемом. Оцените фактические затраты. Рассчитайте индекс выполнения сроков SPI и индекс выполнения стоимости CPI.

Задание 2 Подготовите итоговый отчет проекта, включающий в себя достижение целей проекта дополнительные полезные результаты, фактические сроки, фактические расходы, обоснование отклонения от целей, оклонения результатов от требований.

Практическое занятие 8 Программное обеспечение в области управления проектами

Используя MS Project для анализа проекта

З

а Сколько работ находится на критическом пути? (Фиктивные работы не ***у*** учитываются.) На сколько недель можно отложить выбор верстку шаблонов ***с*** страниц? Чему равно наиболее позднее время завершения работы запуску и ***н*** сопровождению?

и ***Задание*** 2 Учитывая вероятностный характер длительности задач, ***н*** постройте сеть *PERT/CPM* для проекта и определите: Каков ожидаемый срок завершения проекта? Чему равна риск (корень квадратный из дисперсии) ~~Найдите критический путь проекта. Какова вероятность~~ время критического пути проекта? Какова вероятность того, что проект будет выполнен за 46 недели?

Задача 3 Для проекта известно время выполнения работ и затраты. Определите:

Каковы затраты на выполнение проекта при нормальной продолжительности работ?

За какое минимальное время может быть выполнен проект?

4 РАЗДЕЛ КОНТРОЛЯ ЗНАНИЙ

4.1 Вопросы к экзамену по учебной дисциплине «Управление проектами в сфере информационных технологий»

1. История и основные понятия.
2. Отличия программной инженерии от других отраслей.
3. Эволюция подходов к управлению программными проектами.
4. Модели процесса разработки ПО.
5. Проект - основа инноваций.
6. Критерии успешности проекта.
7. Проект и организационная структура компании.
8. Организация проектной команды.
9. Жизненный цикл проекта.
10. Фазы и продукты.
11. Управление приоритетами проектов.
12. Концепция проекта.
13. Цели и результаты проекта.
14. Допущения и ограничения.
15. Ключевые участники и заинтересованные стороны.
16. Ресурсы. Сроки. Риски. Критерии приемки.
17. Обоснование полезности проекта.
18. Уточнение содержания и состава работ.
19. Планирование управления содержанием.
20. Планирование организационной структуры.
21. Планирование управления конфигурациям.
22. Планирование управления качеством
23. Базовое расписание проекта.
24. Планирование управления рисками.
25. Идентификация рисков.
26. Качественный анализ рисков.
27. Количественный анализ рисков.
28. Планирование реагирования на риски.
29. Главные риски программных проектов и способы реагирования.
30. Управление проектом, направленное на снижение рисков.
31. Мониторинг и контроль рисков.
32. Оценка - вероятностное утверждение.
33. Негативные последствия «агрессивного» расписания.
34. Прагматичный подход.
35. Метод PERT.
36. Обзор метода функциональных точек.
37. Основы методики COCOMO II.
38. Лидерство и управление.

39. Правильные люди.
40. Мотивация.
41. Эффективное взаимодействие.
42. Рабочее планирование.
43. Принципы количественного управления.
44. Завершение проекта
45. Microsoft Project для управления проектами.
46. Составление плана работ.
47. Учет вероятности выполнения работ.
48. Учет стоимости работ.

4.2 Критерии оценок результатов учебной деятельности магистрантов по учебной дисциплине «управление проектами в сфере информационных технологий» (на основании письма Министерства образования Республики Беларусь от 28.05.2013 г. № 09-10/53-ПО)

Десятибалльная шкала в зависимости от величины балла и отметки включает следующие критерии:

10 (десять) баллов, зачтено выставляется за систематизированные, глубокие и полные знания по всем разделам учебной программы учреждения высшего образования по учебной дисциплине, а также по основным вопросам, выходящим за её пределы; за точное использование научной терминологии (в том числе на иностранном языке), грамотное, логически правильное изложение ответа на вопросы; за безупречное владение инструментарием учебной дисциплины, умение его эффективно использовать в постановке и решении научных и профессиональных задач; за выраженную способность самостоятельно и творчески решать сложные проблемы в нестандартной ситуации; за полное и глубокое усвоение основной, дополнительной литературы, по изучаемой учебной дисциплине; за умение свободно ориентироваться в теориях, концепциях и направлениях по изучаемой учебной дисциплине и давать им аналитическую оценку, использовать научные достижения других дисциплин; за творческую самостоятельную работу на практических, лабораторных занятиях, активное творческое участие в групповых обсуждениях, высокий уровень культуры исполнения заданий.

9 (девять) баллов, зачтено выставляется за систематизированные, глубокие и полные знания по всем разделам учебной программы учреждения высшего образования по учебной дисциплине; за точное использование научной терминологии (в том числе на иностранном языке), грамотное, логически правильное изложение ответа на вопросы; за владение инструментарием учебной дисциплины, умение его эффективно использовать в постановке и решении научных и профессиональных задач; за способность

самостоятельно и творчески решать сложные проблемы в нестандартной ситуации в рамках учебной программы учреждения высшего образования по учебной дисциплине; за полное усвоение основной и дополнительной литературы, рекомендованной учебной программой учреждения высшего образования по учебной дисциплине; за умение ориентироваться в теориях, концепциях и направлениях по изучаемой учебной дисциплине и давать им аналитическую оценку; за систематическую, активную самостоятельную работу на практических, лабораторных занятиях, творческое участие в групповых обсуждениях, высокий уровень культуры исполнения заданий.

