Министерство образования Республики Беларусь

Учреждение образования
«Гомельский государственный технический
университет имени П. О. Сухого»

Институт повышения квалификации и переподготовки

Кафедра «Профессиональная переподготовка»

А. Н. Осипенко

УПРАВЛЕНИЕ WEB-ПРОЕКТАМИ

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ
по одноименному курсу для слушателей специальности 1-40 01 74 «Web-дизайн и компьютерная графика» заочной формы обучения

Гомель 2016

УДК 681.3.06(075.8)
ББК 32.973.202я73
О-74

Рекомендовано к изданию кафедрой «Профессиональная переподготовка» ИПКиП ГГТУ им. П. О. Сухого
(*протокол № 4 от 14.12.2015 г.*)

Рецензент: зав. каф. «Информатика» ГГТУ им. П. О. Сухого
канд. физ.-мат. наук, доц. *Т. В. Тихоненко*

Осипенко, А. Н.

О-74 Управление Web-проектами : учеб.-метод. пособие по одноим. курсу для слушателей специальности 1-40 01 74 «Web-дизайн и компьютерная графика» заоч. формы обучения / А. Н. Осипенко. – Гомель : ГГТУ им. П. О. Сухого, 2016. – 75 с. – ─ Систем. требования: PC не ниже Intel Celeron 300 МГц ; 32 Mb RAM ; свободное место на HDD 16 Mb ; Windows 98 и выше ; Adobe Acrobat Reader. – Режим доступа: https://elib.gstu.by. – Загл. с титул. экрана.

Пособие содержит основные теоретические сведения, необходимые для понимания принципов управления Web-проектами, а также для организации эффективной работы менеджера Web-проекта.

Издание адресовано слушателям ИПКиП специальности 1-40 01 74 «Web-дизайн и компьютерная графика».

УДК 681.3.06(075.8)
ББК 32.973.202я73

Оглавление

Предисловие

Веб-разработки являются перспективным направлением деятельности в современной информационной сфере. Стремительный рост числа пользователей интернета и открытие большого числа сайтов привело к тому, что создание и поддержка веб-сайтов стало востребованным направлением IT-технологий. Вместе с тем, в данный момент создание сайтов происходит по далеким от совершенства методикам. В частности, почти полностью отсутствует менеджмент веб-проектов.

Настоящее пособие разработано в соответствие с программой дисциплины «Управление Web-проектами» для обучения слушателей ИПК и П заочной формы обучения по специальности 1-40 01 74 «Web-дизайн и компьютерная графика».

Целью методического пособия является повышение профессионального уровня будущих специалистов по веб-дизайну в области менеджмента веб-проектов. Задачей методического пособия является формирование у слушателей современных представлений о разработке веб-проектов, привлечении кадров, распределении времени, работе с заказчиками, а также решении проблем эффективного использования современных инструментов разработки веб-сайтов на основе мирового опыта веб-дизайна, верстки и веб-программирования.

1 Менеджмент в разработке программных изделий

В общем виде проект (англ. project) – это «что-либо», что задумывается или планируется, например, программное изделие.

С точки зрения системного подхода, проект может рассматриваться как процесс перехода из исходного состояния в конечное – результат при участии ряда ограничений и механизмов.

В «Кодексе знаний об управлении проектами» проект – некоторая задача с определенными исходными данными и требуемыми результатами, обуславливающими способ ее решения. Проект включает в себя замысел, средства его реализации и получаемые в процессе реализации результаты.

Методы управления проектами позволяют: определить цели проекта и пронести его обоснование; выявить структуру проекта; определить необходимые объемы и источники финансирования; подобрать исполнителей через процедуры торгов и конкурсов; подготовить и заключить контракты; определить сроки выполнения проекта, составить график его реализации, рассчитать необходимые ресурсы; рассчитать смету и бюджет проекта; планировать и учитывать риски; обеспечить контроль за ходом выполнения проекта.

Разработка программного обеспечения в большинстве случаев должна рассматриваться как коллективный труд специалистов, направленный на удовлетворение потребности пользователей в автоматизации их деятельности. Как и любой другой коллективный труд, она требует организации, в частности – *управления*. Это процесс, порою длительный, связывающий производственными и иными отношениями тех, кого в той или иной степени можно рассматривать в качестве производителей программы. Как и любой труд, тесно связанный с неоднозначными потребностями тех, кто будет использовать продукты труда, необходимым элементом *разработки программ* является решение задач изучения пользователей, с одной стороны, а с другой – обеспечения обратной связи с ними, направляющей производство. Это составляющие, из которых формируются главные задачи управления производством программ. Чаще всего решение таких задач осуществляется руководителем, или, как принято говорить, ***менеджером проекта***.

Понятие "***менеджер проекта***" не обязательно соотносится с конкретной персоной, отвечающей за управление производством программной системы в целом. В небольшом проекте такое

единоначалие чаще всего оправданно: оно позволяет концентрировать все нити управления, исключает проблемы внутреннего для проекта согласования противоречий, обеспечивает централизованную ответственность за проект перед теми, кто заинтересован в его выполнении. Однако по мере перехода к более масштабным проектам менеджерские обязанности становится невозможно концентрировать в одних руках. Обычно в таких случаях образуют ***службу менеджера***, состоящую из его помощников, которым он может поручать различные задания, освобождая себя от рутины постоянного контроля. *Делегирование* можно считать основным инструментом разделения труда в проекте (не только в случае *менеджмента*), когда есть ответственность за некоторую функцию (работу и др.), но для ее выполнения нет собственных ресурсов, а потому приходится прибегать к помощи.

Для небольших групп возможна следующая организация работ: обязанности главного менеджера распределяются по команде разработчиков, которая за счет внутренних механизмов решает, как планировать работы, как их распределять и контролировать. Это характерно для так называемого подхода быстрой *разработки (agile development) программных систем*, объединившего в себе несколько методологий программирования, которые отказываются от многих принципов традиционного *менеджмента*. Наиболее ярким примером подхода быстрой разработки является экстремальное программирование. Отметим, что здесь, как и в других случаях, менеджерские задачи не исчезают, как могло бы показаться на первый взгляд. Просто форма и методы их решения приобретают другой характер.

В работе менеджера всегда присутствуют и неразрывно связаны друг с другом два аспекта:

- *управление проектом* как деятельность, продвигающая процесс производства к определенным целям,
- *руководство людьми*, *участниками разработки*.

С организационной точки зрения в *разработке программного обеспечения* можно выделить три варианта целей, определяющие деятельность *менеджера*:

- **производство программ не для продажи**, напрямую не связанное с получением дохода.
- **производство *рыночного продукта***, обеспечивающего прибыль за счет распространения (продажи) получаемых результатов.

• **разработка ведется под заказ**, когда все производство программы, от стадии замысла до передачи в эксплуатацию, финансируется внешними по отношению к разработчикам, но весьма заинтересованными агентами, обычно называемыми заказчиками.

Главная и постоянная задача *менеджмента разработки программного обеспечения* – продвижение проекта к обозначенным в начале его развития результатам. Если оставить в стороне приемы и методы, с помощью которых достигается решение этой задачи, то она сводится к распределению и контролю эффективного использования имеющихся ресурсов проекта: времени, финансов, технических средств и производственного потенциала работников. Множественность критериев, необходимость согласования интересов участников проекта и его заказчиков, разнообразие видов деятельности, составляющих развитие проекта, – вот тот организационный и производственный контекст, который вынужден учитывать менеджер.

Характерной особенностью *разработки программного обеспечения* является стремление к *переиспользованию* ранее созданных программных компонентов. Задачи *переиспользования* – это:

• во-первых, определение программных продуктов или каких-либо иных изделий и инструментов, имеющихся в распоряжении разработчиков, использование которых могло бы способствовать прогрессу развития проекта,

• во-вторых, выявление компонентов данного проекта, которые было бы полезно задействовать в других разработках.

Второе дополнительное направление – это задача *распространения построенного программного продукта*, в частности, продвижение сайта. Если с самого начала не рассматривать ее и относить распространение лишь к этапам, следующим за разработкой, есть опасность сделать не то, что нужно потребителю продукции, и выпустить из рук рычаги, с помощью которых можно влиять на сферу возможного применения создаваемых программных продуктов.

С точки зрения управления ***участники проекта*** – это абстрактные действующие агенты, которые выполняют заданные функции. Здесь под функцией понимается некое действие, при выполнении которого потребляются определенные ресурсы и производится определенный результат. Функциональный взгляд на *участников разработки проекта* делает их взаимозаменяемыми,

обезличенными в пределах компетенции, соответствующей выполнимости функции. Он приводит к понятию роли, назначаемой работнику для выполнения соответствующих обязанностей.

Ключевым качеством коллектива, определяющим его успешность, является слаженность. В идеальном коллективе все понимают друг друга с полуслова, есть взаимопонимание и уважение, не происходят или сведены к минимуму внутренние конфликты и противоречия. В реальности менеджеру редко приходится иметь дело с таким коллективом, а значит, необходимы меры, способствующие не только росту индивидуальной квалификации работников, но и дееспособности формируемой команды в целом. Эти меры следует рассматривать в качестве задач менеджера как руководителя коллектива.

Таким образом, при обсуждении задач менеджера в разработке программных изделий мы сталкиваемся с тремя взаимосвязанными *направлениями его деятельности*.

- Первое направление – это те *функции*, которые необходимо выполнять для успешного развития проекта. Здесь следует выяснить, какие роли сотрудников требуются для данного проекта.
- Второе направление – планирование и контроль хода проекта в соответствии с *жизненным циклом создаваемого программного обеспечения*.
- Наконец, третье направление определяется кругом задач формирования коллектива и, в частности, кадровым обеспечением проекта.

Вопросы для самоконтроля:

1. Назовите три варианта целей, определяющих деятельность менеджера.
2. Назовите основные задачи менеджера по разработке программных проектов.
3. Назовите основные направления деятельности менеджера по разработке программных проектов.

2 Роли в коллективе разработчиков и определение кадрового ресурса

2.1 Функциональные роли в коллективе разработчиков

Для целенаправленного выполнения проекта должен быть выполнен ряд работ, различных как по своему назначению, так и по квалификационным требованиям, предъявляемым к *разработчикам*. Иными словами, в ходе развития проекта командой *разработчиков* выполняются те или иные функции.

Функции, выполняемые *разработчиками*, – понятие неформализованное. В разных проектах оно может обретать свое содержание. Тем не менее, *типовые функции*, которые предполагают практически все программные проекты, можно перечислить. Так, в любом программном проекте есть функции кодирования, т.е. записи программы на алгоритмическом языке по имеющимся спецификациям, анализа требований, т.е. выявления истинной потребности в создаваемой программе, тестирования и отладки. Далее мы рассмотрим эти и другие функции *разработчиков*, связанные с проектом, а пока лишь заметим, что в рамках деятельности менеджера любого проекта необходимо организовать *распределение функций* проекта между исполнителями. Вполне естественно считать эти действия одной из функций менеджера. В результате ее выполнения члены команды, выполняющей проект, начинают играть соответствующие роли.

