Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»
Институт информационных технологий БГУИР
Кафедра микропроцессорных систем и сетей

И.В. Кашникова

Управление веб-проектами

Учебно-методическое пособие для слушателей переподготовки специальности «Web-дизайн и компьютерная графика»

Минск БГУИР 21

УДК
ББК

Рецензенты:

Кашникова И.В.
Управление веб-проектами
Учебно-методическое пособие : / И.В.Кашникова. - Минск :
БГУИР, 2021. – 45 с. : ил.
ISBN

Пособие включает теоретический материал, вопросы для обсуждения и практические задания по основным темам курса «Управление веб-проектами».

УДК
ББК

ISBN

2021

ТЕМА 1. ПРОИЗВОДСТВЕННАЯ АРХИТЕКТУРА И КОНКРЕТНЫЕ ИТ-ПРОЕКТЫ

1.1 Основы проектной деятельности

В самом общем виде под проектом понимают деятельность, направленную на создание новых продуктов, услуг, достижение уникальных целей.

Все проекты обладают основными и дополнительными признаками.

Основные признаки проекта:
• Проект направлен на достижение конкретных целей • Он имеет ограниченную деятельность с определенным началом и концом • Проект в определенной степени неповторим и уникален

Тот факт, что проект направлен на достижение цели, оказывает большое влияние на планирование и организацию управления проектом. Прежде всего, необходимо четко формулировать цель, начиная с высшего уровня, а затем опускаясь до наиболее детализированных целей и результатов.

Проект заканчивается тогда, когда достигнуты его основные цели или принято решение о досрочном прекращении проекта. Таким образом, проекты выполняются в течение ограниченного периода времени. У проекта существуют достаточно четко определенные начало и окончание. Значительная часть усилий при управлении проектом направлена именно на обеспечение того, чтобы проект был завершен в намеченное время. Отличие проекта от операционной деятельности заключается в том, что проект является однократно не циклической деятельностью.

Проекты - мероприятия в определенной степени неповторимые, уникальные. Вместе с тем степень уникальности может сильно различаться от одного проекта к другому. Уникальность может быть связана как с конечными целями проекта, так и с технологиями создания продукта, с внешними и внутренними условиями реализации проекта.

Дополнительные признаки проекта:
• Проект предполагает последовательную разработку, уточнение целей и планов • Проект предполагает координированное выполнение взаимосвязанных действий

Свойство последовательной разработки проектов является следствием их уникальности. Чем выше уникальность проекта, тем выше уровень неопределенности в начале проекта. Последовательная разработка проекта означает, что результаты и содержание работ проекта уточняются по мере его реализации.

Подавляющее большинство проектов предполагает привлечение различных исполнителей для выполнения отдельных работ проекта. Именно необходимость организации и координации работы исполнителей для выполнения уникальных работ определяет особые требования к задачам планирования и управления проектами. Теоретически можно представить проект, который полностью выполняется одним человеком от начала (постановка целей) и до завершения (достижение конечного результата). Но это возможно лишь для очень простой деятельности.

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.

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.

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.

Примеры операций: эксплуатация электростанции; регулярный расчет заработной платы с использованием типовой компьютерной программы; работа сборочного конвейера.

Примеры проектов: реконструкция электростанции; разработка и внедрение компьютерной программы для расчета заработной платы; разработка и ввод в действие сборочного конвейера.

1.2 Специфика и особенности интернет-проектов

Термин "ИТ-проект" обычно используется для обозначения деятельности, связанной с использованием или созданием некоторой информационной технологии. Это приводит к тому, что ИТ-проекты охватывают очень разнообразные сферы деятельности: разработку программных приложений, создание информационных систем, *развертывание* ИТ-инфраструктуры и пр.

Интернет - проекты являются одними из наиболее сложных. Они связаны со значительными рисками, как технологическими, так и рисками внедрения (человеческий фактор). Процент незавершенных проектов и проектов, завершенных со значительными срывами по срокам и перерасходом бюджета, является одним из самых высоких. В то же время число и сложность этих проектов постоянно возрастают.

С одной стороны, эти работы соответствуют классическому определению проекта , с другой стороны, они обладают известными отличительными особенностями:

- разделение на уровне идеологии заказчика и исполнителя: заказчиком, как правило, является бизнес, а исполнителем – ИТ-специалисты, и есть трудности в выявлении требований, ожиданий от проекта, в формировании технического задания. Существует также проблема эффективных коммуникаций;
- ответственность за результат проекта имеет "солидарный" характер. То есть здесь нельзя возложить ответственность за успех проекта только на исполнителя, точно так же, как нельзя говорить, что исключительно заказчик виновен в том, что проект не удался. В ИТ-проекте должны создаваться определенные условия для взаимодействия сторон, и стороны, участвующие в нем, несут равную ответственность за результаты проекта;
- зачастую реализация ИТ-проекта предусматривает изменение существующих организационных структур на предприятии;
- обычно в ИТ-проект вовлечено множество подразделений организации;
- существует высокая вероятность конфликтов между руководителем проекта, высшим руководством, руководителями подразделений и персоналом организации;
- многие ИТ-проекты имеют колоссальные бюджеты. В крупных компаниях масштабы проектной деятельности в области информационных технологий (ИТ) измеряются миллионами долларов, причем реализация новых проектов происходит постоянно. Если, например, промышленное предприятие достаточно один раз построить – и оно будет работать, не требуя регулярных инвестиций, то развитие ИТ-инфраструктуры в растущих компаниях требует больших и регулярных вложений. Большие бюджеты, в свою очередь, подразумевают больший уровень ответственности и, соответственно, больший уровень компетенции тех людей, которые этими проектами управляют.

Если говорить о реализации ИТ-проектов, следует обратить внимание на следующие особенности:

- зачастую в компании заказчика одновременно выполняются несколько ИТ-проектов;
- приоритеты выполнения проектов постоянно корректируются;
- по мере реализации проектов выполняется уточнение и корректировка требований и содержания проектов;
- велико влияние человеческого фактора: сроки и качество выполнения проекта в основном зависят от непосредственных исполнителей и коммуникации между ними;
- каждый исполнитель может принимать участие в нескольких проектах;
- налицо трудности планирования творческой деятельности, отсутствуют единые нормативы и стандарты;
- сохраняется повышенный уровень риска, вплоть до непредсказуемости результатов;
- происходит постоянное совершенствование технологии выполнения работ.

Анализ статистики показывает, что примерно 90 процентов ИТ-проектов аналогичны уже выполненным. У руководителя проекта имеется *опыт* реализации таких задач и понимание возможных проблем. В этих случаях иерархическая структура проекта и *работ* (ИСП/ИСР) формируется с применением подхода Top-down (сверху вниз), используется *типовая* структура проектной команды, планы проекта (план управления рисками, *план коммуникаций* и пр.) аналогичны планам предыдущих проектов. Однако 10 процентов проектов – инновационные, реализуемые "с нуля" и требующие творчества, нестандартных решений и управленческой смелости. Принятие решений в таких проектах характеризуется высокими рисками, что требует от руководителя глубоких знаний методики проектного управления и понимания особенностей её применения в сфере информационных технологий.

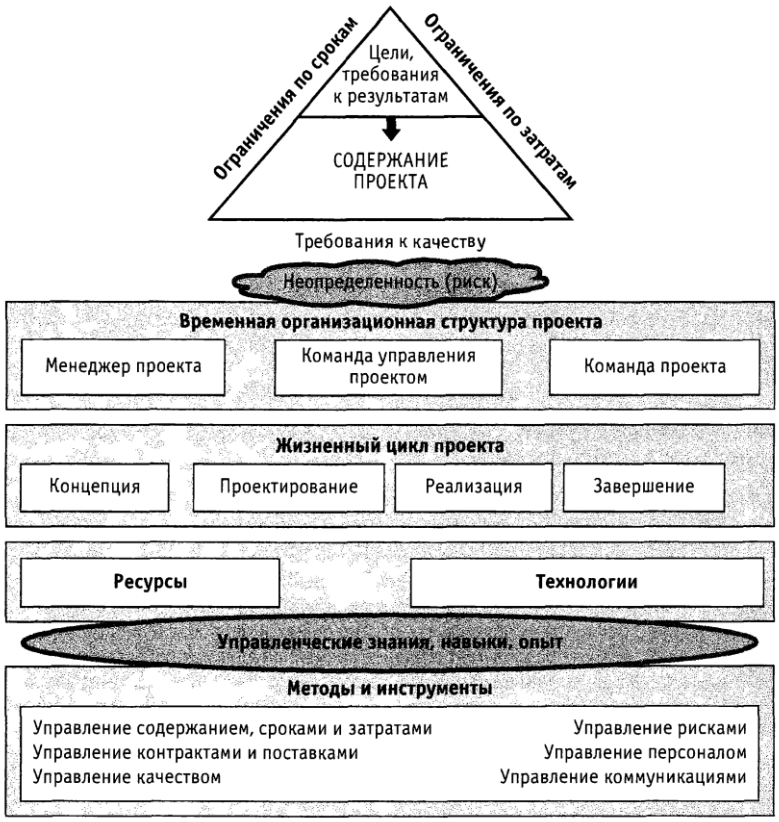
Применение методологии управления проектами позволяет зафиксировать цели и результаты проекта, дать им количественные характеристики, определить временные, стоимостные и качественные параметры проекта, создать реалистичный *план выполнения* проекта, выделить, оценить риски и предотвратить возможные негативные последствия во время реализации проекта.

Для эффективного управления проект должен быть хорошо структурирован. Суть этого процесса сводится к выделению следующих основных элементов:

- фазы жизненного цикла проекта, этапов, работ и отдельных задач;
- организационная структура исполнителей проекта;
- структура распределения ответственности.

1.3 Ключевые концепции проектного менеджмента

Концепция 1. Цели проекта определяют его содержание



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.

Тем самым, цели – это основополагающий мотиватор, который вдохновляет людей на действия. Ясные цели означают, что в выполняемой работе видится смысл, который вызывает высокую степень активности для достижения желаемых результатов. Отсюда вытекают следующие требования к целям: они должны быть такими, чтобы их можно было представить, реализовать, чтобы их достижение было желаемым. Как же на практике можно выделить цель, которая обещает принести успех? Для определения первостепенно важных для проекта целей необходимо четко сформулировать, что ожидается от проекта и исполнительной команды, выяснить заказ

Одной из возможностей быстрого и однозначного формулирования и контроля целей проекта является применение концепции SMART. Критериями концепции SMART являются:

Specific (специфика): то есть точное выражение того, что должно быть достигнуто.

Measurable (измеримость): то есть достижение целей однозначно определяется с помощью графиков, показателей или статистических данных.

Achievable (достижимость): означает, что цели ставятся высокие, но при определенных усилиях могут быть достигнуты. Здесь также имеется в виду, что достижение целей является желательным для всех участников.

Realistic (реалистичность): означает, что достижение целей финансово-технически возможно. Технические и человеческие ресурсы должны присутствовать в достаточном объеме. Особенно следует проверить вопрос имеющегося в распоряжении ноу-хау.

Timely (своевременность): отведенные рамки времени реалистичны. Должна существовать ясность, когда и какие частичные цели и ключевые результаты должны быть достигнуты.

Таким образом, Smart-цели можно подразделить на четыре стандартные группы:

- предметная цель задает, по возможности, точно сформулированное задание, которое должно быть выполнено, и указывает определенные признаки качества, которые при этом должны быть достигнуты.
- цели по срокам
- рамки времени, в течение которого проект должен быть успешно завершен.
- цели по расходам

Концепция 2. Тройственное ограничение

Задача проекта – достижение конкретной бизнес-цели, при соблюдении ограничений «железного треугольника» (Рисунок 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.