8 (восемь) баллов, зачтено выставляется за систематизированные, глубокие и полные знания по всем разделам учебной программы учреждения высшего образования по учебной дисциплине в объеме учебной программы учреждения высшего образования по учебной дисциплине; за использование научной терминологии (в том числе на иностранном языке), грамотное, логически правильное изложение ответа на вопросы, умение делать обоснованные выводы и обобщения; за владение инструментарием учебной дисциплины (методами комплексного анализа, техникой информационных технологий), умение его использовать в постановке и решении научных и профессиональных задач; за способность самостоятельно решать сложные проблемы в рамках учебной программы учреждения высшего образования по учебной дисциплине; за усвоение основной и дополнительной литературы, рекомендованной учебной программой учреждения высшего образования по учебной дисциплине; за умение ориентироваться в теориях, концепциях и направлениях по изучаемой учебной дисциплине и давать им аналитическую оценку; за активную самостоятельную работу на практических, лабораторных занятиях, систематическое участие в групповых обсуждениях, высокий уровень культуры исполнения заданий.

7 (семь) баллов, зачтено выставляется за систематизированные, глубокие и полные знания по всем разделам учебной программы учреждения высшего образования по учебной дисциплине; за использование научной терминологии (в том числе на иностранном языке), грамотное, логически правильное изложение ответа на вопросы, умение делать обоснованные выводы и обобщения; за владение инструментарием учебной дисциплины, умение его использовать в постановке и решении научных и профессиональных задач; за свободное владение типовыми решениями в рамках учебной программы учреждения высшего образования по учебной дисциплине; за усвоение основной и дополнительной литературы, рекомендованной учебной программой учреждения высшего образования по учебной дисциплине; за умение ориентироваться в основных теориях, концепциях и направлениях по изучаемой учебной дисциплине и давать им аналитическую оценку; за самостоятельную работу на практических, лабораторных занятиях, участие в групповых обсуждениях, высокий уровень культуры исполнения заданий.

6 (шесть) баллов, зачтено выставляется за достаточно полные и

систематизированные знания в объеме учебной программы учреждения высшего образования по учебной дисциплине; за использование необходимой научной терминологии, грамотное, логически правильное изложение ответа на вопросы, умение делать обобщения и обоснованные выводы; за владение инструментарием учебной дисциплины, умение его использовать в решении учебных и профессиональных задач; за способность самостоятельно применять типовые решения в рамках, учебной программы учреждения высшего образования по учебной дисциплине; за усвоение основной литературы, рекомендованной учебной программой учреждения высшего образования по учебной дисциплине; за умение ориентироваться в базовых теориях, концепциях и направлениях по изучаемой дисциплине и давать им сравнительную оценку; за активную самостоятельную работу на практических, лабораторных занятиях, периодическое участие в групповых обсуждениях, высокий уровень культуры исполнения заданий.

5 (пять) баллов, зачтено выставляется за достаточные знания в объеме учебной программы учреждения высшего образования по учебной дисциплине; за использование научной терминологии, грамотное, логически правильное изложение ответа на вопросы, умение делать выводы; за владение инструментарием учебной дисциплины, умение его использовать в решении учебных и профессиональных задач; за способность самостоятельно применять типовые решения в рамках учебной программы учреждения высшего образования по учебной дисциплине; за усвоение основной литературы, рекомендованной учебной программой учреждения высшего образования по учебной дисциплине; за умение ориентироваться в базовых теориях, концепциях и направлениях по изучаемой учебной дисциплине и давать им сравнительную оценку; за самостоятельную работу на практических, лабораторных занятиях, фрагментарное участие в групповых обсуждениях, достаточный уровень культуры исполнения заданий.

4 (четыре) балла, зачтено выставляется за достаточный объем знаний в рамках образовательного стандарта высшего образования; за усвоение основной литературы, рекомендованной учебной программой учреждения высшего образования по учебной дисциплине; за использование научной терминологии, логическое изложение ответа на вопросы, умение делать выводы без существенных ошибок; за владение инструментарием учебной дисциплины, умение его использовать в решении стандартных (типовых) задач; за умение под руководством преподавателя решать стандартные (типовые) задачи; за умение ориентироваться в основных теориях, концепциях и направлениях по изучаемой учебной дисциплине и давать им оценку; за работу под руководством преподавателя на практических, лабораторных занятиях, допустимый уровень культуры исполнения заданий.

3 (три) балла, не зачтено выставляется за недостаточно полный объем знаний в рамках образовательного стандарта высшего образования; за знание части основной литературы, рекомендованной учебной программой учреждения высшего образования по учебной дисциплине; за использование

научной терминологии, изложение ответа на вопросы с существенными, логическими ошибками; за слабое владение инструментарием учебной дисциплины, некомпетентность в решении стандартных (типовых) задач; за неумение ориентироваться в основных теориях, концепциях и направлениях изучаемой учебной дисциплины; за пассивность на практических и лабораторных занятиях, низкий уровень культуры исполнения заданий.

2 (два) балла, не зачтено выставляется за фрагментарные знания в рамках образовательного стандарта высшего образования; за знания отдельных литературных источников, рекомендованных учебной программой учреждения высшего образования по учебной дисциплине; за неумение использовать научную терминологию учебной дисциплины, наличие в ответе грубых, логических ошибок; за пассивность на практических и лабораторных занятиях, низкий уровень культуры исполнения заданий.

1 (один) балл, не зачтено выставляется за отсутствие знаний и (компетенций) в рамках образовательного, стандарта высшего образования, отказ от ответа, неявка на аттестацию без уважительной причины.

Экзаменуемые имеют право пользоваться во время проведения зачета учебной программой курса. Результаты зачета объявляются, как правило, в день его проведения и заносятся в зачетную книжку экзаменуемого и экзаменационную ведомость.

Магистранты, не выполнившие в полном объеме требования по изучению данной дисциплины, не допускаются кафедрой к сдаче экзамена.

4.3 Самостоятельная работа по дисциплине «Управление проектами в сфере информационных технологий»

В ходе освоения магистрантам рекомендуется использовать в качестве информационно-методической поддержки электронные презентации лекционных занятий, лабораторный практикум, включающий примеры решения типовых задач, задания для самостоятельной работы, контрольные вопросы. Наиболее эффективными формами и методами организации самостоятельной работы магистрантам являются: выполнение заданий, написание рефератов.