Таким образом, в рамках любого проекта возникает задача повышения квалификации сотрудников. Для различных схем ведения проектов эта задача решается по-разному. Крайнюю точку зрения на проблему соответствия квалификации работников требованиям проекта отражает идея ***экстремального программирования***, когда используется неформальная организация группы исполнителей проекта без четкого распределения ролей, а значит, и обязательств сотрудников. В этом случае провозглашается принцип, когда каждый в группе делает то, что он умеет делать лучше всего. И хотя все функции, которые должны выполняться, остаются, создается впечатление, что в группе исполнителей проекта исчезают роли. В результате возможны пробелы в разработке, в частности при анализе и декомпозиции проектируемой системы. Чтобы этого избежать, *разработчики* должны понимать, какую абстрактную роль они исполняют в каждый конкретный момент, выполнение каких

проектных функций необходимо сейчас, как связаны между собой работы, как должны распределяться ресурсы. Словом, они должны обращать внимание на выполнение распределенных по группе менеджерских функций. И даже в этом случае схему *экстремального программирования* можно рекомендовать лишь для слаженных групп исполнителей с высоким уровнем коллективной ответственности.

В *модели проектной группы концепции Microsoft Solution Framework (MSF)* предлагается образовывать небольшие мобильные коллективы как атомарные производственные единицы с общей ответственностью за выполняемые задания – так называемые проектные группы. Такие группы строятся как многопрофильные команды, члены которых распределяют между собой ответственность и дополняют *области компетентности* друг друга. Группа состоит не более чем из 10 человек. Все они считаются обладающими сходным уровнем профессиональной подготовки, хотя и в разных областях индивидуальной специализации. Провозглашается равноправие членов группы и коллективная ответственность за выполняемые задания: проектная группа – команда равных. Все это позволяет сохранять внутри группы неформальные отношения.

Вместо понятия роли для группы в целом определяются ***ролевые кластеры***, которые заполняются точно так же, как происходит распределение ролей. В то время как за успех проекта ответственна вся команда, каждый из ее *ролевых кластеров*, определяемых моделью, ассоциирован с одной из проектных целей и работает над ее достижением. В данной модели именно эти цели задают роли *разработчиков*, которые определяются кластерами. В терминологии MSF используется понятие *области компетенции*, или *области функциональной специализации* (functional area), обозначающее ту или иную роль, которую выполняет кластер группы в проекте. Принципиальное отличие распределения исполнителей по *ролевым кластерам* от распределения ролей заключается лишь в том, что ответственность за это несет не *менеджер проекта*, а сама группа. Менеджер проекта выдает задания и контролирует их выполнение лишь в целом для группы, не вмешиваясь в ее работу.

Определено шесть *ролевых кластеров*, которые соответствующим образом структурируют проектные функции *разработчиков* (рисунок 2.1).

Рисунок 2.1 – Ролевые кластеры модели проектной групп MSF

- ***Управление продуктом*** (product management). Ключевая цель кластера – обеспечивать удовлетворение интересов *заказчика*. Для ее достижения кластер должен содержать следующие области компетенции:
  - планирование продукта,
  - планирование доходов,
  - представление интересов *заказчика*,
  - маркетинг.
- ***Управление программой*** (program management). Задача – обеспечить реализацию решения в рамках ограничений проекта, что может рассматриваться как удовлетворение требований к бюджету проекта и к его результату. *Области компетенции* кластера:
  - управление проектом;
  - выработка архитектуры решения;
  - контроль производственного процесса;
  - административные службы.
- ***Разработка*** (development). Первостепенной задачей кластера является построение решения в соответствии со спецификацией. *Области компетенции* кластера:
  - технологическое консультирование;
  - проектирование и осуществление реализации;
  - разработка приложений;
  - разработка инфраструктуры.
- ***Тестирование*** (test). Задача кластера – одобрение выпуска продукта только после того, как все дефекты выявлены и устранены. *Области компетенции* кластера:
  - разработка тестов;
  - отчетность о тестах;

- планирование тестов.

• ***Удовлетворение потребителя*** (user experience). Цель кластера – повышение эффективности использования продукта. *Области компетенции* кластера: – общедоступность (возможности работы для людей с недостатками зрения, слуха и др.);

- интернационализация (эксплуатация в иноязычных средах);
- обеспечение технической поддержки;
- обучение пользователей;
- удобство эксплуатации (эргономика);
- графический дизайн.

• ***Управление выпуском*** (release management). Задача кластера – беспрепятственное внедрение и сопровождение продукта. *Области компетенции* кластера:

- инфраструктура (infrastructure);
- сопровождение (support);
- бизнес-процессы (operations);
- управление выпуском готового продукта (commercial release management).

Мы придерживаемся ролевой структуры проекта, которая предложена Центром объектно-ориентированной технологии компании IBM. Эта структура включает достаточно полный перечень типичных ролей, согласованный со многими реальными дисциплинами развития программных проектов. В то же время она представляет роли *разработчиков* в организационном контексте, т.е. рассматривает не только *разработчиков*, но и тех, кто, не участвуя в проекте в качестве исполнителей, оказывает влияние на постановку задач проекта, на выделение ресурсов и обеспечение осуществимости развития работ.

• ***Заказчик*** (Customer) – реально существующий (в организации, которой подчинена команда, или вне ее) инициатор разработки или кто-либо иной, уполномоченный принимать результаты (как текущие, так и окончательные) разработки.

• ***Планировщик ресурсов*** (Planner) – выдвигает и координирует требования к проектам в организации, осуществляющей данную разработку, а также развивает и направляет план выполнения проекта с точки зрения организации.

• ***Менеджер проекта*** (Project Manager) – отвечает за развитие проекта в целом, гарантирует, что распределение заданий и ресурсов позволяет выполнить проект, что работы и предъявление результатов

идут по графику, что результаты соответствуют требованиям. В рамках этих функций *менеджер проекта* взаимодействует с *заказчиком* и *планировщиком ресурсов*.

- ***Руководитель команды*** (Team Leader) – производит техническое руководство командой в процессе выполнения проекта. Для больших проектов возможно привлечение нескольких руководителей подкоманд, отвечающих за решение частных задач.
- ***Архитектор*** (Architect) – отвечает за проектирование архитектуры системы, согласовывает развитие работ, связанных с проектом.
- ***Проектировщик подсистемы*** (Designer) – отвечает за проектирование подсистемы или категории классов, определяет реализацию и интерфейсы с другими подсистемами.
- ***Эксперт предметной области*** (Domain Expert) – отвечает за изучение сферы приложения, поддерживает направленность проекта на решение задач данной области.
- ***Разработчик*** (Developer) – реализует проектируемые компоненты, владеет и создает специфичные классы и методы, осуществляет кодирование и автономное тестирование, строит продукт. Это широкое понятие, которое может подразделяться на специальные роли (например, *разработчик* классов). В зависимости от сложности проекта команда может включать различное число *разработчиков*.
- ***Разработчик информационной поддержки*** (Information Developer) – создает документацию, сопровождающую продукт, когда выпускается версия. Включаемые в нее инсталляционные материалы, равно как ссылочные и учебные, а также материалы помощи предоставляются на бумажных и машинных носителях. Для сложных проектов возможно распределение этих задач между несколькими *разработчиками информационной поддержки*.
- ***Специалист по пользовательскому интерфейсу*** (Human Factors Engineer) – отвечает за удобство применения системы. Работает с *заказчиком*, чтобы удостовериться, что пользовательский интерфейс удовлетворяет требованиям.
- ***Тестировщик*** (Tester) – проверяет функциональность, качество и эффективность продукта. Строит и исполняет тесты для каждой фазы развития проекта.

• ***Библиотекарь*** (Librarian) – отвечает за создание и ведение общей библиотеки проекта, которая содержит все проектные рабочие продукты, а также за соответствие рабочих продуктов стандартам.

Взаимодействие менеджера с *заказчиком*, *планировщиком* и другими *инициаторами работ* – прямая обязанность *менеджера проекта*. Таким образом, для него закреплен круг функций, которые являются внешними по отношению к работам над проектом.

Выполнение *внешних функций* создает лишь условия для разработки. Как видно из перечня ролей, другой круг функций, возлагаемый на менеджера, связан с взаимодействиями с действующими лицами из команды *разработчиков* проекта. Это ***внутренние функции менеджера***.

В зависимости от проекта и условий его выполнения роли участников проекта могут совмещаться. Предельный случай – программист разрабатывает проект для себя (по собственному заказу), сам планирует распределение ресурсов (сроки выполнения работы, использование вычислительной техники и др.), сам принимает проектные решения (управляет и руководит собою) и сам же занимается разработкой, экспертизой и обслуживанием.

В большинстве случаев *заказчик* и *планировщик ресурсов* являются действительно внешними по отношению к проекту действующими лицами, а потому *совмещение этих ролей* с другими – нечто экзотическое. Тем не менее, роль *заказчика* как члена коллектива *разработчиков*, аккумулирующего точки зрения всех *инициаторов работ*, весьма полезна. В частности, подход *экстремального программирования* считает это обязательным, чтобы развитие проекта всегда гарантированно было направлено в сторону, нужную пользователям.

Менеджер проекта по своему назначению является выделенным в команде. Он берет на себя взаимодействие с *заказчиком* и *планировщиком ресурсов*, с одной стороны, а с другой – распределяет работы среди членов команды. Последнее означает, что он должен обладать полной информацией о декомпозиции проекта. Как следствие, совмещение его роли с ролью *архитектора* проекта является весьма желательным, а потому, довольно частым. Единственным условием для такого совмещения является требование четко знать, когда и чьи функции выполняет данное действующее лицо.

Вполне возможно продуктивное *совмещение ролей руководителя команды* и *архитектора* – это дает ощутимые результаты, когда команда достаточно крепко связана со своим руководителем, например, по предшествующим работам, но при условии не слишком высокой сложности задач декомпозиции для данного проекта.

Совмещение ролей *руководителя команды* и менеджера допустимо, но лишь тогда, когда осознается и учитывается противоречивость целевых установок этих ролей: *руководитель команды* действует в условиях, которые формируются менеджером.

Нежелательно *совмещение ролей руководителя команды* и *проектировщика* какой-либо подсистемы. И это обусловлено противоречивостью их ролевых интересов.

Если используется *перекрестное совмещение ролей руководителя команды* и *проектировщика подсистемы*, то возникает еще одно противоречие их ролевых интересов: у руководителя могут быть предпочтения в пользу "своего" компонента и, как следствие, возможен дисбаланс проекта в целом в пользу выделенных его составляющих, который придется компенсировать дополнительными усилиями менеджера.

По тем же причинам не допускается *совмещение ролей* менеджера и *разработчика*. Здесь запрет более жесткий, поскольку контрольные функции менеджера несовместимы с исполнительскими задачами *разработчика*. Даже в тех случаях, когда у менеджера остается свободное время после выполнения своих прямых обязанностей, ему не следует "помогать" *разработчикам*. Лучше занять себя другим делом, в частности выступить в роли *разработчика* другого проекта.

В то же время *совмещение ролей различных разработчиков* – обычное дело для больших проектов. Оно является частью распределения работ по исполнителям. Решают эту задачу совместно *руководитель команды* и менеджер, когда основные архитектурные положения утверждены. Способы решения зависят от того, как формируется команда. Если разработка поручается готовой команде, то возрастает роль руководителя, а менеджер лишь утверждает его предложение. Когда команда создается для проекта, растет удельный вес менеджерских функций в подборе кадров для работ. Но в любом случае при *совмещении ролей различных разработчиков* нужно учитывать уровень квалификации работников, их предпочтения и

другие индивидуальные особенности. Необходимо знать, что, выделяя одному действующему лицу несколько работ, следует стремиться к однородности заданий и указывать их сравнительные приоритеты.