Очевидно, что расходы, сроки и качество непосредственно зависят друг от друга. Если экономить на средствах, то снижается качество или увеличиваются сроки. Если сильно поджимают сроки, нужно увеличить количество сотрудников, следовательно, увеличиваются издержки. Будет ли при этом достигнуто нужное качество, это вопрос. Если ставится большое количество производственных целей или предъявляются повышенные требования к качеству, то возрастают расходы на увеличение количества или более высокую квалификацию персонала и/или увеличиваются сроки.

Какой из трех целей: срокам, расходам или предметным целям/качеству отдать предпочтение, зависит от каждого отдельного проекта. На этот вопрос нельзя дать общий ответ. Кроме этого в процессе работы над проектом приоритеты могут меняться. И все же, чаще всего, на первом месте стоят предметные цели, а две другие цели имеют подчиненное значение. При этом очень важно, чтобы каждый участник проекта был хорошо проинформирован о приоритетах, чтобы приспосабливать свои отдельные действия к соответствующим подчиненным целям.

Концепция 3 Учет неопределенности

Управление рисками и неопределенностью – это процессы, связанные с идентификацией, анализом рисков и принятием решений, которые включают максимизацию положительных и минимизацию отрицательных последствий наступления рисковых событий. Процесс управления рисками проекта обычно включает выполнение следующих процедур:

Планирование управления рисками – процесс принятия решений по применению и планированию управления рисками для конкретного проекта. Этот процесс может включать в себя решения по организации, кадровому обеспечению процедур управления рисками проекта, выбор предпочтительной методологии, источников данных для идентификации риска, временной интервал для анализа ситуации. Важно спланировать

управление рисками, адекватное как уровню и типу риска, так и важности проекта для организации.

Идентификация рисков – определение рисков, способных повлиять на проект, и документирование их характеристик. Идентификация рисков не будет эффективной, если она не будет проводиться регулярно на протяжении реализации проекта.

Качественная оценка рисков – процесс представления качественного анализа идентификации рисков и определения рисков, требующих быстрого реагирования. Такая оценка рисков определяет степень важности риска и выбирает способ реагирования. Доступность сопровождающей информации помогает легче расставить приоритеты для разных категорий рисков. Качественная оценка рисков – это оценка условий возникновения рисков и определение их воздействия на проект стандартными методами и средствами. Использование этих средств помогает частично избежать неопределенности, которые часто встречаются в проекте. В течение жизненного цикла проекта должна происходить постоянная переоценка рисков.

Количественная оценка – количественный анализ вероятности возникновения и влияния последствий рисков на проект. Количественная оценка рис-ков часто сопровождает качественную оценку и также требует процесс идентификации рисков. Количественная и количественная оценка рисков могут использоваться по отдельности или вместе, в зависимости от располагаемого времени и бюджета, необходимости в количественной или качественной оценке рисков.

Планирование реагирования на риски – это разработка методов и технологий снижения отрицательного воздействия рисков на проект. Берет на себя ответственность за эффективность защиты проекта от воздействия на него рисков. Планирование включает в себя идентификацию и распределение каждого риска по категориям. Эффективность разработки реагирования прямо определит, будут ли последствия воздействие риска на проект положительными или отрицательными. Стратегия планирования реагирования должна соответствовать типам рисков, рентабельности ресурсов и временным параметрам. Вопросы, обсуждаемые во время встреч, должны быть адекватны задачам на каждой стадии проекта, и согласованы со всеми членами группы по управлению проектом. Обычно требуются несколько вариантов стратегий реагирования на риски.

Мониторинг и контроль рисков – мониторинг рисков, определение остающихся рисков, выполнение плана управления рисками проекта и оценка эффективности действий по минимизации рисков. Все эти процедуры взаимодействуют друг с другом, а также с другими процедурами. Каждая процедура выполняется, по крайней мере, один раз в каждом проекте. Несмотря на то, что процедуры, представленные здесь, рассматриваются как дискретные элементы с четко определенными характеристиками, на практике они могут частично сов- падать и взаимодействовать.

Концепция 4. Концепция жизненного цикла проекта

Жизненный цикл проекта – последовательность фаз проекта, задаваемая исходя из потребностей управления проектом. Жизненный цикл проекта принято делить на фазы, фазы – на стадии, стадии – на этапы. Стадии жизненного цикла проекта могут различаться в зависимости от сферы деятельности и принятой системы организации работ. Однако у каждого проекта можно выделить начальную (прединвестиционную) стадию, стадию реализации проекта и стадию завершения работ по проекту. Это может показаться очевидным, но понятие жизненного цикла проекта является одним из важнейших для менеджера,поскольку именно текущая стадия определяет задачи и виды деятельности менеджера, используемые методики и инструментальные средства.

Традиционным является разбиение проекта на 4 крупных этапа: формулирование проекта, планирование, осуществление и завершение.

Концепция 5. Концепция временной организационной структуры

Для реализации и управления выполнением проекта как правило формируется временная структура. Она должна обеспечить привлечение исполнителей к выполнению проекта, четкое распределение ответственности за достижение целей проекта между конкретными руководителями и специалистами.

Задача менеджера проекта – выстроить команду проекта и организовать ее работу.

Концепция 6 Интеграция технологических и управленческих компетенций

Успех проекта зависит как о т наличия необходимых технологий и квалифицированных исполнителей, тай и от грамотного управления. Можно организовать эффективное управления, но не добиться 4успеза проекта из-за отсутствия квалифицированных исполнителей, использования устаревшего оборудования или ненадежности выбранных технологических решений. И наоборот, имея все ресурсы и технические компетенции, из-за отсутствия грамотного управления можно не реализовать проект.

Тема 2 ЖИЗНЕННЫЙ ЦИКЛ ПРОЕКТА

Базовым стандартом в области жизненного цикла программных средств и систем является международный стандарт ISO/IEC 12207: 1995 [1]. В Республике Беларусь действует аутентичный стандарт СТБ ИСО/МЭК 12207-2003 – Информационная технология – Процессы жизненного цикла программных средств.

В соответствии со стандартом СТБ ИСО/МЭК 12207-2003 под жизненным циклом (ЖЦ) программного средства или системы подразумевается совокупность процессов, работ и задач, включающая в себя разработку, эксплуатацию и сопровождение ПС или системы, охватывающая их жизнь от формулирования концепции до прекращения использования.

В соответствии с данным стандартом жизненный цикл программных средств состоит из процессов. Каждый процесс ЖЦ разделен на набор работ.

Каждая работа разделена на набор задач.

Процессы ЖЦ ПС делятся на следующие группы:

· основные;

· вспомогательные;

· организационные.

Основные процессы жизненного цикла

Основные процессы жизненного цикла состоят из пяти процессов, которые реализуются под управлением основных сторон, вовлеченных в жизненный цикл ПО.

Под основной стороной понимают одну из тех организаций, которые инициируют или выполняют разработку, эксплуатацию или сопровождение программных продуктов. Основными сторонами являются заказчик, поставщик, разработчик, оператор и персонал сопровождения программных продуктов. Основными процессами считают:

1. Процесс заказа (acquisition process). Определяет работы заказчика, то есть организации, которая приобретает ПО.

2. Процесс поставки (supply process). Определяет работы поставщика, то есть организации, которая поставляет ПО заказчику.

3. Процесс разработки (development process). Определяет работы разработчика, то есть организации, которая создает программный продукт.

4. Процесс эксплуатации (operation process). Определяет работы оператора, то есть организации, эксплуатирующей вычислительную систему.

5. Процесс сопровождения (maintenance process). Определяет работы сопровождающей организации, которая предоставляет услуги по сопровождению программного продукта, состоящие в контролируемом изменении программного продукта с целью сохранения его исходного состояния и функциональных возможностей. Данный процесс охватывает перенос ПО в другую операционную среду и снятие ПО с эксплуатации.

Вспомогательные процессы жизненного цикла

Вспомогательные процессы жизненного цикла состоят из восьми процессов.

Вспомогательный процесс считается целенаправленной составной частью другого процесса, обеспечивающей успешную реализацию и качество выполнения программного проекта. Вспомогательный процесс при необходимости инициируется и используется другим процессом. Вспомогательными процессами являются:

1. Процесс документирования (documentation process). Определяет работы по описанию информации, формируемой в процессе жизненного цикла.

2. Процесс управления конфигурацией (configuration management process). Определяет работы по управлению конфигурацией (конфигурация ПО — это совокупность функциональных и физических характеристик, установленных в технической документации и реализованных в ПО). Управление конфигурацией позволяет организовать, систематически учитывать и контролировать внесение изменений в ПО на всех стадиях жизненного цикла.

3. Процесс обеспечения качества (quality assurance process). Определяет работы по объективному обеспечению того, чтобы программный продукт соответствовал установленным требованиям и создавался в рамках утвержденных планов. В качестве методов обеспечения качества могут использоваться совместные оценки, аудиторские проверки, верификация и аттестация.

4. Процесс верификации (verification process). Определяет работы (заказчика, поставщика или независимой стороны) по верификации (проверке реализации конкретных требований) программных продуктов по мере реализации программного проекта.

5. Процесс аттестации (validation process). Определяет работы (заказчика, поставщика или независимой стороны) по аттестации (проверке полноты реализации всех требований) программных продуктов программного проекта.

6. Процесс совместной проверки (joint review process). Определяет работы по оценке состояния и результатов какой-либо работы. Данный процесс может использоваться двумя любыми сторонами, когда одна из сторон (проверяющая) проверяет другую сторону (проверяемую) на совместном совещании.

7. Процесс аудита (audit process). Определяет работы по выявлению соответствия требованиям, планам и договору. Данный процесс может использоваться двумя сторонами, когда одна из сторон (проверяющая) контролирует программные продукты или работы другой стороны (проверяемой). Аудит иначе называют ревизией, проводимой независимым лицом для выявления реального положения дел.

8. Процесс решения проблемы (problem resolution process). Определяет процесс анализа и устранения проблем (включая несоответствия), независимо

от их характера и источника, которые были обнаружены во время осуществления разработки, эксплуатации, сопровождения или других процессов.

Организационные процессы жизненного цикла

Организационные процессы жизненного цикла применяются для объединения взаимосвязанных процессов и персонала, а также для постоянного совершенствования результатов объединения. Существуют четыре разновидности организационных процессов. Организационными процессами являются:

1. Процесс управления (management process). Определяет основные работы по управлению, включая управление проектом при реализации процессов жизненного цикла.
2. Процесс создания инфраструктуры (infrastructure process). Определяет работы по выбору и поддержке средств и действий, обеспечивающих жизненный цикл.
3. Процесс усовершенствования (improvement process). Определяет основные работы, которые организация (заказчика, поставщика, разработчика, оператора, персонала сопровождения или администратора другого процесса) выполняет при создании, оценке, контроле и усовершенствовании выбранных процессов жизненного цикла.
4. Процесс обучения (training process). Определяет работы по соответствующему обучению персонала.

Перечислим основные стандарты:
1.PMBOKR Guide (PMI, 1996 г.) является ISO 9001-совместимым. Кроме того, он имеет международное распространение и поддержку.
2.ISO 10006 (Guidelines to Quality in Project Management) (ISO, 1997 г.).
3.BS 6079 (British Standards Board, 1996 г.).
4.DIN 69 900 series x-50-100 series (German standards DIN 69 900 to 69 903 and 69 905).
5.APM BOK (версия. 3.0) (APM Association for Project Managers: Body of Knowledge, пересм. март 1996 г. (версия 3), High Wycombe, 1996 г.).
6.ICB IPMA Competence Baseline (IPMA, 1999 г.).
7.Australian National Competency Standards for Project Management (AIPM (Sponsor), 1996 г.).
8.Prince 2 (PRojects IN Controlled Environments).
9.ANSI/EIA-748-98 Earned Value Management Systems (EVMS), Июль 1998 г.
10.DSDM (Dynamic Systems Development Method).