4.4 Образец тестовых заданий

по учебной дисциплине «Управление проектами в сфере информационных технологий»

::1::Два вида организации человеческой деятельности в управлении проектами:
=операционная и проектная;
~проектная и программная;
~операционная и процессорная;
~процессорная и программная;
~портфельная и программная

::2::В операционной деятельности отсутствует
~известные внешние условия;
~многократно изученные производственные операции;
= необходимость новых возможностей и творчества;
~известные функции исполнителей;
~постоянные функции исполнителей

::3::В каком случае применяется проектная деятельность?
~известны внешние условия;
~многократно изучены производственные операции;
~известны функции исполнителей;
~постоянны функции исполнителей;
=необходимость новых возможностей и творчества

::4::Главное различие операционной деятельности от проектной состоит в
~уникальности;
=повторяемости процесса;
~отсутствии ограничений по времени;
~отсутствие правил;
~отсутствие нормативов

::5::Задача проектной деятельности состоит в
~установлении нормативов;
~ограничении ресурсов;
~обеспечении развития бизнеса;
~поиске новых возможностей;
=достижении конкретной бизнес-цели

::6::Задача операционной деятельности состоит в
~установлении производственных нормативов;
~ограничении доступности ресурсов;
=обеспечении нормального течения бизнеса;
~поиске новых возможностей;
~достижении конкретной бизнес-цели

::7::Проект – это
~средство оперативного развития;
=средство стратегического развития;
~средство тактического развития;
~средство бизнес-развития;
~средство производственного развития

::8::Цель проекта есть
~определение способа достижения прибыли;
~преобразование проекта в организацию;
~разработка нормативов для работы в операционной деятельности;
=описание того, что необходимо достичь;
~обеспечение нормальной работы

::9::Правильная последовательность проекта
=Цель → Стратегия → Портфель → Программа → Проект → Работа;

~Стратегия → Цель → Программа → Портфель → Проект → Работа;
~Цель → Стратегия → Портфель → Программа → Проект;
~Стратегия → Цель → Программа → Проект → Работа;
~Цель → Стратегия → Программа → Проект → Портфель → Работа

::10::Проект способствует достижению следующих целей
~тактических;
=стратегических;
~оперативных;
~аналитических;
~временных
::11::Какие потери можно отнести к материальным?
~ущерб репутации;
~ущерб здоровью;
~невыполнение сроков сдачи объекта;
~потери рабочего времени;
=потери сырья

::12::На основании чего формируются новые риски в процессе реализации проекта?
=отклонения фактических от базовых значений уже реализованных работ проекта;
~интервью со специалистами по предметной области;
~новых требований к проекту;
~интервью с заинтересованными сторонами;
~интервью с членами проектной команды

::13::Используются ли количественные показатели при выборе перечня рисков?
~в зависимости от типа проекта;
=да, используются для более точной оценки рисков;
~нет, не используются;
~в зависимости от типа организации;
~в зависимости от длительности проекта

::14::Используется ли матрица вероятности и влияния на этапе формирования плана мероприятий по управлению рисками?
~в зависимости от типа организации;
~в зависимости от длительности проекта;
~в зависимости от типа проекта;
=да, используется для выбора наиболее влиятельных рисков;
~нет, не используется

::15::Кто ответственный за реализацию антирисковых мероприятий?
~координатор проекта;
~главный инженер проекта;
~руководитель проекта;
~менеджер по рискам;
=любой участник команды, ответственный за риск

::16::Управление риском проекта - это
=системное применение политики, процедур и методов управления к *задачам определения ситуации,* идентификации, анализа, оценки, обработки, мониторинга риска и обмена информацией, для обеспечения снижения потерь и увеличения рентабельности;

~системное применение политики, процедур и методов управления *целями проекта*, анализа, оценки, обработки, мониторинга информацией, для обеспечения снижения потерь и увеличения рентабельности;
~системное применение политики, процедур и методов *управления командой* проекта и обмена информацией, для обеспечения снижения потерь и увеличения рентабельности;
~системное применение политики, процедур и методов управления к задачам определения ситуации, мониторинга риска и обмена информацией, для обеспечения снижения потерь;
~системное применение политики, процедур и методов управления к задачам определения ситуации, мониторинга риска и обмена информацией, для обеспечения снижения потерь;
~системное применение политики, процедур и методов управления к выбору спонсоров

::17::К способам снижения проектного риска относится
~мотивирование;
~планирование;
=диверсификация;
~контроль;
~управление

5 Оценка трудоемкости и сроков разработки ПО

::18::Оценка трудоемкости
=носит вероятностный характер;
~всегда конкретна;
~точно известна;
~зависит от заказчика;
~зависит от спонсора

::19::К принципам «агрессивного» расписания проекта относится
~сложность задач проекта;
=слишком оптимистичные сроки проекта;
~численность персонала проекта;
~наличие офиса;
~сложность кода

::20::Одним из принципов «агрессивного» расписания проекта является
=необоснованные ожидания на применение новых технологий и средств разработки;
~увеличение численности персонала проекта;
~сложность задач проекта;
~мотивирование высокой заработной платой;
~тестирование программы

::21::«Агрессивное» расписание проекта подразумевает
=директивное занижение сроков реализации проекта;
~фиксированную заработную плату;
~конкретное число разработчиков;
~делегирование полномочий;
~высокий уровень ответственности

::22::Согласно методу PERT неопределенность трудозатрат характеризуется
=наиболее вероятной, оптимистической и пессимистической оценками трудозатрат;
~наиболее сложной, среднесложной и легкой постановками задач;

~количеством работников команды проекта: 3, 5 или 8;
~количеством спонсоров проекта: 3б 5 или 6;
~наличием бухгалтера, юриста и менеджера проекта