Допустимы и другие *совмещения ролей*. Так, довольно часто создание документации распределяется между всеми исполнителями проекта, а на *руководителя команды* и менеджера возлагается задача интеграции документов. Для обозримых проектов функции библиотекаря можно поручить одному из *разработчиков*, соответственно скорректировав его индивидуальное задание. Специалистом по пользовательскому интерфейсу вполне может быть, например, менеджер, поскольку именно он осуществляет контакты с *заказчиком* (в данной ситуации *заказчик* либо сам является пользователем, либо выступает как представитель пользователя программного изделия).

По поводу *тестировщиков* необходимы некоторые разъяснения. Следует различать два вида тестирования: тестирование модулей, автономное по своей сути, и тестирование предоставляемых функций, которое может быть и комплексным (проверка совместного выполнения функций), и автономным, когда проверяется работоспособность отдельного аспекта системы. В первом случае задачи *тестировщика* невозможно выполнить без знания не только назначения, но и структуры проверяемого модуля. Лучше всего в этом разбирается *разработчик* модуля, а значит, именно ему этот модуль и отлаживать. Автономное тестирование второго вида, по существу, есть другая сторона проверки работоспособности модуля, реализующего определенную функцию. Во многих случаях даже нет нужды различать его и тестирование модуля. Поэтому и здесь разумно говорить о деятельности *разработчика*. В обоих случаях самотестирование для *разработчика* есть не дополнительная роль в проекте, а часть его профессиональных обязанностей в рамках автономной отладки.

Другой метод, нисколько не противоречащий априорному тестированию, – перекрестное тестирование, когда *разработчики* независимых компонентов проекта тестируют функциональность друг друга. Здесь можно говорить о *перекрестном совмещении ролей разработчиков* и *тестировщиков*, поскольку для выполнения перекрестного тестирования нужно различное представление об испытываемом модуле.

В таблице 2.1 приводятся краткие характеристики *совмещения ролей*, которые можно рассматривать как рекомендации для менеджмента. Не стоит жестко фиксировать распределение ролей и в одном проекте. Более того, распределение ролей можно рассматривать в качестве одного из инструментов, с помощью которого менеджер может руководить коллективом. Если этим инструментом пользоваться умело, то удается добиться наиболее эффективного разделения труда, соответствующего индивидуальным особенностям участников проекта.

Таблица 2.1 – Совместимость ролей

Роли	**Характеристика совмещения ролей**
Менеджер и архитектор	Желательно
Менеджер и руководитель команды	Противоречиво
Руководитель команды и архитектор	Возможно
Руководитель команды и проектировщик подсистемы	Нежелательно
Менеджер и разработчик	Не допускается
Для различных разработчиков	с ограничениями
Создание документации (все сотрудники)	Успешно распределяется
Специалист по интерфейсу и менеджер	Разумно
Эксперт предметной области и менеджер	Зачастую разумно
Специалист по интерфейсу и эксперт предметной области	Редко бывает эффективно
Эксперт предметной области и разработчик	Бывает полезно
Специалист по интерфейсу и разработчик	Часто полезно
Библиотекарь и один из разработчиков	Допустимо
Тестировщики и другие члены команды	Перекрестно
Эксперт предметной области, тестировщик	Оправдано

2.2 Ключевые роли коллектива разработчиков и задача определения кадровых ресурсов проекта

Руководство коллективом разработчиков – постоянная задача менеджера. Очень часто она выходит за пределы проекта, поскольку опытный менеджер, когда берется за проект, всегда имеет в виду команду, с которой ему будет комфортно работать, которая справится с заданием вполне гарантированно. Подобные команды не складываются сразу, а формируются в процессе выполнения проектов. В конечном счете, за руководство стоит браться только тогда, когда есть шанс в процессе выполнения работ получить такую команду.

В ходе априорного распределения ресурсов менеджер определяет кандидатуры на *ключевые роли коллектива разработчиков*. Само понятие ключевой роли указывает на то, что от сотрудников, которым такие роли назначены, в наибольшей степени зависит успех проекта. Какие роли в проекте являются ключевыми, во многом зависит от специфики разработки.

Намного лучше, когда лидерство сотрудников обнаруживается или заранее известно на ранней стадии проекта. Это позволяет сразу правильно выстроить административную структуру. Тем не менее, и здесь есть свои подводные камни. Не все *лидеры* готовы или хотят брать на себя управленческие обязательства. В этом случае лучше с самого начала назначить администратора, который освободит фактического *лидера* от перманентных административных проблем. Другая проблема – соперничество за лидерство в коллективе, что является потенциально рискованной для проекта ситуацией. Если такое случается, а еще лучше, когда соперничество только зарождается, сразу же разделить сферы ответственности конкурентов.

Есть еще одна причина, по которой ситуация с менеджером в качестве *лидера* может отрицательно сказываться на результативности выполнения проекта: в таком случае ему неизбежно придется вникать в детали производственного процесса, которые не соответствуют менеджерской функции распределения ресурсов, работ и контроля.

Наличие единственного *лидера* в группе, на которого может положиться менеджер проекта, – одна из главных предпосылок формирования продуктивно работающего коллектива. Но если

появляется *противодействующий лидер*, ситуация в проекте становится крайне неустойчивой. И менеджеру нужно всячески избегать ее: распознавать противодействие как можно раньше, быть терпимым и терпеливым при обсуждении вариантов решений даже тогда, когда кажется, что решение лежит на поверхности. Следует подчеркнуть, что в такой ситуации "хирургическое вмешательство" способно разрушить коллектив со всеми вытекающими отсюда последствиями. Борьба с *лидером*, особенно со скрытым неформальным *лидером*, к тому же заметная членам команды, также плохое средство. Она может быть действенной только в тех случаях, когда руководитель уверен в своей победе без административных воздействий. По сути, такая борьба должна рассматриваться в рамках мер по смене *лидера* команды. И наиболее трудной она оказывается тогда, когда *противодействующий лидер* выполняет одну из *ключевых ролей* в проекте.

Следующий перечень ключевых ролей *характеризует наиболее типичные ситуации для программных проектов:*

- архитектор проекта;
- проектировщики подсистем;
- руководители команд разработки подсистем;
- специалист по пользовательскому интерфейсу;
- эксперт предметной области.

Безусловно, *ключевой ролью* в проекте является лишь архитектор. В связи с этим уместно отметить, что провозглашаемый сторонниками экстремального программирования принцип программирования без архитектора может означать только одно: его роль явно или неявно распределяется между членами команды. При таком подходе говорят, что все разработчики команды занимают *ключевые роли* с самого начала проекта. Поэтому ясно, почему команды, действующие по схемам экстремального программирования, открываются для привлечения дополнительных кадров довольно редко. При необходимости выполнять работы, выходящие за рамки компетенции сотрудников, они чаще всего прибегают к субподряду.

- Во-первых, менеджер может заранее знать возможных кандидатов (хорошо себя зарекомендовавших сотрудников фирмы, с которыми менеджер уже имел дело либо знает по чужим работам). Это самый комфортный вариант для менеджера, который вполне может использовать личные, иногда лишь интуитивные

представления о сотрудниках для выбора кандидатов. Следует, однако, иметь в виду, что ни в коем случае нельзя подменять знание о человеке как о работнике сведениями личного характера (приятельские отношения, коммуникабельность и др.).

• Во-вторых, менеджер может подбирать кандидатов из числа сотрудников фирмы, с которыми он еще не имел дело. В этом случае ему не обойтись без изучения послужного списка сотрудника, *собеседований* и прочих способов получения сведений о человеке.

• Третий вариант, на который стоит рассчитывать, если возможности предыдущих вариантов исчерпываются, – это принять на работу нового сотрудника. Ситуация почти такая же, как в предыдущем случае, но несколько хуже: про сотрудника фирмы довольно просто получить объективную информацию, тогда как, нанимая нового человека, приходится довольствоваться сведениями из документов и результатами *собеседований*.

Решение *задачи подбора кадров* для проекта зависит от условий, в которых действует менеджер. Ситуации могут быть следующие:

1. У менеджера есть коллектив потенциальных исполнителей, готовый приступить к работе над проектом.

2. Менеджерский коллектив потенциальных исполнителей недостаточен: среди членов команды нет сотрудников, которые обладают нужной квалификацией.

3. В поле зрения менеджера есть независимые потенциальные исполнители, из которых можно сформировать коллектив для работы над проектом.

4. Менеджеру приходится прибегать к найму рабочей силы со стороны.

Таким образом, менеджмент проекта рассматривается как обеспечение поставок продукта, разработка которого требует выполнения определенного объема работ (область действия) с привлечением затрат, не выходящих за определенные пределы, укладывающаяся в заданные рамки времени и удовлетворяющая приемлемому уровню качества. Это широко известный "***треугольник менеджмента проектов***", представленный на рисунке 2.2. По разным причинам соблюдать баланс по всем параметрам чаще всего не удается, а потому менеджер вынужден выбирать в качестве первичной цели только один или два параметра, ставя остальные в зависимость от них. Это действие приводит к такой интерпретации рисунка: "треугольник хорошо-быстро-дешево – выбери два из них".

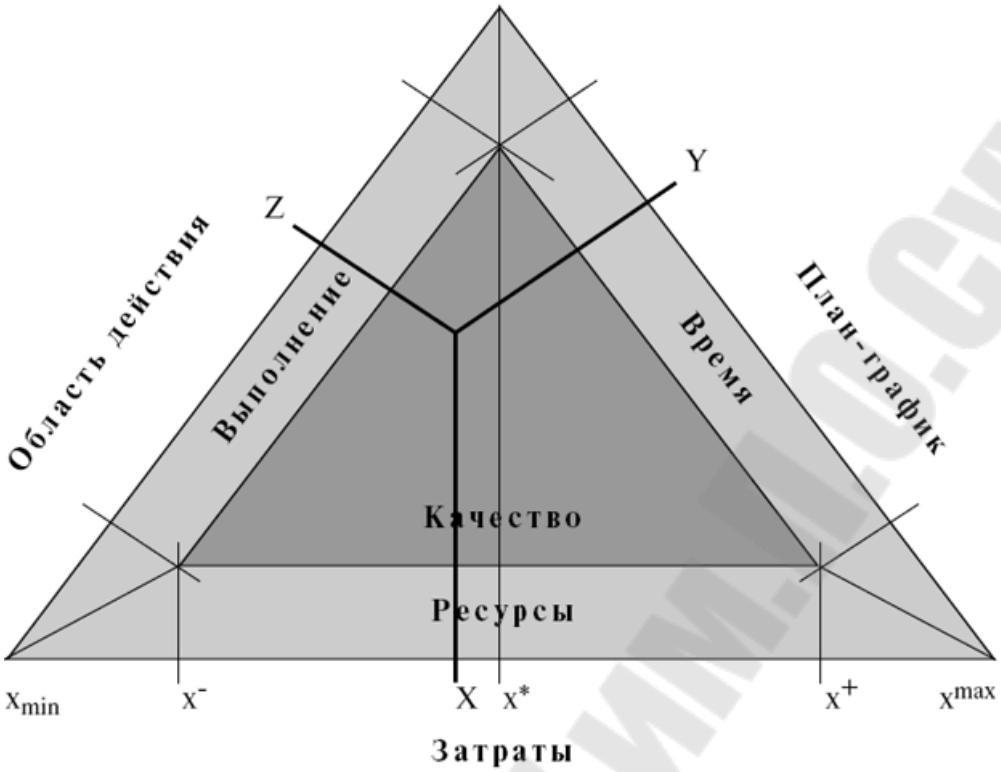


Рисунок 2.2 – Треугольник менеджмента проектов

В ходе управления должно быть обеспечено рациональное использование всех ресурсов, необходимых для создания и функционирования изделия. Процесс поиска оптимального соотношения между различными ресурсами при подготовке к реализации проекта называется *принципом совмещения методологий*.