Название стандарта	Краткое описание
A Guide to the Project Management Body of Knowledge (PMBOK Guide) 1996 Edition	Свод знаний по управлению проектами (PMBOK 1996) стандарт, разработанный Project Management Institute (PMI) версия 1996 г. Является единственным стандартом в области Project Management, который соответствует ISO 9001. В стандарте описаны разные жизненные циклы проекта и организационные структуры исполняющей организации, определены группы процессов (инициирования, планирования, исполнения, контроля, завершения) и их взаимодействие между собой, выделены основные и поддерживающие процессы, определены девять областей знаний (управление интеграцией, замыслом, временем, стоимостью, качеством, человеческими ресурсами, коммуникациями, рисками, контрактами и поставками). Стандарт базируется на процессном подходе. Для каждой области знаний определены входы, выходы и процедуры преобразования (tools and techniques) входных данных в выходные. Полностью определены взаимодействия между всеми процессами, которые включены в области знаний управления проектами.
A Guide to the Project Management Body of Knowledge (PMBOK Guide) 2000 Edition	Свод знаний по управлению проектами (PMBOK 2000) стандарт, разработанный Project Management Institute (PMI) версии 2000 г. Принят в качестве национального стандарта ANSI (ANSI/PMI 99-001-2000) 27 марта 2001 г.). По содержанию и структуре практически полностью соответствует PMBOK 1996. Полностью переработана глава по Управлению рисками в проекте. Расширены и добавлены разделы, относящиеся к управлению проектами на основе Менеджмента освоенного объема (EVM Earned Value Management). Добавлены новые процедуры (tools and techniques) преобразования входных данных в выходные.
Project Management Institute Practice Standard for WorkBreakdown Structures (Exposure Draft Version)	Стандарт на Иерархические последовательности работ проекта.Предварительная черновая версия, 2000 г. Приведено практическое применений WBS (Work Breakdown Structure) для проектов из различных предметных областей. Описано применение для улучшения процессов планирования и контроля проекта. Приведены стандартизированные структуры для проектов по нефтехимическим производствам, управлению окружающей средой, улучшению процессов, фармацевтическим производствам, строительной индустрии, индустрии сервиса, разработке Web-сайтов, телекоммуникациям, очистке, строительству

	по государственным контрактам, внедрению программных комплексов. Тесным образом связан со стандартами PMBOK 1996 и PMBOK 2000.
Revised Exposure Draft from the Risk Management SIG Committee. 2000	Управление рисками проекта. Предварительная черновая версия, 2000 г. Заново переписанная глава PMBOK 1996, вошедшая в PMBOK 2000.
ISO 10006:1997Quality Management Guidelines to Quality in Project Management	Менеджмент качества. Руководство качеством при управлении проектами, 1997 г. Руководство нацелено на обеспечение заданного уровня качества проекта как на уровне процессов, так и на уровне продуктов. В большой мере по содержанию основано на PMBOK 1996, совпадение вплоть до названий областей знаний управления проектами. Первый стандарт ISO серии 9000, в котором применен процессный подход.

Тема 3 МОДЕЛИ ПРОЦЕССА РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

2.1 Модель водопада

Общая концепция подхода была представлена доктором Уинстоном Ройсом ещё в 1970 году. В его основе лежит логическая последовательность шагов, которые должна быть предприняты на протяжении жизненного цикла разработки ПО. Каждый этап согласовывает компетентными сотрудниками, документируется и передаётся дальше.

Модель состоит из 8 блоков, каждый из которых охватывает свою область ответственности.

Основной постулат Waterfall модели разработки ПО заключается в том, что следующий этап не может быть начат, пока не завершен предыдущий. При этом произвольные переходы вперед или назад не допускаются, а этапы не перекрывают друг друга.

Рис. 2.1 Модель водопада

Основной идеей, лежащей в основе фаз, является разделение интересов Таким образом, большая и сложная задача создания программного обеспечения разбивается на более мелкие задачи (которые сами по себе все еще довольно сложны): определение требований, разработка проекта и т. Д. Разделение проблем и сосредоточение внимания на избранных на этапе дает лучшую помощь инженерам и менеджерам в решении проблемы.

Этап анализа требований упоминается как «анализ и планирование».
Линейный порядок работ имеет некоторые важные последствия.

Для явного распознавания завершения очередного этапа и начала следующего, требуются использовать некоторый способ подтверждения в конце каждого этапа.

Это обычно выполняется с помощью некоторой проверки (verification) и утверждения (validation), которые будут гарантировать, что:

- результат выполнения этапа согласуется с его входными данными (что являлось результатом работы (выходом) предыдущего этапа);
- результат выполнения данного этапа согласуется с общими требованиями заказчика к системе.

Следствием потребности в подтверждении является то, что каждый этап должен иметь конкретный результат, который м.б. оценен и утвержден.

Результат каждого последующего этапа часто называется продуктом труда и обычно имеет форму некоторого документа (например, описания требований или описание проекта, программный код).

Хотя набор документов, создаваемых в проекте зависит от его реализации, следующие документы обычно составляют обоснованный набор, который должен быть создан в каждом проекте:

- спецификация требований;
- план проекта;
- документы по проектированию (архитектура системы, описание подсистем, детальный проект);
- план тестирования и отчет о результатах ;
- описание конечной программной реализации;
- руководства по работе с ПО (администратора, пользователя).

На практике эти этапы перекрывают друг друга и передают информацию друг другу.
Модель водопада согласуется с другими моделями инженерных процессов, и документация создается на каждом этапе.
Это делает процесс видимым, чтобы менеджеры могли следить за ходом выполнения плана развития.
Его главная проблема - негибкое разбиение проекта на отдельные этапы.
В принципе, модель водопада должна использоваться только тогда, когда требования хорошо поняты и вряд ли радикально изменится во время разработки системы.

2.2 V-образная модель

Концепция V-образной модели была разработана Германией и США в конце 1980-х годов независимо друг от друга:

− Немецкая V-модель была разработана аэрокосмической компанией IABG в Оттобрунне рядом с Мюнхеном в содействии с Федеральным департаментом по закупке вооружений в Кобленце, для Министерства

обороны Германии. Модель была принята немецкой федеральной администрацией для гражданских нужд летом 1992.

− Американская V-Model (VEE) была разработана национальным советом по системной инженерии (международным — с 1995 года) для спутниковых систем, включая оборудование, программное обеспечение и взаимодействие с пользователями.

Модель была предложена именно для того, чтобы устранить недостатки каскадной модели, а название – V-образная, или шарнирная – появилось из-за ее специфического графического представления.

V-модель — продвинутый вариант классической каскадной модели. Он предусматривает глубокий контроль текущего процесса перед переходом на следующий этап. При использовании V-модели тестирование начинается еще со стадии написания требований.

Под тестированием подразумевается ранний контроль качества путём ряда обзоров и проверок — так называемое статическое тестирование. Этот метод позволяет выявить огрехи на самых ранних стадиях развития проекта и минимизировать ошибки в дальнейшем.

Каждый уровень тестирования имеет отдельный тест-план. Во время тестирования текущего уровня мы строим стратегию для работы над каждым его элементом. При создании тест-планов — определяем ожидаемые результаты тестирования, а также указываем критерии входа и выхода для каждого уровня.

В V-модели тестирование идёт строго параллельно последовательным этапам разработки. Схема этой модели представляет собой букву V, из-за которой она, собственно, и получила своё название: левая часть отражает процесс дизайна программного обеспечения, а правая — процесс разработки и прямого тестирования. Обратите внимание, во многих компаниях одни и те же этапы разработки и тестирования могут обозначаться по-разному. Пересечение обеих частей модели находится внизу буквы V, начиная с модульного тестирования на самом глубоком уровне и переходя к интеграционному, системному и приёмочному тестированию на более высоких.

V-образная модель применима к системам, которым особенно важно бесперебойное функционирование. Например, прикладные программы в клиниках для наблюдения за пациентами, интегрированное ПО для механизмов управления аварийными подушками безопасности в транспортных средствах и так далее. Особенностью модели можно считать то, что она направлена на тщательную проверку и тестирование продукта, находящегося уже на первоначальных стадиях проектирования. Стадия тестирования проводится одновременно с соответствующей стадией разработки, например, во время кодирования пишутся модульные тесты.

2.3 Модель прототипирования

Цель процесса разработки на основе прототипов заключается исправление первого ограничения модели водопадов.

Основная идея – вместо замораживания требований, создавать, перед началом проектирования и кодирования, одноразовый прототип, который может помочь понять требования заказчика.

Прототип сильно – это упрощенный вариант разрабатываемого ПО.

Он разрабатывается на основе текущих известных требований.

Очевидно, что разработка прототипа тоже проходит этапы проектирования, кодирования и тестирования, но каждый их этих этапов, но каждый из этих этапов выполняется не формально и не тщательно.

Прототипирование является привлекательной идеей для сложных и больших систем, для которых нет ручного процесса или существующей системы, которые могут помочь выявить требования.

В таких ситуациях, предоставление заказчику возможности «поиграть» с прототипом позволяет выявить неоценимые и неосязаемые результаты, которые могут помочь определить требования к системе.

Ситуации, в которых требуется прототипировнаия:

- для ***демонстрации осуществимости некоторого нового подхода***;
- при разработке новых систем, когда не понятно, какие ограничения могут возникнуть, или какие алгоритмы могут быть разработаны для реализации требований.

В обоих ситуациях риск, связанный с данными проектами м.б. уменьшен, за счет выполнения прототипирования.

Последовательность действий при макетировании представлена на рис.

Процесс разработки с использованием прототипа обычно выполняется следующим образом:

1. Разработка прототипа обычно начинается, когда разработана предварительная версия документа с спецификациями требований.
2. После разработки прототипа, конечные пользователи и заказчику могут использовать и исследовать прототип.

– Описываются и передаются разработчикам отзывы: что было сделано правильно; чего не хватает; что требуется добавить и т.п.

3. На основе отзывов выполняется доработка прототипа, а затем пользователям и заказчикам опять предоставляется возможность использовать данную ПС.
4. Такой цикл повторяется до тех пор, пока выгода от дальнейшего изменения прототипа превышает стоимость его доработки.
5. На основе такой обратной связи (отзывов), начальные требования модифицируются, чтобы получить окончательную спецификацию требований, которая затем используется разработки ПС производственного уровня.
6. Стоимость разработки прототипа должна быть очень низкой.
7. Для этого в прототип включаются только те возможности, которые будут полезными для получения отзыва от пользователей.

Обычно прототипирование хорошо подходит для проектов, в которых трудно выявить требования и доверие к заявленным требованиям низкая.

В тех проектах, у которых требования не совсем понятны в начале работы, использование модели процесса «прототипирование» может быть наиболее привлекательным методом разработки ПО.

Данный метод также является хорошим способом уменьшения некоторых рисков, связанных с проектом.

2.4 Итеративная модель процесса разработки ПО

Итеративная модель процесса разработки ПО нацелена преодоление 3 и 4 недостатков модели водопадов.

Данная МП пытается объединить достоинства модели прототипирования и модели водопада.