::23::Согласно метода PERT среднее время выполнения i-й работы равна
=$$E_i=\frac{P_i+4M_i+O_i}{6}$$. ;
~$$E_i=P_i+4M_i+O_i$$;
~$$E_i=\frac{P_i+M_i+O_i}{6}$$;
~$$E_i=\frac{P_i+4M_i+O_i}{2}$$;
~$$E_i=\frac{P_i+4M_i+O_i}{10}$$

::24::Согласно метода PERT среднеквадратическое отклонение времени выполнения i-й работы равна

=$${\sigma _i}=\frac{P_i-O_i}{2}$$;
~$${\sigma _i}=\frac{P_i+O_i}{2}$$;
~$${\sigma _i}=\frac{P_i-O_i}{10}$$;
~$${\sigma _i}=\frac{P_i-M_i}{2}$$;
~$${\sigma _i}=\frac{P_i-M_i}{6}$$

::25::Анализ функциональных точек – это
=метод измерения размера программного продукта с точки зрения пользователей системы;
~метод измерения требуемой численности персонала;
~метод измерения размера денежной суммы на проект;
~метод измерения размера офиса;
~ метод подсчета числа компьютеров для реализации проекта

::26::Какой этап не предусматривается при анализе методом функциональных точек:
~определение типа оценки;
~определение области оценки и границ продукта;
~подсчет функциональных точек, связанных с данными;
~подсчет функциональных точек, связанных с транзакциями;
=определение языка кода

::27::Анализ методом функциональных точек не предусматривает
~подсчет функциональных точек, связанных с данными;
~подсчет функциональных точек, связанных с транзакциями;
~определение значения фактора выравнивания (FAV);
~расчет количества выровненных функциональных точек (AFP);
=расчет используемого программного обеспечения

::28::Метод функциональных точек предусматривает оценки для
=проекта разработки, проекта развития, продукта;
~инициации проекта, завершения проекта, целей проекта;
~спонсоров проекта, целей проекта, решений проекта;
~заказчика проекта, целей проекта, миссии проекта;
~целей проекта, содержания проекта, продукта проекта

::29::Проект разработки предусматривает оценку
=количества функциональности в первом релизе продукта;
~количество функциональности в последнем релизе продукта;

~количество доработок продукта проекта;
~количество разработчиков проекта;
~количество спонсоров проекта

::30::Проект развития предусматривает в функциональных точках оценку
=добавление, изменение и удаление функционала;
~первого релиза, второго релиза, последнего релиза;
~спонсора, работников команды, целей проекта;
~численности разработчиков, наличия ЭВМ, количества тестировщиков;
~написание кода, тестирование и выпуск продукта

::31::Оценка продукта предусматривает
=оценку объема существующего и установленного продукта;
~оценку объема разрабатывающегося продукта;
~оценку объема первого релиза продукта;
~оценку объема финансирования разработки;
~ оценку численности разработчиков и тестировщиков в проекте

::32::Область оценки для проекта разработки включает
=все разрабатываемые функции;
~все задействованные роли в проекте;
~все финансирование проекта;
~все цели проекта;
~все этапы планирования проекта

::33::Область оценки для проекта поддержки включает
=все добавляемые, изменяемые и удаляемые функции;
~все задействованные роли в проекте;
~все финансирование проекта;
~все цели проекта;
~все этапы планирования проекта
::34::Требования в проекте могут быть
=функциональными;
~окончательными;
~простейшими;
~строгими;
~кардинальными

::35::Результаты проекта должны быть
~мнимыми;
~абстрактными;
=измеряемыми;
~философскими;
~выдуманными

::36::Целью проекта по разработке ПО не может быть
~автоматизация ряда бизнес-процессов;
~реализация стратегических планов;
~выполнение контрактов;
~разрешение специфических проблем;
=получение премии Дарвина

::37::В каком случае проект считается успешным?
~частично удовлетворены требования заказчика;
=удовлетворены все требования;
~удовлетворены все требования работников;
~реализация проекта за 3 дня;
~увеличение бюджета проекта;

::38::Реализация проекта НЕ включает этап
~анализ;
~управление;
=сравнение;
~производство;
~тестирование

::39::Спонсор проекта – это
~лицо, которое будет использовать результат проекта
~представитель исполнителя, уполномоченный принимать решение о выделении ресурсов и изменениях в проекте
=лицо или группа лиц, предоставляющие финансовые ресурсы для проекта
~представитель исполнителя, ответственный за реализацию проекта в срок, в пределах бюджета и с заданным качеством
~субподрядчик или поставщик

::40::Руководитель проекта – это
~лицо или организация, которые будут использовать продукт, услугу или результат проекта
~субподрядчик или поставщик
=представитель исполнителя, ответственный за реализацию проекта в срок, в пределах бюджета и с заданным качеством
~лицо или группа лиц, предоставляющая финансовые ресурсы для проекта
~представитель исполнителя, уполномоченный принимать решение о выделении ресурсов и изменениях в проекте

::41::Выберите понятие фазы завершения
~разработка концепции;
~планирование, как делать;
~материализация идей в виде документированного программного продукта;
=подтверждение, что разработан именно тот продукт, который запланирован в концепции проекта;
~материализация идей в виде протестированного программного продукта

::42::Как можно добиться наибольшей скорости выполнения проекта?
~выполнять все работы параллельно;
~выполнять все работы последовательно;
=выполнять все работы параллельно и последовательно по ситуации;
~взять дополнительного работника в команду;
~ничего не делать.