Вопросы для самоконтроля:

1. В чем заключается идея экстремального программирования?
2. Назовите ролевые кластеры, структурирующие проектные функции разработчиков.
3. Назовите участников ролевой структуры проекта, предложенной компанией IBM.
4. Какие роли в коллективе разработчиков целесообразно совмещать?
5. Какие роли в коллективе разработчиков совмещать крайне нежелательно?
6. Какие типы тестирования программного обеспечения существуют, и с какой ролью целесообразно совмещать роль тестировщика?

3 Жизненный цикл программного изделия и традиционные модели

3.1 Понятие жизненного цикла программного изделия

Очевидно, что функции, выполняемые разработчиками проекта, в ходе его *развития* претерпевают изменения, как, впрочем, и сам проект. Сначала он существует в виде заявки на *разработку*, затем – как функциональные и технические требования, далее – как спецификации разрабатываемого изделия, набор программных модулей, скомпонованная из модулей система и т.д. Этот перечень можно рассматривать как один из примеров *модели жизненного цикла программного изделия*, т.е. представления эволюции *разработки* и последующего *использования* программной системы.

Жизненный цикл следует рассматривать как основу деятельности менеджера программного проекта: с ним связываются и цели проекта – окончательные и промежуточные, распределение и контроль расходования ресурсов, а также все другие аспекты управления *развитием проекта*. Прежде всего, эта привязка обусловлена разбиением *производства* любой программы на этапы, которые ассоциируются с определенными видами работ или функций, выполняемых разработчиками в тот или иной момент *развития проекта*. Этапы характеризуются направленностью выполняемых функций на достижение локальных (для этапа) целей проекта. Необходимость отслеживания целей приводит к понятию контрольных точек – моментов *разработки*, когда осуществляется подведение промежуточных итогов, осмысление достигнутого и ревизия сделанных ранее предположений.

Таким образом, в рамках обсуждения менеджмента программных проектов вопросы *жизненного цикла* должны рассматриваться как первостепенные. В этой и последующих лекциях они разбираются в той мере, которой достаточно для самостоятельного изучения конкретных подходов и методов, рекомендуемых для применения при *производстве* программных систем.

Программы не подвержены физическому износу, но в ходе их эксплуатации обнаруживаются ошибки (неисправности), требующие исправления.

Ошибки возникают также от изменения условий *использования* программы. Последнее является принципиальным свойством программного обеспечения, иначе оно теряет смысл. Поэтому правомерно говорить о *старении программ*, правда, не о физическом, а о "моральном". Необходимость внесения изменений в действующие программы (как из-за обнаруживаемых ошибок, так и по причине развития требований) приводит, по сути дела, к тому, что *разработка* программного обеспечения продолжается после передачи его пользователю и в течение всего времени жизни программ. Деятельность, связанная с решением довольно многочисленных задач такой продолжающейся *разработки*, получила название ***сопровождения*** программного обеспечения

Первоначально понятие *жизненного цикла* рассматривалось как *цикл разработки*. Однако понимание того, что стоимость программного обеспечения включает издержки в течение всего времени жизни системы, а не только затраты на *разработку* или исполнение программ, привело к естественной трансформации исходного понятия *цикла разработки*. ***Жизненный цикл*** – это проекция пользовательского понятия "время жизни" на понятие разработчика "технологический цикл (***цикл разработки***)". Комбинацией этих понятий объясняется происхождение самого термина "*жизненный цикл программного обеспечения*".

Существуют различные подходы к *моделированию жизненного цикла* программного изделия, отражающие те или иные аспекты *разработки* программ и связанной с ней деятельности. В рамках тематики настоящего курса следует использовать такие модели, которые в полной мере дают представление об управленческой деятельности. Кроме того, особое внимание уделяется тем моделям, которые хорошо согласуются с прогрессивными методами *разработки* программных систем и, прежде всего, рассчитаны на построение систем согласно методологии объектной ориентированности.

Традиционный подход к *разработке* программных систем предполагал, что технологичное *развитие проекта* возможно только тогда, когда предварительно собраны и проанализированы все требования к будущему изделию. В результате такого анализа, с одной стороны, выясняется истинное назначение системы, а с другой – появляется полная информация, необходимая для разбиения проекта на части, допускающие независимую *разработку*, –

декомпозиции проекта. После *декомпозиции* автономно реализуются выделенные части – модули или подсистемы (в зависимости от сложности требуемого программного продукта), причем в том же стиле, т.е. с предварительным полным анализом требований и, возможно, с дальнейшей *декомпозицией* частей. Затем осуществляется сборка – компоновка модулей в единую систему (подсистему – для выделенных частей). В результате продукт, поставляемый для *использования*, появляется в конце *разработки* всех частей и их компоновки. Если для проекта удается создать такие условия, что можно выделить для анализа все требования, то это гарантирует качество результатов. Однако на практике такая ситуация встречается крайне редко.

В описании традиционного подхода явно просматривается стремление следовать методологической стратегии, которую мы назвали определением этапов проекта и ***последовательным его развитием***. По существу, это механический перенос методики решения инженерных задач в промышленном *производстве* на область программирования. И как мы только что могли убедиться, условия материальной деятельности, в которых возможен и даже необходим предварительный сбор всех требований к проекту здесь выполнить не получается. Как следствие, стратегия *последовательного развития проектов,* по крайней мере, нуждается в коррективе.

При *объектно-ориентированном* подходе к *проектированию* провозглашается принцип ***возвратно-поступательного развития***, или итеративного наращивания системы, суть которого состоит в следующем. На каждой фазе проекта строятся работоспособные продукты, развиваемые в дальнейшем путем обогащения функциональности и интерфейса, а не в жестких рамках предварительного технического описания в целом, построенного в ходе специального этапа конструирования, предусматриваемого в традиционных схемах. Как следствие, фазы *развития проекта* (в традиционном понимании этого слова) при выполнении отдельной итерации оказываются незавершенными, они дополняются (наращиваются) на последующих итерациях.

Таблица 3.1 – Сравнение параметров схем разработки проектов

Традиционные схемы	Объектно-ориентированная схема
Полностью завершенные фазы проектирования и программирования	Итеративно наращиваемые возможности. Традиционные фазы распределяются по итерациям
Продукт в конце периода разработки	Рабочие продукты на каждой итерации
Техническое описание как итог конструирования	Рабочее описание, дополняемое на каждой итерации
Последовательная разработка	Возвратно-поступательная разработка
Модули действий, операций	Иерархии классов объектов
Структурная, пошаговая детализация	Наследование, переопределение, полиморфизм

3.2 Модели традиционного представления о жизненном цикле

Общепринятая модель

Понятие жизненного цикла связано с систематизацией работ в соответствии с производственным процессом. В ***общепринятой модели жизненного цикла*** ПО (рисунок 3.1) программные системы проходят в своем развитии две фазы: разработка и сопровождение.

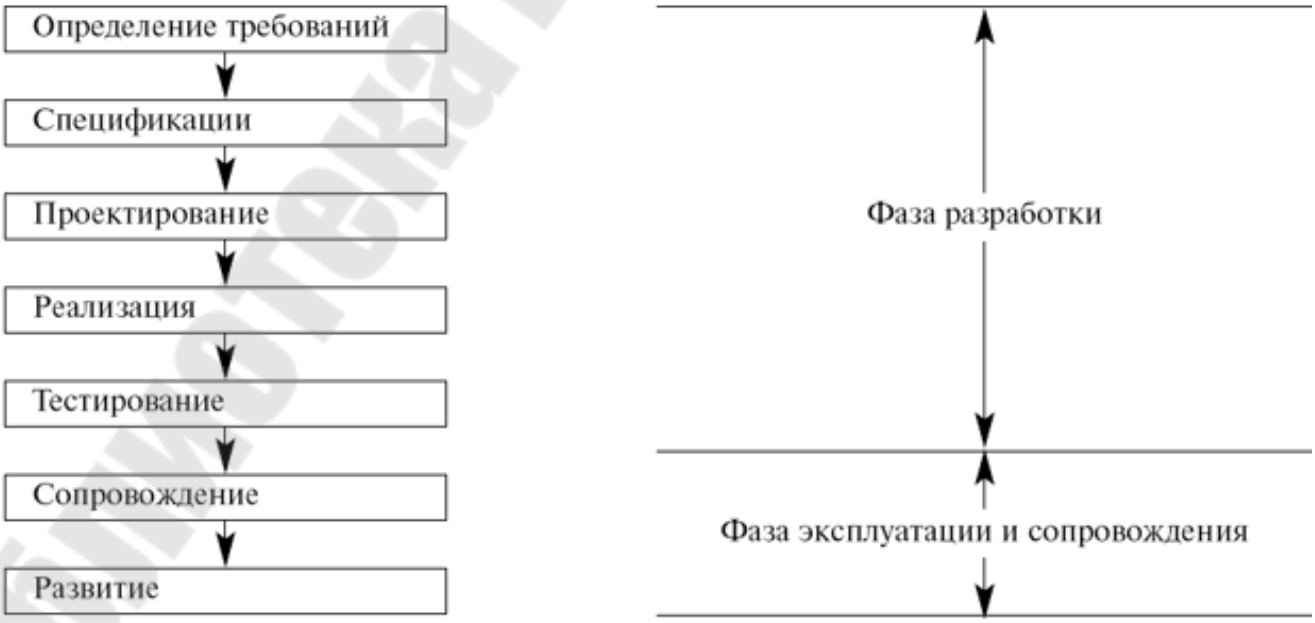


Рисунок 3.1 – Схема общепринятой модели жизненного цикла проекта

Разработка начинается с *идентификации потребности* в новом приложении, а заканчивается передачей продукта разработки в эксплуатацию.

Первым этапом *фазы разработки* является постановка задачи и *определение требований*. Определение требований включает описание общего контекста задачи, ожидаемых функций системы и ее ограничений. На данном этапе заказчик совместно с разработчиками принимает решение о создании системы. В процессе согласования требований со стороны разработчиков в обсуждении участвует менеджер проекта, а также, возможно, планировщик как действующее лицо, заинтересованное в соблюдении условий ведения проекта с точки зрения фирмы и осведомленное о ее ресурсном потенциале. Принципиально, что на данном этапе самого проекта еще нет и можно говорить только о предпроектных работах, в которых участвуют исполнители, занимающие указанные выше роли.

Разработка проектных решений, отвечающих на вопрос о том, как должна быть реализована система, чтобы она могла удовлетворять специфицированным требованиям, выполняется на этапе *проектирования*. Поскольку сложность системы в целом может быть очень высока, главной задачей этого этапа является последовательная декомпозиция системы до уровня очевидно реализуемых модулей или процедур. Наиболее активная роль на данном этапе – архитектор, для которого декомпозиция системы есть главная задача в проекте.