Основная идея данного подхода заключается в том, что ПО должно разрабатываться пошагово (постепенно, итеративно, инкрементально), на каждом шаге к создаваемой ПС должны добавляться некоторые функциональные возможности, до тех пор, пока система не будет реализована полностью.

Итеративная модель улучшения является примером данного подхода.

Модель предполагает разбиение жизненного цикла проекта на последовательность итераций, каждая из которых напоминает «мини-проект», включая все процессы разработки в применении к созданию меньших фрагментов функциональности, по сравнению с проектом в целом.

Цель каждой *итерации* — получение работающей версии программной системы, включающей функциональность, определённую интегрированным содержанием всех предыдущих и текущей итерации.

Результат финальной итерации содержит всю требуемую функциональность продукта.

Таким образом, с завершением каждой итерации продукт получает приращение — *инкремент* — к его возможностям, которые, следовательно, развиваются *эволюционно*.

Итеративность, инкрементальность и эволюционность в данном случае есть выражение одного и то же смысла разными словами со слегка разных точек зрения.

Различные варианты итерационного подхода реализованы в большинстве современных методологий разработки (RUP, MSF, XP).

2.5 Спиральная модель

Спиральная модель (англ. spiral model) была разработана в середине 1980-х годов Барри Боэмом.

Она основана на классическом цикле Деминга PDCA (plan-do-check-act).

При использовании этой модели ПО создается в несколько итераций (витков спирали) методом прототипирования.

Каждая итерация соответствует созданию фрагмента или версии ПО, на ней уточняются цели и характеристики проекта, оценивается качество полученных результатов и планируются работы следующей итерации.

На каждой итерации оцениваются:

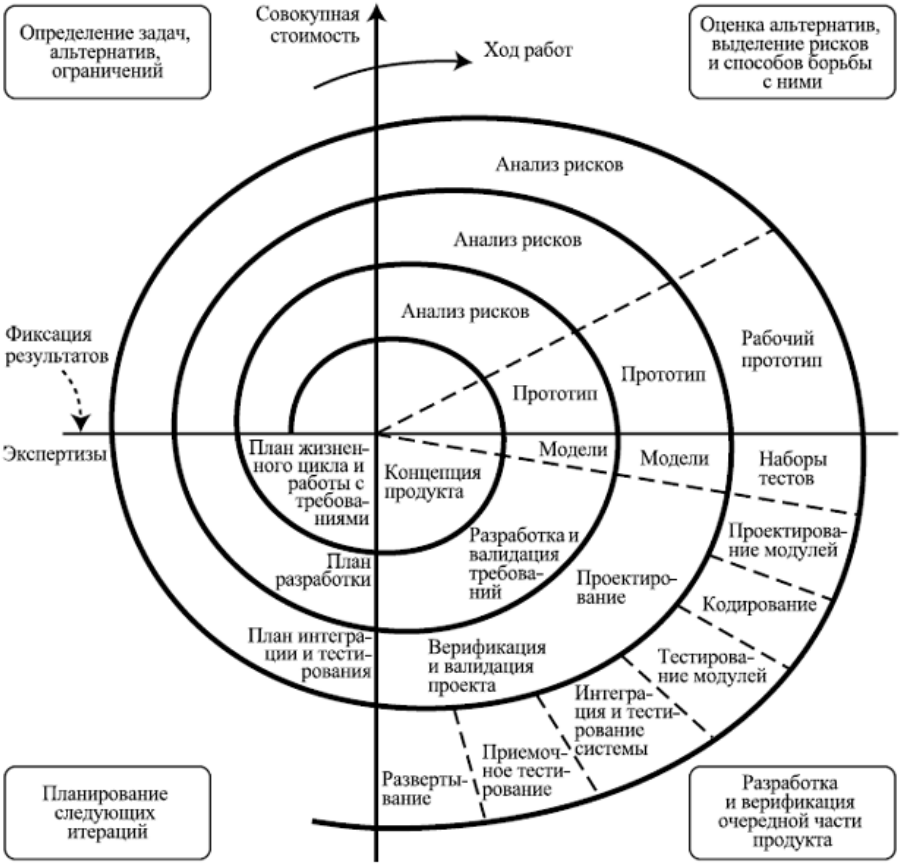
- риск превышения сроков и стоимости проекта;
- необходимость выполнения ещё одной итерации;

- степень полноты и точности понимания требований к системе;
- целесообразность прекращения проекта.

Важно понимать, что спиральная модель является не альтернативой эволюционной модели (модели IID), а специально проработанным вариантом.

К сожалению, нередко спиральную модель либо ошибочно используют как синоним эволюционной модели вообще, либо (не менее ошибочно) упоминают как совершенно самостоятельную модель наряду с IID.

Отличительной особенностью спиральной модели является специальное внимание, уделяемое рискам, влияющим на организацию жизненного цикла, и контрольным точкам.



Боэм формулирует 10 наиболее распространённых (по приоритетам) рисков:

1. Дефицит специалистов.
2. Нереалистичные сроки и бюджет.
3. Реализация несоответствующей функциональности.
4. Разработка неправильного пользовательского интерфейса.
5. 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.

Большая часть этих рисков связана с организационными и процессными аспектами взаимодействия специалистов в проектной команде.

Каждый виток спирали соответствует созданию фрагмента или версии программного обеспечения, на нем уточняются цели и характеристики проекта, определяется его качество и планируются работы следующего витка спирали. Таким образом углубляются и последовательно конкретизируются детали проекта и в результате выбирается обоснованный вариант, который доводится до реализации. Каждый виток разбит на 4 сектора:

- оценка и разрешение рисков,
- определение целей,
- разработка и тестирование,
- планирование.

На каждом витке спирали могут применяться разные модели процесса разработки ПО. В конечном итоге на выходе получается готовый продукт. Модель сочетает в себе возможности модели прототипирования и водопадной модели. Разработка итерациями отражает объективно существующий спиральный цикл создания системы. Неполное завершение работ на каждом этапе позволяет переходить на следующий этап, не дожидаясь полного завершения работы на текущем. При итеративном способе разработки недостающую работу можно будет выполнить на следующей итерации. Главная задача — как можно быстрее показать пользователям системы работоспособный продукт, тем самым активизируя процесс уточнения и дополнения требований. Основная проблема спирального цикла — определение момента перехода на следующий этап. Для ее решения необходимо ввести временные ограничения на каждый из этапов жизненного цикла. Переход осуществляется в соответствии с планом, даже если не вся запланированная работа закончена. План составляется на основе статистических данных, полученных в предыдущих проектах, и личного опыта разработчиков.

Спиральная модель ориентирована на большие, дорогостоящие и сложные проекты. В условиях, когда бизнес цели таких проектов могут измениться, но требуется разработка стабильной архитектуры, удовлетворяющей высоким требованиям по нагрузке и устойчивости,

Тема 4. ГИБКИЕ ПРОЦЕССЫ РАЗРАБОТКИ ПО

Гибкая методология разработки (англ. Agile software development, agile-методы) — набор подходов к разработке программного обеспечения, ориентированных на использование итеративной разработки и динамическое формирование требований и обеспечение их реализации в результате постоянного взаимодействия внутри самоорганизующихся рабочих групп, состоящих из специалистов различного профиля.

Большинство гибких методологий нацелены на минимизацию рисков путём сведения разработки к серии коротких циклов, называемых итерациями, которые обычно длятся две-три недели. Каждая итерация сама по себе выглядит как программный проект в миниатюре и включает все задачи, необходимые для выдачи мини-прироста по функциональности: планирование, анализ требований, проектирование, кодирование, тестирование и документирование. Хотя отдельная итерация, как правило, недостаточна для выпуска новой версии продукта, подразумевается, что гибкий программный проект готов к выпуску в конце каждой итерации. По окончании каждой итерации команда выполняет переоценку приоритетов разработки.

Agile-методы делают упор на непосредственное общение лицом к лицу. Большинство agile-команд расположены в одном офисе, иногда называемом англ. bullpen. Как минимум, она включает и «заказчиков» (англ. product owner — заказчик или его полномочный представитель, определяющий требования к продукту; эту роль может выполнять менеджер проекта, бизнес-аналитик или клиент). Офис может также включать тестировщиков, дизайнеров интерфейса, технических писателей и менеджеров.

Основной метрикой agile-методов является рабочий продукт. Отдавая предпочтение непосредственному общению, agile-методы уменьшают объём письменной документации по сравнению с другими методами. Это привело к критике этих методов как недисциплинированных.

Суть agile-подхода изложена в “манифесте”, но для заказчика ее можно коротко сформулировать так:

- разработка ведется короткими циклами (итерациями), продолжительностью 1-4 недели;
- в конце каждой итерации заказчик получает ценное для него приложение (или его часть), которое можно использовать в бизнесе;
- команда разработки сотрудничает с Заказчиком в ходе всего проекта;
- изменения в проекте приветствуются и быстро включаются в работу.

В настоящее время agile-принципы используются в работе десятки тысяч команд по всему миру.

Scrum (Скрам) — это не аббревиатура, этот термин взят из регби, который обозначает схватку вокруг мяча.

Сам термин Scrum можно определить так — это методология управления проектами, которая построена на принципах тайм-менеджмента. Основной ее особенностью является вовлеченность в процесс всех участников, причем у каждого участника есть своя определенная роль. Суть в том, что не только команда работает над решением задачи, но все те, кому интересно решение задачи. Не просто поставили задачу и расслабились, а постоянно «работают» с командой и эта работа не означает только постоянный контроль.

Основные термины, которые используются в методологии:

- **Владелец продукта (Product owner)** — человек, который имеет непосредственный интерес в качественном конечном продукте, он понимает, как это продукт должен выглядеть/работать. Этот человек не работает в

команде, он работает на стороне заказчика/клиента (это может быть как другая компания, так и другой отдел), но этот человек работает с командой. И это тот человек, который расставляет приоритеты для задач.

- **Scrum-мастер** — это человек, которого можно назвать руководителем проекта, хотя это не совсем так. Главное, что это человек «зараженный Scrum-бациллой» настолько, что несет ее как своей команде, так и заказчику и, соответственно, следит за тем, чтобы все принципы Scrum соблюдались.

- **Scrum-команда** — это команда, которая принимает все принципы Scrum и готова с ними работать.

- **Спринт** — отрезок времени, который берется для выполнения определенного (ограниченного) списка задач. Рекомендуется брать 2-4 недели (длительность определяется командой один раз).

- **Бэклог (backlog)** — это список всех работ. Можно сказать, это ежедневник общего пользования. Различают 2 вида бэклогов: Product-бэклог и спринт-бэклог.

  1.Product-бэклог — это полный список всех работ, при реализации которых мы получим конечный продукт.

  2. Спринт-бэклог — это список работ, который определила команда и согласовала с Владельцем продукта на ближайший отчетный период (спринт). Задания в спринт-бэклог берутся из product-бэклога.

- **Планирование спринта** — это совещание, на котором присутствуют все (команда, Scrum-мастер, Владелец продукта). В течение этого совещания Владелец продукта определяет приоритеты заданий, которые он хотел бы увидеть выполненными по истечении спринта. Команда оценивает по времени, сколько из желаемого они могут выполнить. В итоге получается список заданий, который не может меняться в течение спринта и к концу спринта должен быть полностью выполнен.

В начале каждого спринта проводится планирование этого самого спринта. В планировании спринта участвуют заказчики, пользователи, менеджмент, Product Owner, Скрам Мастер и команда.

Планирование спринта состоит из двух последовательных митингов.

1)Планирование спринта, митинг первый.