::43::Сетевой график проекта предназначен для
=управления затратами времени на выполнение комплекса работ проекта;

~управления материальными затратами;
~управления конфликтами проектной команды;
~управления рисками проекта;
~управления маркетингом проекта

::44::Задача менеджера на стадии отдачи:
~взять отпуск;
=поддерживать мотивацию участников;
~сообщить вышестоящему руководству;
~сдать полномочия;
~подсчитать расходы

::45::Переход на новые технологии снизил производительность разработки. Оптимальное решение проблемы
~заменить команду;
~мотивировать сотрудников работать больше;
=упростить разработку рефакторингом кода;
~сообщить руководству;
~урезать функционал

::46::При внедрении новых технологий на что нужно обратить внимание в первую очередь?
~на версию базы данных;
=на возможные проблемы совместимости;
~на объём кода приложения;
~на данные из логов;
~на результаты тестирования

::47::В какой ситуации смена правил и процессов в работе команды неуместна?
~после сдачи очередного релиза;
=перед релизом;
~в самом начале разработки;
~в случае глубокого кризиса проекта;
~когда команда потеряла мотивацию

::48::Что НЕ предполагает доктрина командного менеджмента?
~ясность общих ценностей;
~ясность общих целей;
~взаимный контроль;
=взаимную ненависть;
~взаимозаменяемость

::49::Согласно доктрине командного менеджмента, ответственность за результат несёт
=команда;
~руководитель;
~заказчик;
~подрядчик;
~менеджмент

::50::Выберите положительную черту коллективизма:
~стадность;
=ощущение себя элементом органичной системы;

~подавление личности;
~слепое подчинение меньшинства большинству;
~отсутствие инициативы

4.5 Критерии оценки результатов ККР (компьютерное тестирование)

10 – балльная шкала

10 баллов = 100% (правильных ответов)
98% ≤ 9 баллов ≤ 99%
96% ≤ 8 баллов ≤ 97%
93% ≤ 7 баллов ≤ 95%
82% ≤ 6 баллов ≤ 92%
71% ≤ 5 баллов ≤ 81%
60% ≤ **4 балла** ≤ 70%
50% ≤ 3 балла ≤ 59%
40% ≤ 2 балла ≤ 49%
1 балл < 40%
0 баллов тест не закончен

Для получения положительной отметки (**4 балла**) магистранту необходимо ответить правильно на 30 - 34 вопроса из 50 вопросов, случайно выбранных компьютерной программой по учебной дисциплине.

5 ВСПОМОГАТЕЛЬНЫЙ РАЗДЕЛ

5.1 Учебная программа по учебной дисциплине «Управление проектами в сфере информационных технологий»

Учреждение образования
«Гомельский государственный университет имени Франциска Скорины»

УТВЕРЖДАЮ
Проректор по учебной работе
ГГУ имени Ф.Скорины»
_____ И.В. Семченко
(подпись)
_____
(дата утверждения)
Регистрационный № УД-10209-231/уч.

УПРАВЛЕНИЕ ПРОЕКТАМИ В СФЕРЕ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Учебная программа учреждения высшего образования по учебной дисциплине для специальности
1-40 80 04 Информатика и технологии программирования

2019 г.

Учебная программа дисциплины составлена на основе требований образовательного стандарта высшего образования, ОСВО 1-40 80 04 2019. Высшее образование. Вторая ступень (магистратура). Специальность 1-40 80 04 «Информатика и технологии программирования» и учебного плана, регистрационный № I 40-20-01/Д-19, дата утверждения 09.04.2019.

СОСТАВИТЕЛЬ:
Марченко Л. Н. – заведующий кафедрой фундаментальной и прикладной математики ГГУ имени Франциска Скорины, кандидат технических наук, доцент

РЕЦЕНЗЕНТЫ:
Д. С. Кузьменков – заведующий кафедрой вычислительной математики и программирования УО «Гомельский государственный университет имени Франциска Скорины», кандидат физико-математических наук, доцент;
И. П. Шабалина – доцент кафедры «Высшая математика» УО «Белорусский торгово-экономический университет потребительской кооперации», кандидат физико-математических наук, доцент

РЕКОМЕНДОВАНА К УТВЕРЖДЕНИЮ:

кафедрой фундаментальной и прикладной математики УО «Гомельский государственный университет имени Франциска Скорины»,
(протокол № 9 от 29.04.2019);

научно-методическим советом УО «Гомельский государственный университет имени Франциска Скорины»
(протокол № 8 от 17.05.2019)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Разработка программного обеспечения в большинстве случаев рассматривается как коллективный труд специалистов, направленный на удовлетворение потребности пользователей в автоматизации их деятельности. Поэтому разработка программного обеспечения требует специальной организованности, в частности управления. Необходимым элементом разработки программ является решение задач изучения пользователей, с одной стороны, а с другой – обеспечения с ними обратной связи, направляющей производство. Важным моментом здесь является специфика работы под заказ, когда все производство программы от стадии замысла до передачи в эксплуатацию финансируется внешними по отношении к разработчикам, но весьма заинтересованными заказчиками. Следующая важная характеристика разработки программного обеспечения – стремление к переиспользованию ранее созданных программных компонентов. Особое место занимает задача распространения построенного программного продукта. Поэтому изучение дисциплины «Управление проектами в сфере информационных технологий» является необходимым и актуальным аспектом при подготовке специалистов II ступени высшего образования, и соответствует требованиям современности.

Целью изучения дисциплины государственного компонента «Управление проектами в сфере информационных технологий» является изучение магистрантами теоретических основ и приобретение практических навыков управления проектами в сфере информационных технологий, то есть последовательного достижения ожидаемых результатов при выполнении неповторяющегося мероприятия (действия, процесса) по созданию нового уникального продукта или услуги в сфере информационных технологий, определенного качества, имеющего временные и бюджетные ограничения и характеризующегося неповторимостью условий осуществления.

Основные задачи, решаемые при изучении дисциплины:

– определить цели, предметную область и структуры проекта;

– составить организационно-технологическую модель проекта;

– рассчитать календарный план осуществления проекта;

– сформировать основные разделы сводного плана проекта;

– осуществлять контроль и регулирование хода выполнения проекта по его основным параметрам;

– использовать программные средства для решения основных задач управления проектом.