На следующем этапе *реализации*, или кодирования каждый из модулей, выявленных при декомпозиции, программируется на наиболее подходящем для данного приложения языке. С точки зрения автоматизации этот этап традиционно является наиболее развитым. Основные действующие лица этапа – руководитель команды и разработчики. Традиционно именно данный этап считали основой проекта в целом, что, как мы уже успели убедиться, не отражает современного взгляда на проект как на постоянно развивающийся артефакт. В обсуждаемой модели специально не выделяется этап сборки, который заключается в комплексации (интеграции) построенных и используемых модулей в систему. Считается, что это один из видов работ этапа *реализации*.

В *общепринятой модели фаза разработки* заканчивается этапом *тестирования* (автономного и комплексного) и передачей системы в эксплуатацию – следующие два этапа.

Фаза эксплуатации и *сопровождения* включает в себя всю деятельность по обеспечению нормального функционирования программных систем, в том числе фиксирование в скрытых во время исполнения программ ошибок, поиск их причин и исправление, повышение эксплуатационных характеристик системы, адаптацию системы к окружающей среде, а также, при необходимости, и более существенные работы по совершенствованию системы. Все это дает право говорить об эволюции системы. В связи с этим фаза эксплуатации и *сопровождения* разбивается на два этапа: собственно *сопровождение* и *развитие*. В ряде случаев на данную фазу приходится большая часть средств, расходуемых в процессе жизненного цикла программного обеспечения.

Классическая итерационная модель

Общепринятая модель жизненного цикла не является идеальной уже потому, что только очень простые задачи проходят все этапы без каких-либо *итераций* – возвратов на предыдущие шаги производственного процесса. При программировании, например, может обнаружиться, что *реализация* некоторой функции очень громоздка, неэффективна и вступает в противоречие с требуемой от системы производительностью. В этом случае необходимо перепроектирование, а может быть, и переделка *спецификаций*. При разработке больших нетрадиционных систем итеративность возникает регулярно на любом этапе жизненного цикла как из-за допущенных на предыдущих шагах ошибок и неточностей, так и из-за изменений внешних требований к условиям эксплуатации системы.

Классическая итерационная модель (рисунок 3.2) абсолютизирует возможность возвратов на предыдущие этапы. Однако это обстоятельство отражает существенный недостаток программных разработок, проводимых в традиционном стиле: стремление заранее предвидеть все ситуации использования системы и невозможность в подавляющем большинстве случаев достичь этого. Все подобные методологии программирования направлены лишь на то, чтобы минимизировать возвраты. Но суть от этого не меняется: при возврате всегда приходится повторять построение того, что уже считалось готовым.

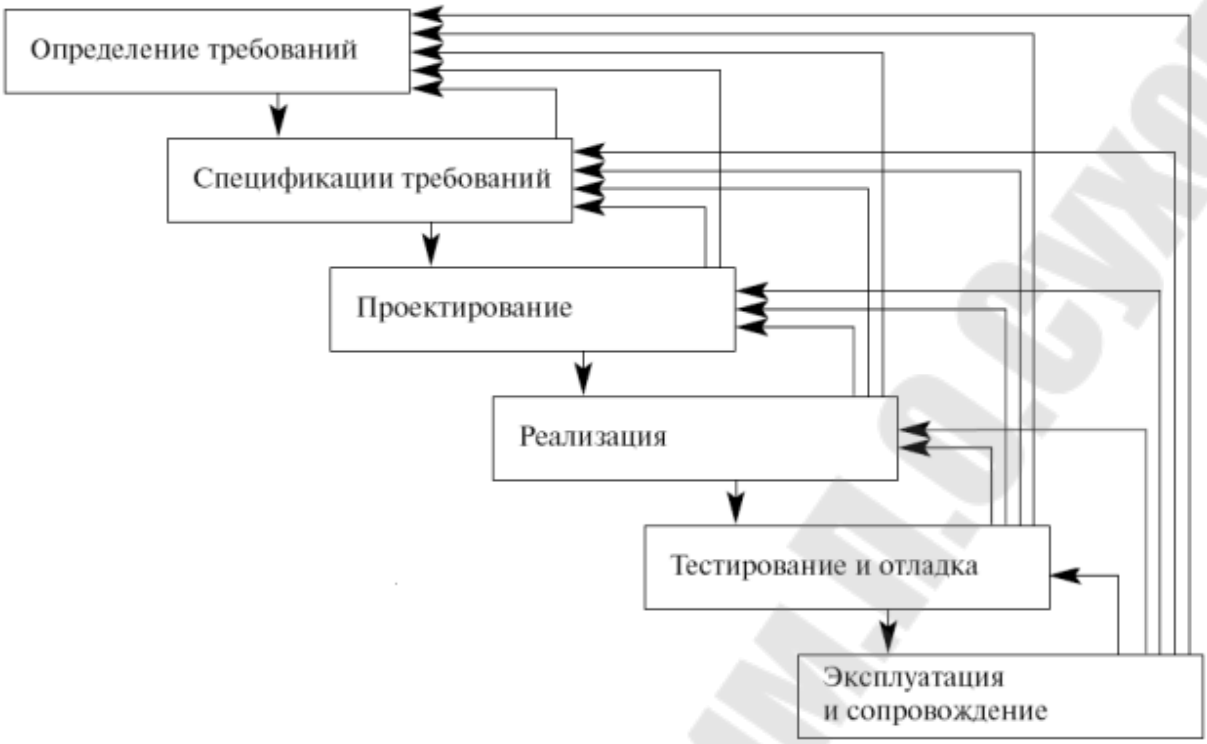


Рисунок 3.2 – Схема классической итерационной модели жизненного цикла проекта

Каскадная модель

Более строгой разновидностью классической итерационной модели является так называемая ***каскадная модель*** (рисунок 3.3), которую можно рассматривать в качестве показательного примера того, какими методами можно минимизировать возвраты.

Характерные черты каскадной модели:

- завершение каждого этапа (они почти те же, что и в классической модели) проверкой полученных результатов, с целью устранить как можно большее количество проблем, связанных с разработкой изделия;
- циклическое повторение пройденных этапов (как в классической модели).

Мотивация *каскадной модели* связана с так называемым *управлением качеством* программного обеспечения. В связи с ней уточняются понятия этапов, некоторые из них структурируются. Результат *проектирования* верифицируется, т.е. проверяется, обеспечивают ли принятая структура системы и реализационные механизмы выполнимость специфицированных функций. Реализация контролируется путем *тестирования* компонентов, а после интеграции компонентов в систему и комплексной отладки

проводится аттестация, т.е. проверка-фиксация фактически реализованных функций системы, описание ограничений *реализации.*

Рисунок 3.3 – Каскадная модель

В *каскадной модели* верификация и аттестация приписаны к разным этапам. Если рассматривать их как метод проверки каких бы то ни было проектных результатов, то нужно иметь в виду отличие этих родственных видов деятельности. Кратко их можно охарактеризовать следующим образом:

- верификация отвечает на вопрос, правильно ли создана программная система;
- аттестация отвечает на вопрос, правильно ли работает программная система.

Вопросы для самоконтроля:

1. Что такое жизненный цикл проекта?
2. В чем отличие последовательного и итеративного развития проекта?
3. Назовите модели традиционного представления о жизненном цикле проекта.
4. Назовите характерные черты каскадной модели.

4 Управление рисками и качеством при разработке проекта

4.1 Управление рисками

- ***причины риска*** – определенные события или обстоятельства в проекте или в его окружении, которые вызывают неопределенность;
- ***риски*** – неопределенные события или обстоятельства, возникновение которых может привести к воздействию на процесс достижения целей проекта:
  - к негативному воздействию, если в результате траектория проектной деятельности выходит из области допустимости;
  - нейтральному воздействию, если траектория не выходит из этой области;
  - к позитивному воздействию, когда новая траектория не только остается допустимой, но и по разным причинам оказывается предпочтительнее других траекторий, которые могли бы реализоваться, если бы *рисковая ситуация* не возникла;
- *последствия* – незапланированные отклонения траектории выполнения проекта, возникшие в результате того, что событие или обстоятельство, квалифицированное как риск, имели место.

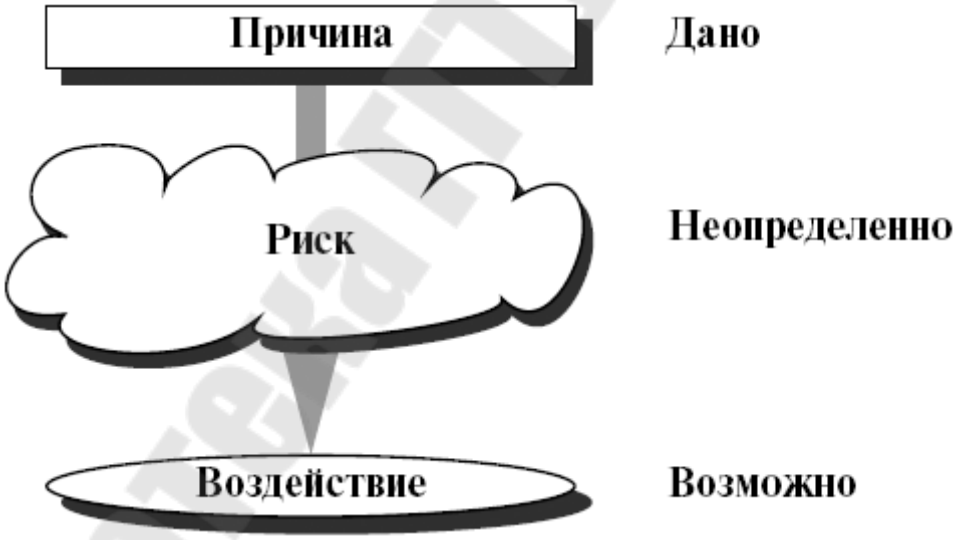


Рисунок 4.1 – Составляющие риска

Реальные риски – это те события, которые действительно характеризуются неопределенностью. Если причина детерминированно ведет к событию, то это не риск, а штатная ситуация, план преодоления которой к *управлению рисками* отношения не имеет. Следовательно, нужен анализ связей причин и

событий. В результате идентификации рисков формируется список идентифицированных реальных рисков.

Вторая стадия составления плана *управления рисками* при любой методологии – выставление приоритетов рискам. Иными словами, нужно определить сравнительную важность рисков для проекта. Устанавливается, насколько существенным может оказаться *воздействие каждого риска на проект*, и по этому признаку все идентифицированные риски упорядочиваются. После этого оцениваются две вероятности: для причины и возможного воздействия. Удобно расположить все риски в виде точек на плоскости, координаты которых отражают заданные вероятности. В таком случае можно зрительно отделить несущественные риски от существенных и далее работать только с последними. Нужно, однако, предостеречь, что выставление вероятностей – операция субъективная, а потому результирующее отсеивание требует проверки.