Участники: команда, Product Owner, Scrum Master, пользователи, менеджмент

Цель: Определить цель спринта (Sprint Goal) и Sprint Backlog - функциональность, которая будет разработана в течение следующего спринта для достижения цели спринта.

Артефакт: Sprint Backlog

2) Планирование спринта, митинг второй.

Участники: Скрам Мастер, команда

Цель: определить, как именно будет разрабатываться определенная функциональность для того, чтобы достичь цели спринта. Для каждого элемента Sprint Backlog определяется список задач и оценивается их продолжительность.

Артефакт: в Sprint Backlog появляются задачи

Если в ходе спринта выясняется, что команда не может успеть сделать запланированное на спринт, то Скрам Мастер, Product Owner и команда встречаются и выясняют, как можно сократить scope работ и при этом достичь цели спринта.

2.Остановка спринта \(Sprint Abnormal Termination\).
Остановка спринта производится в исключительных ситуациях. Спринт может быть остановлен до того, как закончатся отведенные 30 дней. Спринт может остановить команда, если понимает, что не может достичь цели спринта в отведенное время. Спринт может остановить Product Owner, если необходимость в достижении цели спринта исчезла.

После остановки спринта проводится митинг с командой, где обсуждаются причины остановки спринта. После этого начинается новый спринт: производится его планирование и стартуются работы.

3.Daily Scrum Meeting.
Этот митинг проходит каждое утро в начале дня. Он предназначен для того, чтобы все члены команды знали, кто и чем занимается в проекте. Длительность этого митинга строго ограничена и не должна превышать 15 минут. Цель митинга — поделиться информацией. Он не предназначен для решения проблем в проекте. Все требующие специального обсуждения вопросы должны быть вынесены за пределы митинга.

Скрам митинг проводит Скрам Мастер. Он по кругу задает вопросы каждому члену команды:

• Что сделано вчера?

• Что будет сделано сегодня?

• С какими проблемами столкнулся?

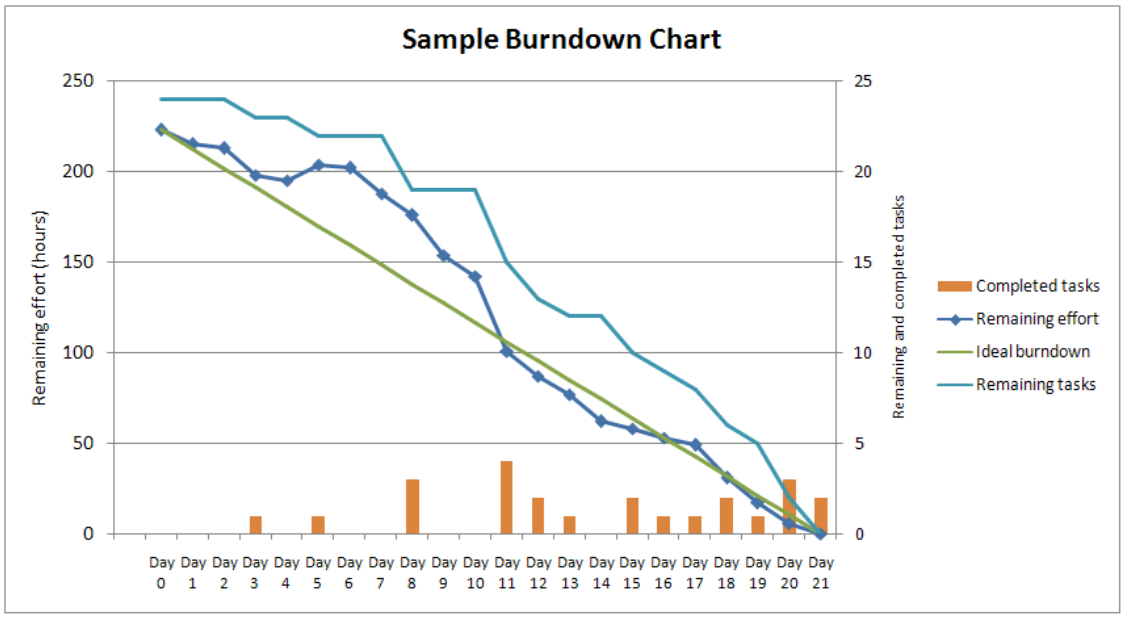
Скрам Мастер собирает все открытые для обсуждения вопросы в виде Action Items, например, в формате что/кто/когда, например:

• Обсудить проблему с отрисовкой контрола.

• Петя и Вася.

• Сразу после скрама.

Диаграмма сгорания задач (Burndown chart)



Диаграмма, которая показывает количество сделанной и оставшейся работы. Обновляется ежедневно с тем, чтобы в простой форме показать подвижки в работе над спринтом. График должен быть общедоступен. Существуют разные виды диаграммы:
• **Диаграмма сгорания работ для спринта** — показывает, сколько уже задач сделано и сколько ещё остаётся сделать в текущем спринте.
• **Диаграмма сгорания работ для выпуска проекта** показывает, сколько уже задач сделано и сколько ещё остаётся сделать до выпуска продукта (обычно строится на базе нескольких спринтов).

Ретроспектива

В конце каждого Спринта Скрам Команда собирается на ретроспективу. **Цель**: пересмотреть качество существующих процессов, взаимоотношения людей и применяемые инструменты. Команда определяет, что прошло хорошо, а что прошло не очень, а также выявляет потенциальные возможности для улучшений. Они создают план улучшений на будущее.

Экстремальное программирование (XP)

Двенадцать основных приёмов экстремального программирования (по первому изданию книги Extreme programming explained) могут быть объединены в четыре группы:

1.Короткий цикл обратной связи (Fine-scale feedback):

• Разработка через тестирование (Test-driven development).

• Игра в планирование (Planning game).

• Заказчик всегда рядом (Whole team, Onsite customer).

• Парное программирование (Pair programming).

2.Непрерывный, а не пакетный процесс:

• Непрерывная интеграция (Continuous integration).

• Рефакторинг (Design improvement, Refactoring).

• Частые небольшие релизы (Small releases).

3.Понимание, разделяемое всеми:

• Простота (Simple design).

• Метафора системы (System metaphor).

• Коллективное владение кодом (Collective code ownership) или выбранными шаблонами проектирования (Collective patterns ownership).

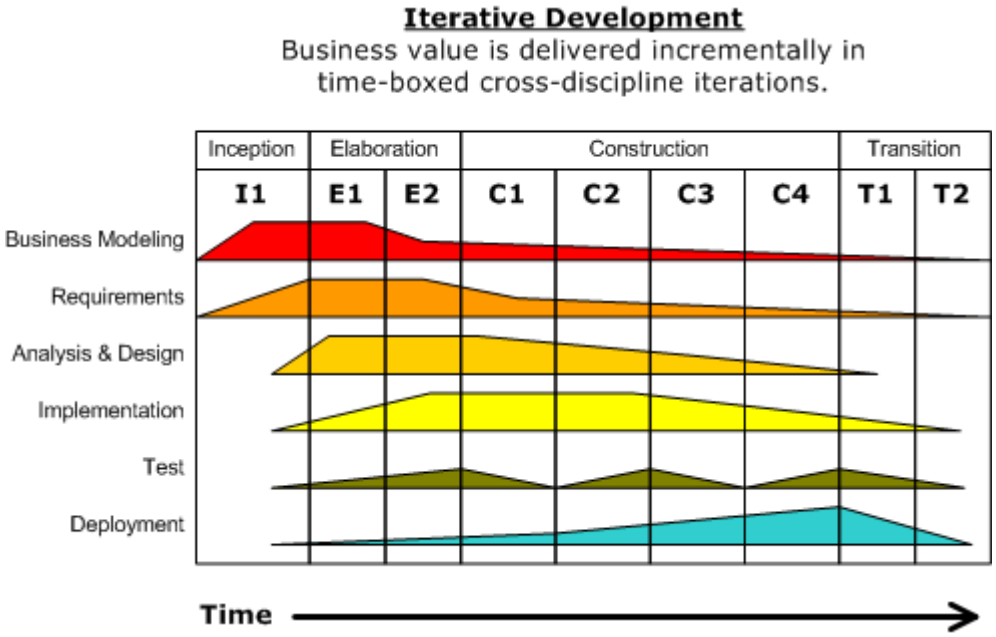
• Стандарт кодирования (Coding standard or Coding conventions).

4.Социальная защищенность программиста (Programmer welfare):

• 40-часовая рабочая неделя (Sustainable pace, Forty-hour week).

RATIONAL UNIFIED PROCESS (RUP)

RATIONAL UNIFIED PROCESS (RUP) — методология разработки программного обеспечения, созданная компанией Rational Software.



В основе методологии лежат 6 основных принципов:

- Компонентная архитектура, реализуемая и тестируемая на ранних стадиях проекта.
- Работа над проектом в сплочённой команде, ключевая роль в которой принадлежит архитекторам.

- Ранняя идентификация и непрерывное устранение возможных рисков.
- Концентрация на выполнении требований заказчиков к исполняемой программе.
- Ожидание изменений в требованиях, проектных решениях и реализации в процессе разработки.
- Постоянное обеспечение качества на всех этапах разработки проекта.

Использование методологии RUP направлено на итеративную модель разработки. Особенность методологии состоит в том, что степень формализации может меняться в зависимости от потребностей проекта. Можно создавать все требуемые документы и достигнуть максимального уровня формализации, по окончании каждого этапа и каждой итерации а можно создавать только необходимые для работы документы, вплоть до полного их отсутствия. За счет такого подхода к формализации процессов методология является достаточно гибкой и широко популярной. Данная методология применима как в небольших и быстрых проектах, где за счет отсутствия формализации требуется сократить время выполнения проекта и расходы, так и в больших и сложных проектах, где требуется высокий уровень формализма, например, с целью дальнейшей сертификации продукта. Это преимущество дает возможность использовать одну и ту же команду разработчиков для реализации различных по объему и требованиям.

Практическое занятие 1 РАЗРАБОТКА ТРЕБОВАНИЙ К ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ

Требованиями (requirements) называют описание функциональных возможностей и ограничений, накладываемых на создаваемую программную систему. Обычно требования выражают, что система должна делать. Здесь не пытаются сформулировать, как добиться выполнения этих функций.

Различают две категории представления требований: требования заказчика (первичные требования) и требования разработчика (детальные требования). Отличаются они друг от друга степенью проработки описаний. Первичные требования документируют желания и потребности заказчика и пишутся на языке, понятном заказчику. Детальные требования документируют требования в специальной, структурированной форме, они детализированы по отношению к первичным требованиям.

Работу по созданию первичных требований будем называть сбором, или формированием требований. Проводится она на этапе подготовки жизненного цикла разработки.

Работу по созданию детальных требований будем называть анализом требований. Проводится она на этапе моделирования жизненного цикла разработки.

Различают два вида требований:

 Функциональные требования. Они описывают поведение системы и сервисы (функции), которые она должна выполнять. При этом исходят из всестороннего анализа проблемной (предметной) области. Рассматриваются разнообразные варианты поведения, определяемые различными данными и состояниями внешней среды.

 Нефункциональные требования. Эти требования относятся к характеристикам системы и ее внешнего окружения. Дополнительно могут перечисляться ограничения, накладываемые на действия и функции системы, а также на условия разработки (ограничения по времени, ограничения на организацию проекта, стандарты и т. д.).

И. Соммервилл предложил выделять три группы нефункциональных требований

 Требования к программной системе. Описывают свойства и характеристики системы. Сюда относятся требования к скорости работы, производительности, емкости необходимой памяти, надежности, переносимости системы на разные компьютерные платформы и удобству эксплуатации.