В результате изучения дисциплины магистрант должен

знать:

- содержание понятий управления проектами;
- виды, состав, содержание проекта, функции его участников;
- основы экспертизы и оценки эффективности проекта;
- методы и приемы мониторинга, координации и контроля разработки и реализации социально-экономического проекта;
- терминологический аппарат управления проектами;

уметь:

– применять современные методы и модели планирования и мониторинга в процессе управления проектами;

– оценивать проекты с помощью анализа результатов их вариантов;

– системно, творчески мыслить при разработке проекта по улучшению административных бизнес-процессов;

– применять полученные знания для разработки и реализации проектов;

– читать и понимать научные, аналитические, статистические материалы по проблематике управления проектами, самостоятельно работать с литературой, писать рефераты, научные записки по актуальным вопросам проектирования; -

владеть:

– методами планирования и расчета потребных материальных, финансовых, кадровых и других ресурсов, источниками их получения и эффективного использования;

– навыками использования инструментария проектного управления для достижения поставленных целей и задач проекта;

– навыками использования программных средств в области управления проектами.

Освоение данной учебной дисциплины должно обеспечить формирование следующих компетенций.

УК-2. Уметь применять современные информационные технологии для эффективного решения научных и профессиональных задач.

УПК-1. Обладать способностью управлять группами (командами) сотрудников, проектами и сетями, осуществлять выбор методологии и технологии разработки программного обеспечения с учетом проектных рисков.

Форма получения высшего образования: дневная.

Дисциплина изучается магистрантами 1-го курса специальности 1-40 80 04 «Информатика и технологии программирования» в 1-м семестре дневной формы получения высшего образования (вторая ступень).

Общее количество часов – 90 часов; аудиторное количество часов – 40 часов (3 зачетные единицы), из них лекций – 20 часов (из них управляемая самостоятельная работа студентов 4 часа), практические занятия – 20 часов. Форма отчетности – экзамен (1 семестр).

СОДЕРЖАНИЕ УЧЕБНОГО МАТЕРИАЛА

Тема 1 Основные положения управления проектами

История и основные понятия. Отличия программной инженерии от других отраслей. Эволюция подходов к управлению программными проектами. Модели процесса разработки ПО. Что надо делать для успеха программного проекта.

Тема 2 Управление проектами. Определения и концепции

Проект - основа инноваций. Критерии успешности проекта. Проект и организационная структура компании. Организация проектной команды. Жизненный цикл проекта. Фазы и продукты.

Тема 3 Инициация проекта

Управление приоритетами проектов. Концепция проекта. Цели и результаты проекта. Допущения и ограничения. Ключевые участники и заинтересованные стороны. Ресурсы. Сроки. Риски. Критерии приемки. Обоснование полезности проекта.

Тема 4. Планирование проекта

Уточнение содержания и состава работ. Планирование управления содержанием. Планирование организационной структуры. Планирование управления конфигурациям. Планирование управления качеством Базовое расписание проекта.

Тема 5. Управление рисками проекта

Основные понятия. Планирование управления рисками. Идентификация рисков. Качественный анализ рисков. Количественный анализ рисков. Планирование реагирования на риски. Главные риски программных проектов и способы реагирования. Управление проектом, направленное на снижение рисков. Мониторинг и контроль рисков.

Тема 6 Оценка трудоемкости и сроков разработки ПО

Оценка - вероятностное утверждение. Негативные последствия «агрессивного» расписания. Прагматичный подход. Метод PERT. Обзор метода функциональных точек. Основы методики COCOMO II.

Тема 7 Формирование команды

Лидерство и управление. Правильные люди. Мотивация. Эффективное взаимодействие.

Тема8 Реализация проекта.

Рабочее планирование. Принципы количественного управления. Завершение проекта

Тема 9 Программное обеспечение в области управления проектами

Microsoft Project. Составление плана работ. Учет вероятности выполнения работ. Учет стоимости работ.

УЧЕБНО-МЕТОДИЧЕСКАЯ КАРТА
УЧЕБНОЙ ДИСЦИПЛИНЫ
очная форма получения высшего образования

Номер раздела, темы	Название раздела, темы	Количество аудиторных часов					Количество часов УСР	Форма контроля знаний
		Лекции	Практические занятия	Семинарские занятия	Лабораторные занятия	Иное		
1	2	3	4	5	6	7	8	9
1	**Основные положения управления проектами** 1 Отличия программной инженерии от других отраслей. 2 Эволюция подходов к управлению программными проектами. 3 Модели процесса разработки ПО.	2	–					Устный опрос
2	**Управление проектами.** Определения и концепции Проект - основа инноваций. Критерии успешности проекта. Проект и организационная структура компании. Организация проектной команды. Жизненный цикл проекта. Фазы и продукты.	2	2					Устный опрос

3	**Инициация проекта** 1 Управление приоритетами проектов. 2 Концепция проекта. 3 Цели и результаты проекта. 4 Допущения и ограничения. 5 Ключевые участники и заинтересованные стороны. 6 Ресурсы. Сроки. Риски. 7 Критерии приемки. 8 Обоснование полезности проекта.	2	2					Устный опрос
4	**Планирование проекта** 1 Уточнение содержания и состава работ. 2 Планирование управления содержанием. 3 Планирование организационной структуры. 4 Планирование управления конфигурациям. 5 Планирование управления качеством 6 Базовое расписание проекта.	–	2			2		Устный опрос Групповая консультация
5	**Управление рисками проекта** 1 Основные понятия. 2 Планирование управления рисками. 3 Идентификация рисков. 4 Качественный анализ рисков. 5 Количественный анализ рисков. 6 Планирование реагирования на риски. 7 Главные риски программных проектов и способы реагирования. 8 Управление проектом, направленное на снижение рисков. 9 Мониторинг и контроль рисков.	2	2			2		Устный опрос Групповая консультация
6	**Оценка трудоемкости и сроков разработки ПО** 1 Оценка - вероятностное утверждение. 2 Негативные последствия «агрессивного» расписания. 3 Прагматичный подход. 4 Метод PERT. 5 Обзор метода функциональных точек. 6 Основы методики COCOMO II.	2	2					Устный опрос

7	**Формирование команды** 1 Лидерство и управление. 2 Правильные люди. Мотивация. 3 Эффективное взаимодействие.	2	2					Устный опрос
8	**Реализация проекта.** Рабочее планирование. Принципы количественного управления. Завершение проекта.	2	2					Устный опрос
9	**Программное обеспечение в области управления проектами** Microsoft Project. Составление плана работ. Учет вероятности выполнения работ. Учет стоимости работ.	2	4					Устный опрос
	Итого	**16**	**20**			4		Экзамен

ИНФОРМАЦИОННО-МЕТОДИЧЕСКАЯ ЧАСТЬ

Перечень тем практических занятий

9. Управление проектами.
10. Инициация проекта.
11. Планирование проекта.
12. Управление рисками проекта.
13. Оценка трудоемкости и сроков разработки ПО.
14. Формирование команды.
15. Реализация проекта.
16. Программное обеспечение в области управления проектами.