На третьей стадии продолжается работа с полученным списком. Устанавливается возможность *влияния на риски*, т.е. на причины рисков и на оказываемое ими *воздействие на проект*. Влияние может осуществляться на нескольких уровнях и для каждого из рисков должны быть определены влияния, планируемые для последовательного выполнения на тех уровнях, на которых это возможно:

1. *Исключение риска* (avoid). Некоторые рискованные ситуации можно исключить полностью. Например, чтобы увольнение работника с ключевой ролью не сильно сказалось на продолжении развития проекта, целесообразно с самого начала предложить для исполнения этой роли двух человек сравнимой квалификации. В начале проекта их дискуссии полезны для выработки объективных решений, а если один из них откажется от контракта, второй все еще сможет продолжать дело. Полезные дискуссии – эта та жертва, которая во многих случаях возможна для *исключения риска*. К сожалению, дублирование не может быть рекомендовано для *исключения* всех рискованных ситуаций.

2. *Переключение риска* (transfer). Это частный случай *исключения*, когда риск переносится из сферы влияния проекта на окружение. Например, если рыночная ценность создаваемого изделия кажется сомнительной, для его разработки комфортнее договориться с заказчиком об авансовых платежах, тем самым заставив его самого

решать задачу преодоления риска и освободив от этого разработчиков (возможно, за счет снижения оплаты проекта). К этому уровню относятся все варианты контрактных соглашений, но не только они. Оперируя рисками, всегда нужно стараться предусмотреть способы *переключения*.

3. *Уменьшение риска* (mitigate). Если риск не может быть исключен, можно постараться уменьшить вероятность его появления на практике (оперирование причинами). Продолжая пример с увольнением сотрудника, для снижения вероятности этого следует предугадать причины поступка и постараться создать для сотрудника более комфортные условия (повысить заработную плату, создать льготы и т.п.). Нужно, чтобы подобные решения делались не в ответ на заявление об увольнении, а заранее. Это сохранит определенную стабильность в коллективе, которая сама по себе является методом *уменьшения риска*.

4. *Предупреждение ущерба от риска* (accept). Когда не получается удовлетворительно уменьшить риск, следует подготовиться к встрече с ним так, чтобы минимизировать потери, т.е. снизить вероятность негативного воздействия и уменьшить само воздействие (оперирование последствиями). Если это удается, то продолжение проекта во многих случаях оказывается успешным. В примере с увольнением следует как можно скорее найти замену данному сотруднику. Естественно, время выполнения проекта увеличится (в частности, потому что нового человека придется вводить в курс дела), но работа все-таки будет продолжена. Это так, но при одном условии: на всех уровнях проектирования заложена возможность отделения результатов труда от разработчиков. Если результаты персонифицированы, то трудности подмены для некоторых ролей могут оказаться непреодолимыми.

4.2 Управление качеством проекта

Планирование мероприятий и стратегии *управления качеством* разрабатываемого программного изделия – традиционная часть подготовительных работ, выполняемых менеджером. Они связаны с определением принципов, которые помогают в решении вопросов *качества* разработки, непременно возникающих на всех стадиях развития проекта:

- утверждение целей проекта с точки зрения *качества* его выполнения, определение критериев успеха и достижения ожидаемых результатов;
- проверка и отслеживание *качества*;
- удаление дефектов;
- указание общих показателей *качества*.

План управления качеством – это перечень мероприятий, которые проводятся в контрольных точках жизненного цикла проекта для измерения и оценки определенных показателей, характеризующих достигаемые результаты. Обычно такой план включает:

- цели, достижение которых удовлетворяет заказчика проекта в целом (удобство использования, возможности и эксплуатационные характеристики и др.);
- цели, связанные *качеством* программирования (удобная для чтения структура, комментарии и их полнота, соответствие спецификациям, переиспользуемость и эффективность кода);
- использование инструментов (какие, для чего, как влияют на *качество*);
- модель исправления дефектов (организационная процедура поиска дефектов и их устранения);
- средства *управления качеством*:
  - соглашение об описании процессов в виде, удобном для изменения и исправления программ;
  - представление порядка отслеживания *качества*;
  - специфичные для проекта соглашения.

В приведенном ниже перечне указаны некоторые довольно типичные ошибки проектирования, относящиеся к *качеству*:

- Подмена автоматизации деятельности предоставлением средств. Эта очень распространенная ошибка разработчиков возникает в связи с тем, что при определении требований к программному продукту фиксируется, что нужно предоставить пользователю, но игнорируется вопрос о том, как данные средства будут включены в его деятельность. В результате выполнение типичных и частых для данной деятельности действий требует усилий со стороны пользователя, что снижает эффективность средства;
- Игнорирование средств окружения, которые можно использовать для автоматизируемой деятельности. В ряде случаев

средства, предоставляемые окружением программы, дублируются. Примеры такого рода: различные мини-редакторы, визуализаторы, копировщики. Иногда это оправданно, но чаще дублирующие средства реализуются только по той причине, что они нужны для данной деятельности. В результате повышается стоимость разработки, а пользователю приходится осваивать дополнительные средства. Как и в предыдущем случае, для исключения дублирования необходимо специальное исследование для определения возможности и целесообразности использования существующих средств (аспект общего плана проекта), а также проверяемые показатели переиспользования (аспект *плана управления качеством*);

- Пользовательский интерфейс, не соответствующий автоматизируемой деятельности: непривычный, эргономически неточный и т.д. Причина этого недостатка все та же: просчеты с исследованием автоматизируемой деятельности. Но, кроме того – игнорирование в общем плане проекта задачи специальной настройки интерфейса;
- Неадекватная реакция системы на ошибки пользователя. Правильная точка зрения на пользовательские ошибки – считать их не исключениями, а обычными явлениями, возникающими при работе программы. В этом случае можно добиться в диагностике ответа на три вопроса, возникающие у пользователя, когда система фиксирует ошибку:
  - **Что случилось?** В диагностическом сообщении причина ошибки и ее последствия должны разъясняться в пользовательских терминах.
  - **Что можно сделать?** Пользователю должна быть предоставлена возможность получить информацию о возможных действиях, на корректное выполнение которых он может рассчитывать.
  - **Как исправить положение?** Следует дать пользователю возможность получить сведения о пути исправления ошибки с уточнением, какие потери его ожидают.

Систему, диагностика которой удовлетворяет указанному критерию во всех ошибочных случаях, следует признать более качественной. К сожалению, и здесь трудно определить количественные показатели *качества*. Подобный критерий можно определить для предупреждающих сообщений;

• Неадекватная реакция на системные ошибки (ошибки разработчиков). Любая программная система не застрахована от ошибок. В качественной системе, заботящейся о комфортных условиях работы пользователя, предусматриваются рекомендации действий, когда обнаруживается ошибка разработчиков. Кроме того, пользователь должен быть информирован о процедуре внесения исправления. Понятно, что для обеспечения этого требуются специальные ресурсы.

Вопросы для самоконтроля:

1. Какие существуют способы уменьшения и исключения рисков?
2. Что включает в себя план управления качеством проекта?
3. Назовите типичные ошибки проектирования, относящиеся к качеству.

5 Этапы реализации веб-проекта

Процесс веб-разработки включает в себя следующие этапы:

- постановка задачи / разработка концепции проекта;
- проектирование;
- маркетинг;
- создание макетов дизайна;
- верстка макетов;
- программирование;
- тестирование, исправление багов;
- поддержка.

Постановка задачи / разработка концепции проекта

Первый этап – наиболее ответственный, т.к. от его успешного выполнения зависит дальнейшая судьба веб-проекта. На этом этапе собирается вся необходимая информация, которая только может потребоваться для его успешной реализации. Этот этап строится на непосредственной коммуникации заказчика и исполнителя.

У заказчика уточняются цели, пожелания и мысли относительно его проекта, поэтому заказчик должен как можно четче, яснее и информативнее ответить на все задаваемые вопросы – это в его интересах.

Обычно уточняются следующие аспекты.

Задача. Какая задача поставлена перед сайтом? Вы хотите продавать какую-либо продукцию, услуги или предоставлять информацию по какой-либо теме?

Цели. Чего Вы хотите добиться, благодаря новому сайту? 2 наиболее популярные цели: распространение информации и получение прибыли.

Целевая аудитория. Кто относится к Вашей целевой аудитории? Постарайтесь как можно точнее обрисовать портрет представителя Вашей целевой аудитории.

Контент. За какой информацией будут обращаться представители Вашей целевой аудитории к Вашему сайту? Может быть, им будет нужна практическая информация или информация об интересующих их товарах или услугах?

Детальный анализ данных, построение логической диаграммы входящих-выходящих данных.

Утверждение платформы, используемых языков программирования.

Предварительное планирование трудовых ресурсов (обсуждается наличие и квалификация специалистов), также определяется наличие специалистов по поддержке проекта после сдачи его заказчику.

Общие рекомендации по срокам исполнения проекта.

Определение стороны, ответственной за публикацию и продвижение проекта в сети. В случае, если это входит в задачи разработчиков, разрабатывается заранее приблизительный план маркетинговой кампании.

Проектирование

На этом этапе используем информацию, полученную на предыдущем этапе, для проектирования будущего сайта. Сейчас мы должны определить все необходимые разделы и подразделы сайта, а также их содержимое. Необходимо помнить о том, что сайт должен быть юзабельным, а его информационная архитектура понятной и прозрачной.

Особое внимание при проектировании следует уделить таким интерактивным элементам как формы, корзина, flash и т.д.

На этом этапе обычно происходит следующее:

- структура данных, определение связей между таблицами базы данных, структуры передачи данных из внешних источников;

- определение уровня автоматизации обработки данных, разработка структуры управления данными;
- проводится спецификация форм и порядок их появления;
- структура пользовательского интерфейса: пункты меню и элементы навигации, необходимые уровни вложенности;
- разработка эскизов дизайна проекта (количество вариантов утверждается заранее).

Маркетинг

Маркетинг – в нашем случае, это исследование рынка, компании заказчика и конкурентной среды. Не все заказчики до конца понимают, зачем нужен данный этап и часто полагают, что это пустая трата времени и денег. Конечно, не для всех проектов маркетинг обязателен. В каких случаях оно действительно необходимо, а в каких – факультативно? Чтобы ответить на этот вопрос, нужно для начала оценить масштабы самого проекта и его прямые цели.

Маркетинг необходим:

- при продвижении/продаже продукта;
- при привлечении новых потребителей;
- для решения задач брендинга;
- для любой другой коммерчески значимой деятельности в Интернете.

Характерные задачи на этапе:

- анализ источников информации;
- описание целевой аудитории;
- анализ спроса;
- определение структуры рынка;
- семантическое ядро;
- определение схемы сайта;
- составление рекомендаций.

Создание макетов дизайна

Дизайн по праву считается одним из наиболее важных и ответственных этапов работы по созданию сайта, т. к. не только привлекает внимание потенциальных заказчиков компании, но и реализует имидж фирмы в среде Интернет. На этом этапе разрабатываются концепции дизайна, графические элементы, которые призваны сделать сайт непохожим на другие. К дизайну предъявляются такие требования как оригинальность, технологичность и функциональность. Дизайн сайта должен

подчиняться одной цели, единой концепции. Он должен адекватно отражать характер деятельности заказчика, органично сочетаться с информационным наполнением, структурой сайта и принципами организации информации.

Наиболее важным фактором, на который необходимо опираться при создании дизайна – портрет представителя целевой аудитории сайта. Например, одинаково ли должны выглядеть сайты, целевая аудитория которых в первом случае молодежь, во втором – бизнесмены? Конечно, нет.

Дизайнер должен создать несколько различных макетов дизайна и предоставить их на суд заказчику. Готовый макет должен быть в виде простой jpg-картинки. После согласования одного из макетов, начинается его доведение до ума (если требуется).