 Организационные требования. Отображают вопросы работы и организации взаимодействия заказчика и разработчика. Они включают стандарты разработки программной системы, требования к методам и средствам разработки, указывают сроки создания и набор документации.

 Внешние требования. Учитывают факторы внешней среды. Они определяют требования по взаимодействию данной системы с внешним окружением, юридические обязательства, а также этические требования, гарантирующие приемлемость системы для пользователей.

Требования должны обладать следующими характеристиками:

1. Единичность. Это означает, что требование относится только к одному свойству. Например, система должна выполнять такое-то действие. Пример хорошего требования: система должна позволять регистрацию пользователей. Пример плохого требования: система должна позволять авторизацию пользователей по данным социальных сетей и запрашивать из социальных сетей и публиковать на стене пользователя в социальной сети определенную информацию. Почему это требование некорректное - не указаны социальные сети, а далеко не все разрешают сторонним приложения размещать информацию на стене пользователя, а, во-вторых, тут объединены два требования.

2. Атомарность. Атомарность означает, что требования дальше нельзя разбить или уточнить. Например, пользователь может авторизоваться с помощью <конкретный список социальных сетей>. Плохое требование:

пользователь может выбрать статью для чтения и прокомментировать ее. Это два требования, которые надо разбивать и конкретизировать дальше.

3. Завершённость. Требование целиком приведено в одном месте документации. Не должно быть такого, чтобы в разных частях документа с требованиями шла речь об одном и том же функционале.

4. Последовательность. Требование не должно противоречить другим требованиям и ограничениям системы. Пример плохого требования: система должна запрашивать из социальных сетей адрес электронной почты пользователя. Дело в том, что социальные сети не дают такой информации. Если требуется адрес электронной почты пользователя, то он должен быть запрошен от самого пользователя, а не от внешних источников.

5. Отслеживаемость (Трассируемость). Это означает, что требования зафиксировано в документации и можно понять откуда оно взялось. Например, требование подтверждения возраста это требование этического комитета заказчика.

6. Актуальность. Это означает, что требование не устарело. Например, требование должно соответствовать современному законодательству. Или техническим реалиям. Вряд ли сейчас найдется много пользователей IE5 и Netscape Navigator.

7. Выполнимость. Требование можно выполнить в рамках существующих технологий. Например, можно требовать, чтобы система давала отклик при маленькой нагрузке (что такое маленькая нагрузка тоже надо конкретизировать) в течение не более 3-х секунд, но нельзя требовать 1 миллисекунду при больших нагрузках.

8. Понятность. Это означает, что нельзя использовать жаргон, фраза должна пониматься всеми одинаково. Т.е. фраза "казнить нельзя помиловать" не должна использоваться.

9. Проверяемость. Это означает, что выполнение требования можно проверить. Пример плохого требования: система должна работать быстро. Или сайт должен иметь понятную систему навигации. Или должен использовать интуитивно-понятный интерфейс. Кстати, на будущее, я напишу отдельную статью по поводу того, что не бывает интуитивного-понятного интерфейса.

10. Обязательность. Без выполнения этого требования пользователь не сможет в полной мере использовать систему. Не самое простое условие к требованию, между прочим. Например, рассмотрим интернет-магазин. Обычно, требования расписываются с точки зрения пользователя: посмотреть описание товара, выбрать товар, оформить заказ и прочее. Но в данном случае необходимым требованиями будут возможность добавления нового товара на витрину и удаление с витрины товара, которого нет в наличии.

11. Полнота. Самое сложное требование. Полнота системы требований – свойство, означающее, что совокупность артефактов, описывающих требования, исчерпывающим образом описывает все то, что требуется от разрабатываемой системы. Иначе говоря, надо зафиксировать все, что система

должна делать. Если на начальном этапе это не указать, то можно сильно ошибиться при проектировании и выборе архитектуры. При этом, надо понимать, что очень трудно сразу указать для системы, которую еще только надо спроектировать, весь функционал. Поэтому, часто на первых этапах указывают требования крупными мазками, а не в терминах функциональных требований.

При фиксации требований очень важно не делать такую ошибку, как формулировать требования в виде "как система должна работать". Надо использовать подход "что система должна делать". Пример плохих требований: Пользователь нажимает красную кнопку в углу формы и переходит на следующий этап выбора товара в интернет-магазине. Пример хорошего требования: при выборе определенного товара пользователю предлагается перейти в корзину для оформления заказа или продолжить просмотр каталога. Дело в том, что вариантов реализации может быть множество и не надо заранее себя ограничивать. Более того, очень важно избегать субъективного подхода. Если что-то удобно именно вам, это не означает, что удобно всем.

Практическое занятие 2. ПЛАНИРОВАНИЕ ПРОЕКТА

Эффективное управление программным проектом напрямую зависит от правильного планирования работ, необходимых для его выполнения. План помогает менеджеру предвидеть проблемы, которые могут возникнуть на каких-либо этапах создания ПО, и разработать превентивные меры для их предупреждения или решения. План, разработанный на начальном этапе проекта, рассматривается всеми его участниками как руководящий документ, выполнение которого должно привести к успешному завершению проекта. Этот первоначальный план должен максимально подробно описывать все этапы реализации проекта.

Кроме разработки плана проекта, на менеджера ложится обязанность разработки других видов планов. Эти виды планов кратко описаны в табл. 1.

Таблица 1 - Виды планов

План	**Описание**
План качества	Описывает стандарты и мероприятия по поддержке качества разрабатываемого ПО
План аттестации	Описывает способы, ресурсы и перечень работ, необходимых для аттестации программной системы
План управления конфигурацией	Описывает структуру и процессы управления конфигурацией
План сопровождения ПО	Предлагает план мероприятий, требующихся для сопровождения ПО в процессе его эксплуатации, а также расчет стоимости сопровождения и необходимые для этого ресурсы
План по управлению персоналом	Описывает мероприятия, направленные на повышение квалификации членов команды разработчиков

Процесс планирования начинается с определения проектных ограничений (временные ограничения, возможности наличного персонала, бюджетные ограничения и т.д.). Эти ограничения должны определяться параллельно с оцениванием проектных параметров, таких как структура и размер проекта, а также распределением функций среди исполнителей. Затем определяются этапы разработки и то, какие результаты документация, прототипы, подсистемы или версии программного продукта) должны быть получены по окончании этих этапов. Далее начинается циклическая часть планирования. Сначала разрабатывается график работ по выполнению проекта или дается разрешение на продолжение использования ранее созданного графика. После этого (обычно через 2-3 недели) проводится контроль выполнения работ и отмечаются расхождения между реальным и плановым ходом работ.

Далее, по мере поступления новой информации о ходе выполнения проекта, возможен пересмотр первоначальных оценок параметров проекта. Это, в свою очередь, может привести к изменению графика работ. Если в

результате этих изменений нарушаются сроки завершения проекта, должны быть пересмотрены (и согласованы с заказчиком ПО) проектные ограничения.

Конечно, большинство менеджеров проектов не думают, что реализация их проектов пройдет гладко, без всяких проблем. Желательно описать возможные проблемы еще до того, как они проявят себя в ходе выполнения проекта. Поэтому лучше составлять "пессимистические" графики работ, чем "оптимистические". Но, конечно, невозможно построить план, учитывающий все, в том числе случайные, проблемы и задержки выполнения проекта, поэтому и возникает необходимость периодического пересмотра проектных ограничений и этапов создания программного продукта.

План проекта

План проекта должен четко показать ресурсы, необходимые для реализации проекта, разделение работ на этапы и временной график выполнения этих этапов. В некоторых организациях план проекта составляется как единый документ, содержащий все виды планов, описанных выше. В других случаях план проекта описывает только технологический процесс создания ПО. В таком плане обязательно присутствуют ссылки на планы других видов, но они разрабатываются отдельно от плана проекта.

План, представленный ниже, принадлежит именно к последнему типу планов. Детализация планов проектов очень разнится в зависимости от типа разрабатываемого программного продукта и организации-разработчика. Но в любом случае большинство планов содержат следующие разделы.

1. *Введение.* Краткое описание целей проекта и проектных ограничений (бюджетных, временны*х* и т.д.), которые важны для управления проектом.
2. *Организация выполнения проекта.* Описание способа подбора команды разработчиков и распределение обязанностей между членами команды.
3. *Анализ рисков.* Описание возможных проектных рисков, вероятности их проявления и стратегий, направленных на их уменьшение. Тема управления рисками рассмотрена ниже.
4. *Аппаратные и программные ресурсы, необходимые для реализации проекта.* Перечень аппаратных средств и программного обеспечения, необходимого для разработки программного продукта. Если аппаратные средства требуется закупать, приводится их стоимость совместно с графиком закупки и поставки.
5. *Разбиение работ на этапы.* Процесс реализации проекта разбивается на отдельные процессы, определяются этапы выполнения проекта, приводится описание результатов ("выходов") каждого этапа и контрольные отметки. Эта тема представлена ниже.
6. *График работ.* В этом графике отображаются зависимости между отдельными процессами (этапами) разработки ПО, оценки времени их выполнения и распределение членов команды разработчиков по отдельным этапам.
7. *Механизмы мониторинга и контроля за ходом выполнения проекта.* Описываются предоставляемые менеджером отчеты о ходе

выполнения работ, сроки их предоставления, а также механизмы мониторинга всего проекта.

План должен регулярно пересматриваться в процессе реализации проекта. Одни части плана, например график работ, изменяются часто, другие более стабильны. Для внесения изменений в план требуется специальная организация документопотока, позволяющая отслеживать эти изменения.

Контрольные отметки этапов работ

При планировании процесса определяются контрольные отметки— вехи, отмечающие окончание определенного этапа работ. Для каждой контрольной отметки создается отчет, который предоставляется руководству проекта. Эти отчеты не должны быть большими объемными документами; они должны подводить краткие итоги окончания отдельного логически завершенного этапа проекта. Этапом не может быть, например, "Написание 80% кода программ", поскольку невозможно проверить завершение такого "этапа"; кроме того, подобная информация практически бесполезна для управления, поскольку здесь не отображается связь этого "этапа" с другими этапами создания ПО.

Обычно по завершении основных больших этапов, таких как разработка спецификации, проектирование и т.п., заказчику ПО предоставляются результаты их выполнения, так называемые контрольные проектные элементы. Это может быть документация, прототип программного продукта, законченные подсистемы ПО и т.д. Контрольные проектные элементы, предоставляемые заказчику ПО, могут совпадать с контрольными отметками (точнее, с результатами выполнения какого-либо этапа). Но обратное утверждение неверно. Контрольные отметки — это внутренние проектные результаты, которые используются для контроля за ходом выполнения проекта, и они, как правило, не предоставляются заказчику ПО.

Для определения контрольных отметок весь процесс создания ПО должен быть разбит на отдельные этапы с указанным "выходом" (результатом) каждого этапа. Например, на рис. 1 показаны этапы разработки спецификации требований в случае, когда для ее проверки используется прототип системы, а также представлены выходные результаты (контрольные отметки) каждого этапа. Здесь контрольными проектными элементами являются требования и спецификация требований.