Средства диагностики результатов учебной деятельности

Оценка уровня знаний магистранта производится по десятибалльной шкале в соответствии с критериями, утвержденными Министерством образования Республики Беларусь.

Для оценки достижений магистранта рекомендуется использовать следующий диагностический инструментарий:

- устный опрос во время занятий;
- оценивание на основе деловой игры, метода проектов;
- тесты по отдельным разделам дисциплины и дисциплине в целом;
- контрольные опросы;
- рефераты.

Методические рекомендации по организации и выполнению управляемой самостоятельной работы магистрантов

При изучении дисциплины рекомендуется использовать следующие формы самостоятельной работы:

- самостоятельная работа в виде решения задач в аудитории во время проведения практических занятий под контролем преподавателя в соответствии с расписанием,
- обучение с помощью учебной литературы.

Примерная тематика рефератов

1. Современное содержание процессов принятия управленческих решений, характеристики и особенности.
2. Интеграция теории управления и теории принятия решений.

3. Универсальная теория принятия решений в свете трансформаций современных концепций менеджмента.

4. Решающая роль креативности и инноваций в создании конкурентных преимуществ; управление креативностью для нахождения нестандартных, творческих, уникальных решений для развития бизнеса.

5. Современные методы и методологии управления инновационным мышлением.

6. Снижение уровня сложности процесса принятия решения: необходимость, основные формы и проблемы.

7. Формулировка норм поведения как проблема принятия решений: эксплицитные и имплицитные нормы поведения; стандартизация и программирование.

8. Характеристики и особенности объектных, организационных, коммуникационных решений.

9. Проблема интеграции объектных и организационных решений; организационных и коммуникационных решений.

10. Возможности и ограничения моделей принятия решений.

11. Описание неопределенности в теории принятия решений. Необходимость снижения уровня сложности процесса принятия решения.

12. Формирование эксплицитных норм поведения; основные положения теории команд.

13. Методы снижения уровня сложности процессов принятия решений:

14. Творческое мышление, методы и техники его развития. Креативные возможности и барьеры.

15. Методы поиска идей для решения проблем в бизнесе.

16. Сущность инноваций в теории принятия решений; инновационные технологии принятия творческих, нестандартных (уникальных) решений.

17. Выбор альтернатив при решении неструктурированных проблем.

18. Риск: исход принятия решения; платежная матрица; функция ценности; ожидаемая полезность.

19. Описание неопределенности в теории принятия решений.

20. Сущность и содержание управленческих решений. Типология и классификация управленческих решений.

21. Функции решений в методологии и организации процессов управления. Формы подготовки и реализации управленческих решений.

22. Научные подходы в организации разработки и реализации управленческих решений. Принципы формирования управленческих решений, разработанные в научных работах Берга А.И., Богданова А.А, Гвишиани Д.М., Райфа Х., Райфа Г., Цыгичко, Саймона Г. и др.

23. Интеграция теории управления и теории принятия решений. Универсальная теория принятия решений в свете трансформаций современных концепций менеджмента.

24. Решающая роль креативности и инноваций в создании конкурентных преимуществ; управление креативностью для нахождения нестандартных, творческих, уникальных решений для развития бизнеса.

25. Современные методы и методология управления инновационным мышлением в принятии решения.

26. Модели принятия решений.

27. Основная модель принятия решений. Детерминанты (факторы решения) первичные и вторичные.

28. Алгоритм принятия управленческих решений. Организация процесса разработки управленческих решений.

29. Диагностика и идентификация проблем (построение дерева проблем).

30. Понятие альтернативы в процессах принятия решении; выбор альтернатив; языки описания альтернатив. Критерии и ограничения выбора альтернатив.

31. Состав процедур разработки, согласования, утверждения и организации выполнения управленческих решений.

32. Влияние личности менеджера на принятие решений, побуждение сотрудников к участию в принятии управленческих решений.

33. Организация ситуационных центров.

34. Характеристики, особенности и взаимосвязь объектных, организационных, коммуникационных решений. Проблема интеграции объектных и организационных решений; организационных и коммуникационных решений.

35. Методы и приемы анализа альтернатив действий.

36. Формулировка норм поведения как проблема принятия решений: эксплицитные и имплицитные нормы поведения; стандартизация и программирование.

37. Творческое мышление, методы и техники его развития. Креативные возможности и барьеры творческого мышления.

38. Методы поиска идей для решения проблем в бизнесе. Сущность инноваций в теории принятия решений; инновационные технологии принятия творческих, нестандартных (уникальных) решений.

39. Техники креативного мышления в принятии управленческих решений.

40. Выбор альтернатив при решении неструктурированных проблем.

41. Классификация методов реализации управленческих решений. Методы организации выполнения управленческих решений.

42. Методы контроля выполнения решений. Организация мониторинга за процессом выполнения управленческих решений.

43. Ответственность в системе разработки и реализации управленческих решений. Виды ответственности.

44. Эффективность управленческих решений и её составляющие. Методы расчета экономической эффективности подготовки и реализации управленческих решений.

45. Системы информационной и интеллектуальной поддержки разработки и реализации управленческих решений.