Верстка макетов

Верстка представляет собой процесс интеграции текстового содержания, графики и программных компонентов в единое целое, то есть придание страницам окончательного вида. В процессе верстки страницы приобретают вид, в котором они предстанут перед конечным пользователем (за исключением информационного наполнения).

Именно на этом этапе начинается непосредственная материализация будущего веб-сайта. К этому моменту макет дизайна сайта уже утвержден. На этом этапе верстаются (по веб-стандартам XHTML+CSS) макеты главной и необходимых внутренних страниц, а также пишется система управления контентом, отвечающая заданным в техническом задании требованиям.

В это время происходит дополнительный контроль качества выполняемой работы, производится оптимизация web-страниц под особенности конкретных браузеров, используемых посетителями сайтов для навигации по Интернету. Учитываются особенности представления страниц при различных настройках глубины цвета и экранных разрешений.

Программирование

На данном этапе происходит разработка и подключение программных компонентов сайта, призванных обеспечить посетителей необходимыми функциональными возможностями. Этот этап является наиболее сложным по реализации в технологическом плане. Большинство решений, разрабатываемых на данном этапе, основываются на технологиях работы с базами данных и на

построении динамически генерируемых страниц сайта на основе информации, содержащейся в базе данных сайта.

Синхронизация

Синхронизация – это соединение сверстанных и черновых шаблонов сайта. Этот этап начинается только после завершения верстки и программирования. Если макеты и техническое задание были согласованы и утверждены заказчиком, то проблем на этом этапе не возникнет. Верстальщик должен получить от программиста хорошо документированные гибкие черновые шаблоны. Рекомендуется пользоваться технологиями xml/xslt.

На данном этапе будет задействован следующий персонал: верстальщик, программист, менеджер проекта.

Тестирование, исправление багов

На этом этапе проводится детальное тестирование сайта, тестирование на идентичное отображение сайта во всех распространенных веб-браузерах, работоспособность всех форм, скриптов, flash-роликов и т.д.

В случае нахождения каких-либо ошибок, недочетов и прочих багов, они в немедленном порядке исправляются.

Как только сайт был тщательно и неоднократно проверен, его можно загружать на клиентский FTP, а также переходить к заключительному этапу.

Публикация

Публикация – это обеспечение хостинга интернет-сайта и «привязка» сайта к предварительно зарегистрированному доменному имени. Обычно, содержимое сайта загружается на интернет-сервер через протокол FTP. В результате, интернет-сайт становится доступным всем пользователям сети Интернет.

Доменное имя – это адрес сайта в сети Интернет. Учитывая специфику деятельности и название компании, обычно составляется список наиболее подходящих доменных имен, и также проверяется их «занятость».

Хостинг – это размещение интернет-сайта на подключенном к сети Интернет сервере с соответствующим набором программного обеспечения, необходимого для корректной работы интернет-сайта. Заказчик сайта зачастую имеет на примете хостинг и после создания сайта просит, чтобы сайт разместили туда. Хотя бывает, что хостинг подыскивает фирма-разработчик веб-сайта.

Наполнение контентом

Контент – это содержимое сайта, та информация, которая может быть полезной его посетителям. Наполнение контента – это внесение графической и текстовой информации в шаблонные web-страницы сайта в рамках, предусмотренных его дизайном.

Наполнение сайта возможно через административный интерфейс (систему управления сайтом при ее наличии) либо путем заполнения статичных страниц после верстки сайта.

Перед тем, как наполнять сайт статьями, необходимо определить перечень ключевых слов по выбранной тематике. Необходимо разделить выбранные ключевые слова на тематические группы, создать на сайте разделы под каждую из таких групп. Статьи, которые будут публиковаться, должны содержать по возможности как можно большее количество определенных выше ключевых слов.

Очень желательно, чтобы контент на сайте был уникальным, то есть таким, которого нет ни на одном другом сайте. В этом случае этот сайт будет гораздо выше ранжироваться поисковыми системами.

Способов добычи уникального контента довольно много. К самым распространенным относятся:

- копирайтинг (самостоятельное написание статей);
- рерайтинг (изменение чужих статей до неузнаваемости с сохранением смысла);
- переводы иностранных статей;
- если этим заниматься нет возможности – можно воспользоваться услугами субподрядчиков или специальных сервисов.

Поддержка

Поддержка сайта – широкое понятие, включающее в себя как поддержание работоспособности сайта, так и его последующее развитие.

Еще на этапе проектирования необходимо задаться вопросом, каким образом будут добавляться новые разделы и материалы, что будет происходить со старыми. Возможно, потребуется создание архива новостей, куда будут попадать новости, потерявшие свою актуальность.

Еще более важной является постоянная актуализация информации на сайте (частота обновления, ответственный за наполнение сотрудник). Важно не забывать о посетителях сайта, поэтому им должно предоставляться больше новой информации.

Продвижение (раскрутка)

Готовый, наполненный информацией сайт еще не гарантирует вам приток новых клиентов. Главная причина этого – пользователи Интернет могут не встретить ваш сайт в сети. По статистике большинство пользователей находят информацию и приходят на сайты через поисковые системы. Именно поэтому популярность ресурса зачастую зависит от его позиции по результатам поиска. Чтобы ваш сайт занимал лидирующие позиции в поисковых системах необходимо провести ряд мероприятий по его продвижению в сети.

Продвижение – это своего рода «рекламная кампания» по узнаванию сайта и повышения его посещаемости. Сюда входит:

- регистрация сайта в поисковых системах;
- обмен ссылками;
- баннерная реклама и т.д.

Вопросы для самоконтроля:

1. Назовите основные этапы реализации веб-проектов.
2. Что такое контент сайта? Какие способы получения уникального контента существуют?
3. Что такое хостинг?
4. Что такое поддержка и продвижение сайта?

6 Основные особенности управления веб-проектом

6.1 Как эффективно организовать бизнес-проект?

Это этап, которым многие web-дизайнеры пренебрегают или не придают ему должного значения. Своевременная и точная разработка сайта и управление им важны не только для клиента или коэффициента окупаемости инвестиций компании. Грамотное планирование и управление жизненно необходимы для обеспечения стабильного долгосрочного функционирования сайта и получения желаемых результатов. Возможные проблемы можно свести к трем основным:

- web-команда не имеет четко определенной структуры;
- управление web-проектам не имеет официально формализованных стандартов;
- различные члены одной команды могут расходиться во мнениях.

На заре развития сети Интернет казалось, что один web-дизайнер может выполнять большинство задач при создании сайта. В

то время не было необходимости в интеграции сложных баз данных и самого сайта или электронной коммерции. На самом деле, большинство технологий, которые используются для выполнения этих сложных задач, начало появляться только с 1995 года. До 1995 года, большинство web-сайтов были «маленькими», поэтому одному человеку было достаточно легко поддерживать такой сайт. Но когда сеть Интернет дошла до потребителей, произошли значительные изменения. Сайтам потребовалась гораздо большая защищенность, эффективность работы, стало необходимо более частое и регулярное обновление и управление. Потребовалось больше людей, тогда и стали возникать web-команды.

Экономические потрясения, произошедшие в последние годы, привели к тому, что многие дизайнерские фирмы закрылись полностью или значительно сократили штат. В результате в различных компаниях, правительственных агентствах и образовательных учреждениях web-проекты стали осуществляться совершенно по-разному. Поскольку web-дизайн – понятие очень широкое, какой-то универсальной модели управления не существует, и, скорее всего, никогда и не будет. Для каждого отдельного проекта надо подбирать свои способы управления.

На стадии коммерческого предложения происходит концептуализация и определение бюджета проекта. На данном этапе необходимо выполнить следующие задачи: собрать исходные требования, составить примерную структуру сайта, оценить сроки выполнения работ, его стоимость и, наконец, сформировать конечное коммерческое предложение. Коммерческое предложение должно быть утверждено обеими сторонами. После этого можно приступать к составлению и подписанию договора между заказчиком и исполнителем проекта. Заказчик вносит исполнителю предоплату в оговоренном размере. На данном этапе будет задействован следующий персонал: менеджер по продажам, менеджер проектов, системный аналитик.

6.2 Специфика менеджмента веб-проектов

Напомним, что веб-проект, к которому относится всякий вебмедиа-ресурс, определяется следующими компонентами:

1) концепцией;
2) контентом (содержание);
3) трафиком (целевая аудитория);

4) дизайном.

Ключевым понятием веб-проектирования является *концепция*, которая включает в себя следующие компоненты:

• существующее или вновь организуемое *издание* (организация или инициативная группа лиц), которое представляет данный веб-проект;

• *идеологию* проекта, т.е. позицию (экологическаю нишу) в медийном и информационно-коммуникативном пространстве;

• *стратегию развития*, что подразумевает не только разработку самого проекта, но и планирование действий по отношению к конкурентам;

• *конечные цели*, достижению которых служит данный вебмедиа-ресурс.

Контент – это все содержание конкретного сайта, которое воспринимается рядовым посетителем в качестве такового. Контент – это не только тексты, но и все информативное наполнение сайта: картинки, фотографии, программы, скрипты, музыка, визуализированная структура сайта. Воспользовавшись метафорой, заимствованной из геш-тальтпсихологии, можно сказать, что контентом является все то, что должно восприниматься посетителем как фигура, а оформлением является все то, что воспринимается посетителем как фон. Легко видеть, что разделение фигура/фон является изменяемым параметром и подается в какой-то степени управлению с помощью дизайна.

Трафик измеряется в посетителях и посещениях (хостах и хитах). Собственно ради трафика и создаются веб-проекты. При управлении трафиком могут быть задействованы две разные стратегии. Первая – трафик максимизируется всеми доступными средствами, что приводит к созданию обширной, но разношерстной и разнородной аудитории, в которой (за счет массового охвата) присутствует и целевая аудитория. Вторая – осуществляется работа только на целевую аудиторию, посредством выхода на аудитории близких по тематике проектов и корпорирования с ними. Вторая стратегия не может дать быстрые результаты и требует кропотливой, методичной раскрутки, зато обеспечивает необходимый качественный состав целевой аудитории, структурируемой веб-проектом.

Дизайн предполагает не просто оформление сайта, а говорящее оформление сайта. Идеальный дизайн тот, который делает все эле-

менты оформления страницы фигурой, тогда фоном становятся другие страницы. В любом случае дизайн должен обеспечивать юзабилити (т.е. адекватность формы и содержания сайта, обложки книги и ее текста). Нельзя не упомянуть о том, что дизайн – это еще и своего рода демонстрация креативности проекта, его идеологического и культурного пространства, в котором позиционируется медиаресурс.

В последнее время дизайн, как модная одежда, выступает в роли индикатора стоимости проекта. О чем свидетельствует, например, логотип продвинутой дизайн-студии, размещаемый на первой странице проекта (например, студии Артемия Лебедева).