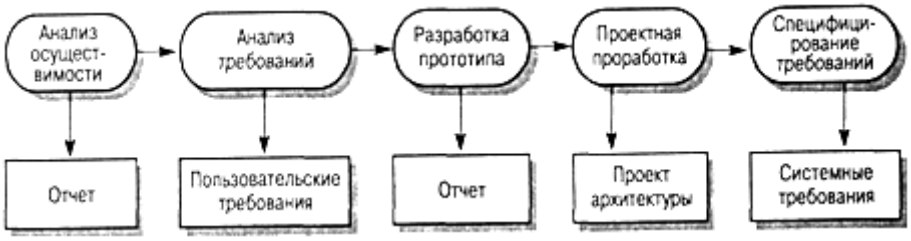


Рис. 1. Этапы процесса разработки спецификации

График работ

Составление графика - одна из самых ответственных работ, выполняемых менеджером проекта. Здесь менеджер оценивает длительность проекта, определяет ресурсы, необходимые для реализации отдельных этапов работ, и представляет их (этапы) в виде согласованной последовательности. Если данный проект подобен ранее реализованному, то график работ последнего проекта можно взять за основу для данного проекта. Но затем следует учесть, что на отдельных этапах нового проекта могут использоваться методы и подходы, отличные от использованных ранее.

В процессе составления графика (рис. 2) весь массив работ, необходимых для реализации проекта, разбивается на отдельные этапы и оценивается время, требующееся для выполнения каждого этапа. Обычно многие этапы выполняются параллельно. График работ должен предусматривать это и распределять производственные ресурсы между ними наиболее оптимальным образом. Нехватка ресурсов для выполнения какого-либо критического этапа - частая причина задержки выполнения всего проекта.

При расчете длительности этапов менеджер должен учитывать, что выполнение любого этапа не обойдется без больших или маленьких проблем и задержек. Разработчики могут допускать ошибки или задерживать свою работу, техника может выйти из строя либо аппаратные или программные средства поддержки процесса разработки могут поступить с опозданием. Если проект инновационный и технически сложный, это становится дополнительным фактором появления непредвиденных проблем и увеличения длительности реализации проекта по сравнению с первоначальными оценками.

Требования к ПО

Диаграммы процессов и временные диаграммы

Рис. 2.- Процесс составления графика работ

Кроме временных затрат, менеджер должен рассчитать другие ресурсы, необходимые для успешного выполнения каждого этапа. Особый вид ресурсов — это команда разработчиков, привлеченная к выполнению проекта. Другими видами ресурсов могут быть необходимое свободное дисковое пространство на сервере, время использования какого-либо специального оборудования и бюджетные средства на командировочные расходы персонала, работающего над проектом.

Существует хорошее эмпирическое правило: оценивать временные затраты так, как будто ничего непредвиденного и "плохого" не может случиться, затем увеличить эти оценки для учета возможных проблем. Возможные, но трудно прогнозируемые проблемы существенно зависят от

типа и параметров проекта, а также от квалификации и опыта членов команды разработчиков. К исходным расчетным оценкам рекомендуется добавлять 30% на возможные проблемы и затем еще 20%, чтобы быть готовым к тому, что невозможно предвидеть.

График работ по проекту обычно представляется в виде набора диаграмм и графиков, показывающих разбиение проектных работ на этапы, зависимости между этапами и распределение разработчиков по этапам.

Временны*е* и сетевые диаграммы

Временные и сетевые диаграммы полезны для представления графика работ. Временная диаграмма показывает время начала и окончания каждого этапа и его длительность. Сетевая диаграмма отображает зависимости между различными этапами проекта. Эти диаграммы можно создать автоматически с помощью программных средств поддержки управления на основе информации, заложенной в базе данных проекта.

Рассмотрим этапы некоего проекта, представленные в табл. 2, из которой, в частности, видно, что этап Т3 зависит от этапа Т1. Это значит, что этап Т1 должен завершиться прежде, чем начнется этап Т3. Например, на этапе Т1 проводится компонентный анализ создаваемого программного продукта, а на этапе Т3 — проектирование системы.

На основе приведенных значений длительности этапов и зависимости между ними строится сетевой график последовательности этапов (рис. 3). На этом графике видно, какие работы могут выполняться параллельно, а какие должны выполняться последовательно друг за другом. Этапы обозначены прямоугольниками. Контрольные отметки и контрольные проектные элементы показаны в виде овалов и обозначены (как и в табл. 2) буквой М с соответствующим номером. Даты на данной диаграмме соответствуют началу выполнения этапов. Сетевую диаграмму следует читать слева направо и сверху вниз.

Таблица 2 - Этапы проекта

Этап	**Длительность (дни)**	**Зависимость**
Т1	8	
Т2	15	
Т3	15	Т1 (М1)
Т4	10	
Т5	10	Т2, Т4 (М2)
Т6	5	Т1, Т2 (М3)
Т7	20	Т1 (М1)
Т8	25	Т4 (М5)
Т9	15	Т3, Т6 (М4)
Т10	15	Т5, Т7 (М7)
Т11	7	Т9 (М6)
Т12	10	Т11 (М8)

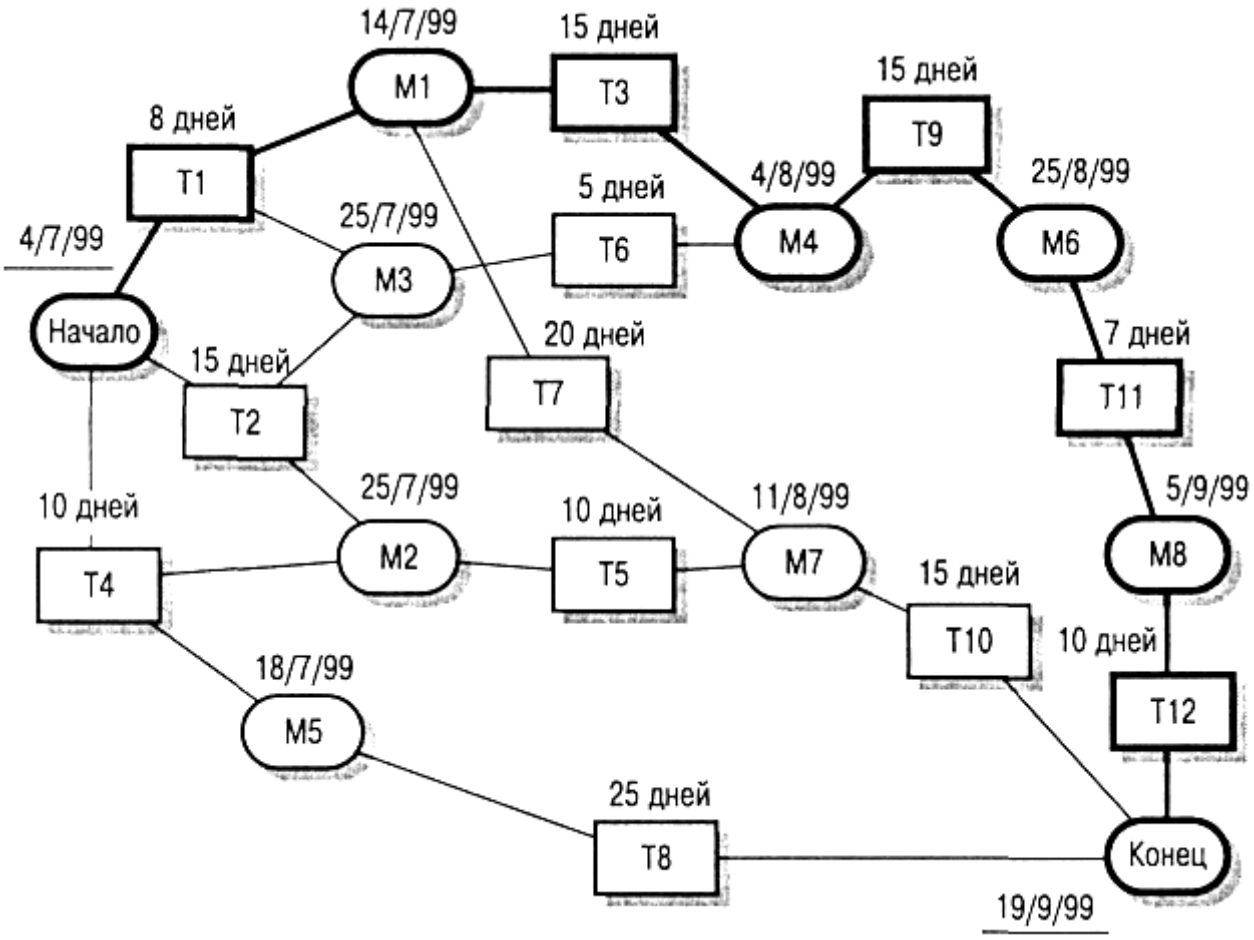


Рис. 3. Сетевая диаграмма этапов

Если для создания сетевой диаграммы используются программные средства поддержки управления проектом, каждый этап должен заканчиваться контрольной отметкой. Очередной этап может начаться только тогда, когда будет получена контрольная отметка (которая может зависеть от нескольких предшествующих этапов). Поэтому в третьем столбце табл. 2 приведены контрольные отметки; они будут достигнуты только тогда, когда будет завершен этап, в строке которого помещена соответствующая контрольная отметка.

Любой этап не может начаться, пока не выполнены все этапы на всех путях, ведущих от начала проекта к данному этапу. Например, этап Т9 не может начаться, пока не будут завершены этапы Т3 и Т6. Отметим, что в данном случае достижение контрольной отметки М4 говорит о том, что эти этапы завершены.

Минимальное время выполнения всего проекта можно рассчитать, просуммировав в сетевой диаграмме длительности этапов на самом длинном пути (длина пути здесь измеряется не количеством этапов на пути, а суммарной длительностью этих этапов) от начала проекта до его окончания (это так называемый критический путь). В нашем случае продолжительность проекта составляет 11 недель или 55 рабочих дней. На рис. 3 критический путь показан более толстыми линиями, чем остальные пути. Таким образом, общая продолжительность реализации проекта зависит от этапов работ, находящихся

на критическом пути. Любая задержка в завершении любого этапа на критическом пути приведет к задержке всего проекта.

Задержка в завершении этапов, не входящих в критический путь, не влияет на продолжительность всего проекта до тех пор, пока суммарная длительность этих этапов (с учтом задержек) на каком-нибудь пути не превысит продолжительности работ на критическом пути. Например, задержка этапа Т8 на срок, меньший 20 дней, никак не влияет на общую продолжительность проекта. На рис. 4 представлена временная диаграмма, на которой показаны возможные задержки на каждом этапе.