Методические рекомендации по организации и выполнению УСР по дисциплине государственного компонента «Управление проектами в сфере информационных технологий»

Для самостоятельного изучения выделяются следующие темы.

1. Планирование проекта.
2. Управление рисками проекта.

Самостоятельное изучение данных тем преследует следующие цели:

– овладеть основными понятиями, определениями,

– уметь проводить необходимые расчеты,

– самостоятельно анализировать полученные результаты, делать соответствующие выводы.

Учебная программа УСР

1 Тема «Планирование проекта» – 2 часа.

Цели: 1) овладеть основными понятиями и определениями по теме;

2) сформировать компетенцию в умении проводить маркетинговые исследования на ИТ-рынке.

Виды заданий УСР с учетом модулей сложности
по теме «Управление проектами в сфере информационных технологий»

А) *Задания, формирующие знания по учебному материалу на уровне узнавания:*

1 Составление глоссария основных определений.
2 Подготовка краткого конспекта по теме.
3 Структурирование материала в виде таблиц или схем.
4 Подготовка презентации по теме «Основы маркетинга в ИТ-сфере».

Форма выполнения заданий – индивидуальная.

Форма контроля выполнения заданий – устное сообщение и обсуждение (1, 2, 3 задания), мультимедийная презентация (4 задание).

Б) *Задания, формирующие компетенции на уровне воспроизведения*:

1 Формулировка основных понятий и определений.
2 Знание основных этапов планирования.
3 Знание основных инструментов планирования.

Форма выполнения заданий – индивидуальная и групповая.

Форма контроля выполнения заданий – устный опрос, групповая консультация.

В) *Задания, формирующие компетенции на уровне применение полученных знаний:*

1 Составление плана по определенному ИТ-продукту или ИТ-услуге.

Форма выполнения заданий – индивидуальная.

Форма контроля выполнения заданий – защита отчета проведенного исследования.

2 Тема «Управление рисками проекта» – 2 часа.

Цели: 1) овладеть основными понятиями и определениями по теме;

2) сформировать компетенцию в умении осуществлять расчеты.

Виды заданий УСР с учетом модулей сложности
по теме «Управление рисками проекта»

А) *Задания, формирующие знания по учебному материалу на уровне узнавания:*

1 Составление глоссария основных определений.

2 Подготовка краткого конспекта по теме.

3 Структурирование материала в виде таблиц или схем.

4 Подготовка презентации по теме «Управление рисками проекта».

Форма выполнения заданий – индивидуальная.

Форма контроля выполнения заданий – устное сообщение и обсуждение (1, 2, 3 задания), мультимедийная презентация (4 задание).

Б) *Задания, формирующие компетенции на уровне воспроизведения*:

1 Формулировка основных понятий по теме.

2 Знание основных рисков программных продуктов.

3 Знание методов управления проектом, направленных на снижение рисков проекта.

Форма выполнения заданий – индивидуальная и групповая.

Форма контроля выполнения заданий – устный опрос, групповая консультация.

В) *Задания, формирующие компетенции на уровне применение полученных знаний:*

1 Описание рисков и их мониторинг для конкретного программного продукта или услуги.

Форма выполнения заданий – индивидуальная.

Форма контроля выполнения заданий – устный опрос.

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

Основная

1. Алиев, В. С.Бизнес-планирование с использованием программы Project Expert (полный курс) : учебное пособие / В. С.Алиев, Д.В.Чистов. – М. : ИНФРА-М, 2014. – 352 с.
2. Беляцкий, Н. П. Менеджмент. Основы лидерства. : уч.пос. / Н.П.Беляцкий. – Мн: Новое знание, 2002. – 250 с.
3. Брукс, Ф.П. Как проектируются и создаются программные комплексы : мифич.человеко-месяц : очерки по систем.программир. / Ф.П. Брукс. – М.: Наука, 1979. – 304 с.
4. Гейзлер, П.С. Управление проектами : учеб.пособие / П.С. Гейзлер. – Мн: Книжный Дом «Мисанта, 2005. – 288 с.
5. Инвестиционный менеджмент : практикум : учеб.пособие. / В.И. Маколов [и др.] – М. : КНОРУС, 2012. — 176 с.
6. Петухова, С.В. Бизнес-планирование: как обосновать и реализовать бизнес-проект : практическое руководство / С.В.Петухова. – М.: Омега-Л, 2014.– 352 с.
7. Смольский, А.П. Деловой менеджмент : учебно-практ.пос. / А.П. Смольский. – М.: Современная школа, 2011. – 230 с.
8. Трофимова, Л. А. Методы принятия управленческих решений : учебник / Л.А. Трофимова. – М.: ЮРАЙТ, 2013. – 335.
9. Фатхутдинов, Р. А. Инновационный менеджмент : учебник для вузов / Р.А.Фатхутдинов. – М.: ЗАО Бизнес-школа Интел-Синтез , 1998. – 272 с.
10. Черняк, В. З. Управление инвестиционными проектами : учеб. / В.З.Черняк. – М.: Юнити-ДАНА, 2004. – 365 с.

Дополнительная

1. Архипенков, С. Лекции по управлению программными проектами: учебное пособие / С. Архипенков. – Москва, 2009. – 128 с. https://ita.sibsutis.ru/sites/csc.sibsutis.ru/files/courses/trpo/sw_project_management.pdf
2. Вольфсон Б. Гибкие методологии разработки. [Электронный ресурс] // URL: http://agilerussia.ru/methodologies/borisvolfson_ebook/
3. Гибкая методология разработки программного обеспечения : курс лекций [электронный ресурс] Режим доступа: http://www.intuit.ru/studies/courses/583/439/info
4. Ильин В., Руководство качеством проектов. Практический опыт / В. Ильин. – СПб.: Вершина, 2006.
5. Управление проектами: Учебное пособие / М.В. Романова. - М.: ИД ФОРУМ: НИЦ ИнфраМ, 2013 http://znanium.com/bookread.php?book=391146