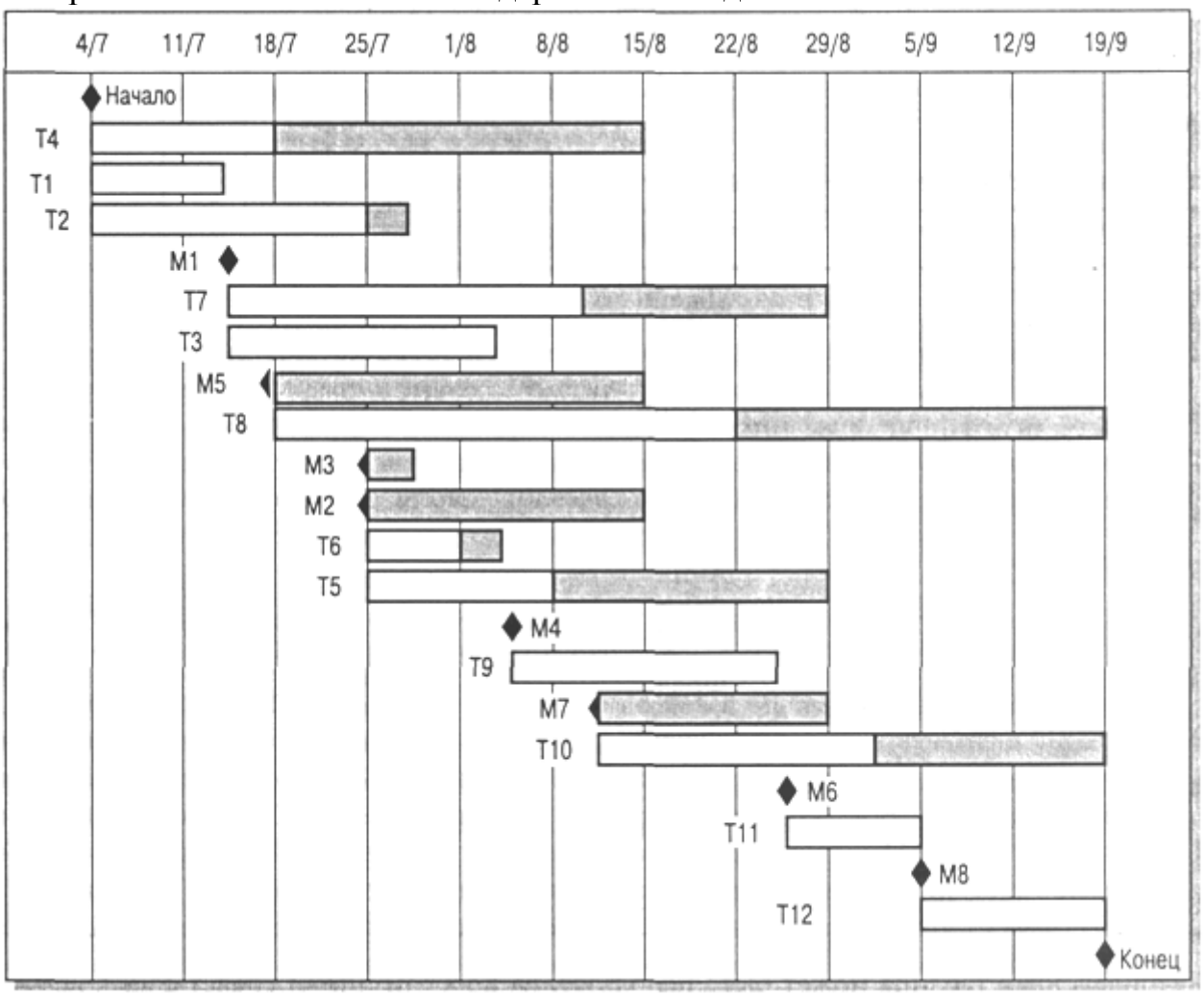


Рис. 4. Временная диаграмма длительности этапов

Сетевая диаграмма позволяет увидеть в зависимости этапов значимость того или иного этапа для реализации всего проекта. Внимание к этапам критического пути часто позволяет найти способы их изменения с тем, чтобы сократить длительность всего проекта. Менеджеры используют сетевую диаграмму для распределения работ.

На рис. 4 показано другое представление графика работ. Это временная диаграмма (иногда называемая по имени ее изобретателя диаграммой Гантта) может быть построена программными средствами поддержки процесса управления. Она показывает длительность выполнения каждого этапа и возможные их задержки (показаны затененными прямоугольниками), а также

даты начала и окончания каждого этапа. Этапы критического пути не имеют затененных прямоугольников; это означает, что задержка с завершением данных этапов приведет к увеличению длительности всего проекта.

Подобно распределению времени выполнения этапов, менеджер должен рассчитать распределение ресурсов по этапам, в частности назначить исполнителей на каждый этап. В табл. 3 приведено распределение разработчиков на каждый этап, представленный на рис. 4.

Таблица 3 - Распределение исполнителей по этапам

Этап	**Исполнитель**
T1	Джейн
T2	Анна
T3	Джейн
T4	Фред
T5	Мэри
T6	Анна
T7	Джим
T8	Фред
T9	Джейн
T10	Анна
T11	Фред
T12	Фред

Приведенная таблица может быть использована программными средствами поддержки процесса управления для построения временной диаграммы занятости сотрудников на определенных этапах работ (рис. 5). Персонал не занят в работе над проектом все время его реализации. В течение периода незанятости сотрудники могут быть в отпуске, работать над другими проектами, проходить обучение и т.д.

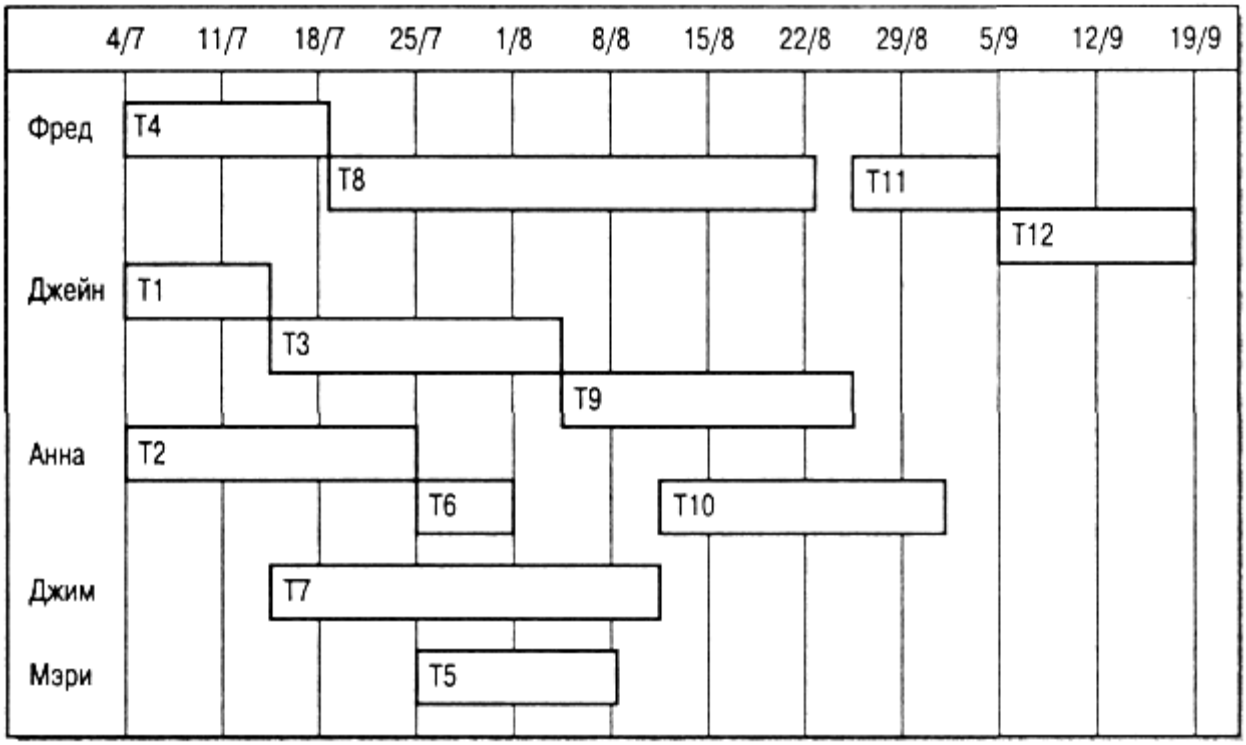


Рис. 5. Временная диаграмма распределения работников по этапам

В больших организациях обычно работает много специалистов, которые задействуются в проекте по мере необходимости. Конечно, такой подход может создать определенные проблемы для менеджеров проектов. Например, если специалист занят в проекте, который задерживается, это может создать прямые сложности для других проектов, где он также должен участвовать.

Первоначальный график работ неизбежно содержит какие-нибудь ошибки или недоработки. По мере реализации проекта рассчитанные оценки длительности выполнения этапов работ должны сравниваться с реальными сроками выполнения этих этапов. Результаты сравнения должны использоваться в качестве основы для пересмотра графика работ еще не реализованных этапов проекта, в частности для того, чтобы попытаться уменьшить длительность этапов критического пути.

Вопросы к зачету

1. Определение веб-проекта
2. Понятие проекта, основные и дополнительные признаки проекта
3. ключевые концепции проектного менеджмента
4. Проект и организационная структура организации
5. Особенности электронной экономики
6. Типы электронного бизнеса
7. Концепция SMART достижения целей
8. Тройственное ограничение ресурсов проекта
9. Концепция жизненного цикла проекта
10. Типы веб-проектов, их особенности
11. Корпоративный сайт
12. Сайт-визитка
13. Интернет-магазин
14. Структура техзадания на создание веб-проекта
15. Модель разработки программного обеспечения:
16. Водопадная модель
17. V –образная модель
18. Модель прототипирования
19. Спиральная модель
20. Инкрементная модель
21. Итеративная модель
22. Экстремальное программирование
23. Agile-модели
24. Методология SCRUM
25. Роли и участники типового проекта разработки ПО
26. Типы проектных структур
27. Этапы командообразования
28. Мотивация персонала
29. Иерархия потребностей Маслоу
30. Мотивирующие факторы при выборе места работы в зависимости от выбора должности
31. Понятие риска проекта
32. Методики управления риском проекта
33. Понятие качества проекта
34. Методики оценки трудоемкости проекта
35. График Ганта
36. Оценка трудоемкости проекта методом PERT
37. Оценка трудоемкости проекта методом аналогий
38. Оценка трудоемкости проектам методами COCOMO

Литература

1. А.Н. Павлов «Управление проектами на основе стандарта PMI PMBOK®. Изложение методологии и опыт применения», 3-е изд., – М.: БИНОМ, 2014, – 217 с.: ил. – ISBN 978-5-9963-0930-6.
2. Джефф Сазерленд «Scrum. Революционный метод управления проектами», Правообладатель: Манн, Иванов и Фербер (МИФ), 2015, -- 320 с.: 15 ил. – ISBN 978-5-00057-722-6
3. «Руководство к своду знаний по программной инженерии». The Guide to the Software Engineering Body of Knowledge, SWEBOK, IEEE Computer Society Professional Practices Committee, 2004.
4. Скотт Беркун, «Искусство управления IT-проектами», пер. с англ., СПб., Питер-Пресс, 2007. – 400 с. ISBN 978-5-91180-005-5
5. Сергей Архипенков «Лекции по управлению программными проектами», 2009.
6. Брукс Фредерик, "Мифический человеко-месяц, или Как создаются программные комплексы", Пер. с англ., СПб., Символ-Плюс, 1999.
7. А. Якобсон, Г. Буч, Дж. Рамбо Унифицированный процесс разработки программного обеспечения. – СПб.: ПИТЕР, 2002. – 496с. ISBN 5-318-00358-3
8. Киреев Н.Б. «Менеджмент в IT. Курс лекций». Мультимедийное пособие, ИИТ БГУИР, Минск, 2012 г.
9. Остервальдер, И. Пинье «Построение бизнес-моделей: Настольная книга стратега и новатора». М., «Альпина Паблишер», 2017
10. К. Вигерс «Разработка требований к программному обеспечению», Москва, «Русская Редакция», 2004
11. А. Перерва, В. Иванова «Путь аналитика. Практическое руководство IT-специалиста» 2-е изд., ПИТЕР, 2015
12. Грашина, М. Основы управления проектами / М. Грашина, В. Дункан. - Санкт-Петербург : Питер, 2016. - 208 с.: ил.
13. Снисаренко, С. В. Технология разработки программного обеспечения систем управления. Лабораторный практикум [+ электр. вариант] : учебное пособие для студентов вузов [рек. УМО РБ] / С. В. Снисаренко, Н. А. Стасевич, Н. А. Капанов. - Минск : БГУИР, 2018. - 76 с. : ил. - (Кафедра систем управления).