К обычному списку менеджерских задач веб-специфика добавит следующие:

• разработка бизнес-планов дочерних интернет-проектов, экспертиза и минимизация возможных рисков;

• разработка, позиционирование, продвижение, поддержка веб-компонентов редакторско-издательской деятельности;

• управление работами с аудиторией сайта, стимулирование увеличения глубины интереса и регулярности посещений, вовлечение пользователей в жизнедеятельность ресурса;

• выработка стратегии по получению максимальной прибыли с проекта, разработка форм конкурентоспособных предложений для продажи рекламы, спонсорских пакетов, прайс-листов и т.д.;

• управление рекламными кампаниями, аудит аудитории сайта и всех форм рекламного воздействия, реализованного на нем;

• анализ текущей активности в сети –экспертиза существующего сайта, проводимых маркетинговых мероприятий в сети;

• организация финансовых операций с рекламодателями, помощь по юридическим вопросам и патентованию;

• подготовка проекта и соответствующей документации для переговоров с инвесторами, текущая оценка капитализации (стоимости) и ликвидности (возможности продать) проекта, поиск и привлечение инвестиций.

Ясно, что для решения всего комплекса задач менеджмента сетевых СМИ необходим профессионализм, соединяющий в себе компетентность журналиста, веб-мастера, менеджера, специалиста PR, т.е. транспрофессионализм, который необходимо вырабатывать в себе как лидеру проекта, так и всем сотрудникам. Это главное условие успешности вебмедиа-проектов и, естественно,

еще один аргумент в пользу серьезного отношения к веб-проектам руководства традиционных медиа, которые по каким-то причинам решили создать веб-версии своих изданий.

6.3 Устранение разногласий при решении проблем

Тема, которая широко обсуждается в управлении web-проектами – это несоответствие взглядов у разных членов команды. Всем нам когда-либо приходилось принимать участие в конфликтах вокруг таких несоответствий.

Программист обычно мыслит абстрактно, линейными кусочками информации, а дизайнер может фокусироваться только на визуальной и творческой части проекта. В отделе маркетинга свои заботы, также как и в финансовом отделе. При работе в web-команде вам приходится сталкиваться не только с принципиально различными точками зрения, но и с расхождениями в понятийном аппарате и, соответственно, языке при выражении мыслей участников.

Несмотря на то что, по идее, все, кто работает в сети, должны обладать примерно одинаковыми стандартами общения, в реальности большинство людей здраво эгоистично. Это указывает на отсутствие образовательных и профессиональных стандартов в данной области.

В результате эффективное общение между людьми, работающими в данной области, становится проблематичным. При работе над проектами в команде этот недостаток может вылиться в настоящие конфликты.

Хороший менеджер проекта сможет решить эти проблемы, если правильно определит роли, обязанности и задачи, организует проект с учетом индивидуальных черт людей, работающих в команде, а также направит команду на работу в такой связке, которая ведет к общей, четкой цели. У менеджера проекта нелегкая работа. Ему приходится выполнять следующие сложные задачи:

- организация и определение ролей для членов команды;
- определение потребностей посетителей, компании и клиента;
- поиск компромисса между этими часто расходящимися потребностями;
- создание общего плана работы над проектом;
- контроль над тем, чтобы работа шла по плану, а проблемы эффективно решались;
- определение бюджета проекта и контроль над тем, чтобы не выйти за его рамки;

- обеспечение взаимосвязи между всеми сторонами, задействованными в проекте;
- сохранение мирного сотрудничества.

У менеджеров проекта должны быть определенные знания по каждой теме, которой коснется проект. Значит ли это, что менеджер должен знать, как установить и поддерживать web-сервер? Необязательно, но основные навыки и знание употребляемого профессионального арго должны быть частью его базовых знаний. Что касается навыков, для менеджера проекта самым главным должен быть навык эффективного общения.

6.4 Управление бюджетом

Независимо от объема проекта и количества людей, задействованных в его реализации, бюджет будет играть огромную роль в порядке работы над проектом. Эффективное управление бюджетом проекта подразумевает под собой следующее:

- определение конкретного объема проекта;
- составление бюджета для людских ресурсов в соответствии с реальной стоимостью работы;
- покупка оборудования, программного обеспечения и прочего попадает в бюджет проекта;
- ограничение затрат на непредвиденные расходы.

Безусловно, «время – деньги», поэтому важным аспектом управления бюджетом является эффективное использование времени.

6.5 Определение целей

Если тема кажется достаточно простой, позвольте уверить что, несмотря на простоту, определением целей часто пренебрегают, либо оно остается проходным пунктом процесса планирования. В этом основная причина возникновения проблем при разработке проекта.

Цели необходимо определить прежде, чем приступить к работе над проектом. Цели можно поделить на группы:

- цели клиента – это цели, которых хочет добиться клиент;
- потребности посетителей – пожалуй, ими пренебрегают чаще всего (надо помнить о посетителях на всех этапах проекта);
- цели сайта – то, для чего создается сайт.

Как ни странно, многие даже не представляют, с какой целью они создают web-сайты. Отсутствие четкого понимания

предназначения сайта может повлечь за собой самые разнообразные проблемы.

Несмотря на то что многие программные пакеты для планирования могут помочь вам с определением некоторых целей, ничто не сравнится с бумагой и ручкой. Сядьте и составьте максимально детализированный список для каждой цели, задачи и способа ее решения.

6.6 Определение ответственного лица

В контексте web-проекта, ответственное лицо – это человек, который принимает окончательные решения. Одной из самых больших проблем в современном web-дизайне является выяснение, кто действительно обладает правом решения в данной ситуации.

Главный секрет успешного управления web-проектом – точно определить, кто же в организации обладает властью с точки зрения принятия конечных решений. Обычно этой работой должен заниматься менеджер проекта, хотя любой человек, осуществляющий административную поддержку, может помочь в данной ситуации.

Проанализировав ситуацию, вы можете выяснить, что заинтересованных лиц несколько.

Несколько советов по определению ответственных лиц:

- всегда обращайте внимание на то, за кем (в фирме-заказчике) остается последнее слово. Этот человек должен быть наверху вашей иерархии, и к нему вам следует идти при возникновении каких-то проблем в самом конце;
- определите вторичных ответственных лиц. Это люди, которые будут принимать решения по отдельным частям проекта, например менеджер по маркетингу, IT-менеджер или арт-директор. Будет полезно, если вы сделаете какие-то пометки относительно их взаимоотношений с главным заинтересованным лицом: менеджер проекта может воспользоваться хорошими отношениями между вторичным и главным ответственными лицами при возникновении споров во время работы над проектом;
- постарайтесь сделать так, чтобы за главным ответственным лицом (либо за тем, кто был для этой функции им назначен) непременно было последнее слово в любом споре;
- попробуйте нарисовать иерархию фирмы-заказчика, чтобы ее командная цепочка была понятнее для вас, исполнителя.

Определение ключевых людей, принимающих решения,

является очень важным шагом к успеху, но также важно правильное взаимодействие с каждым из них. Если роли четко определены, этот процесс становится намного легче. На раннем этапе проекта попробуйте назначить посредника, чья задача будет состоять в смягчении конфликтов по мере их появления. Задачи этого человека будут отличаться от задач менеджера проекта; он будет разрешать споры между менеджером проекта со стороны исполнителя и ответственными лицами со стороны заказчика.

Если внутри вашей организации существуют значительные проблемы и споры нельзя уладить иными способами, вмешательство непредвзято настроенного человека может быть просто необходимо.

6.7 Определение потребностей рынка

Если вы уже определили идею проекта, основные требования клиента, цели сайта и посетителей, настало время осознать, поддержит ли рынок сайт, который вы хотите создать.

В задачи определения потребностей рынка входит следующее:

- понимание объема рынка для продукта или услуги, которую вы будете представлять данным сайтом;
- подготовка к управлению любыми экономическими и другими проблемными факторами внутри данного рынка;
- знание лидеров данного рынка и исследование их успехов и ошибок;
- определение потенциальных конкурентов на долгий срок;
- определение потенциальных партнеров для долгосрочных, взаимовыгодных отношений.

На данном этапе менеджеру проекта будет очень полезно сесть и произвести серьезное исследование, чтобы ответить на следующие вопросы:

Кто лидирует на рынке?

Эффективно или нет они используют свои сайты?

На какую аудиторию они нацелены и отличается ли эта аудитория от той, что вы определили для своего проекта?

Можно ли измерить степень удовлетворения потребностей пользователей?

Конечно, настоящая работа начнется, когда у вас уже будет эта информация. При изучении работы лидеров рынка, экономического, демографического уровней и уровня потребностей потребителя вам будет гораздо легче эффективно подстроить свой проект под данный рынок.

6.8 Определение состава команды

Следующий шаг – определение состава команды. Вы должны точно знать, кто входит в нее и что они могут делать. Даже если вы работаете один, знание вашей собственной роли и тех аспектов, в которых вы полностью уверены, поможет вам определить, в каких разделах проекта вам понадобится посторонняя помощь.

Когда определены игроки команды, намного проще распределить их обязанности. Когда вы будете оценивать членов команды, возможно, вам захочется включить в управление проектом и их особые увлечения и интересы.

Например, системный администратор может быть отличным художником, способным предложить несколько идей. В принципе, команды работают по принципу смешения, хотя некоторые знатоки считают, что каждый человек в команде должен быть ответственен только за свою отдельную область. Подобное профилирование призвано помочь разделить людей в группе на категории и таким образом упростить работу менеджера с людьми, обладающими разными точками зрения и использующими разные профессиональные арго. Однако на практике такое профилирование может сократить возможности людей до их односторонних навыков, не дав им возможности использовать все свои многочисленные таланты на пользу проекта.

Все хорошее – всегда маленькое и компактное, идеальная веб-команда должна состоять примерно из пти человек.

Прежде всего, команде нужен человек, который будет приносить ей работу. Этот человек должен быть несомненно хорошим продавцом, но не назойливым, как те "толкачи", что работают в сетевом бизнесе. Человек должен быть компетентным и харизматичным, способным одинаково хорошо держаться перед советом директоров и на бизнес-вечеринке. Так как ему придется в основном иметь дело с деловыми людьми, лучше всего будет, если это будет человек с опытом работы в отделе продаж. Однако он несомненно должен хорошо разбираться в Вебе.

Далее вам необходим человек, который будет управлять компанией. Этот человек будет следить за выполнением проектов, за тем, что все делается во время и в рамках бюджета. У него должны быть отличные организационные способности, и сам он должен быть очень дисциплинированным. Он будет отвечать за работу офиса, выплату зарплаты, ведение бухгалтерии.

Далее, вам нужен человек, которого иногда называют продюсером, информационным архитектором или пользовательским

дизайнером. Это тот, кто будет работать с клиентом и задавать направление развития проекта. Этот человек должен вырабатывать стратегию проекта, создавать документацию и вести проект к успешному завершению. Для этого у него должны быть технические, творческие и управленческие способности, а берутся такие люди из числа опытных дизайнеров или продюсеров.

В большинстве веб-проектов непосредственно с дизайном связано лишь 10% работы. Канули в лету те дни, когда в команде работал отдельный визуальный дизайнер. Вместо этого в современной веб-команде требуется "гибридный" дизайнер, который не брезгует Фотошопом и умеет написать CSS-код. Этот человек должен разбираться как в визуальном дизайне, так и в веб-стандартах, владеть понятиями "юзабилити" (usability), "универсальный дизайн" (universal design) и "дизайн для всех" (accessibility).

Ну и наконец, вам нужен талантливый программист. Все больше и больше проектов становятся своеобразными веб-приложениями, поэтому программирование занимает все большую и большую долю работы. Однако вместо того, чтобы брать в команду “элитного” программиста, вам нужен такой человек, который не в последнюю очередь умеет общаться с людьми так же хорошо, как и писать код.

В маленькой команде, чтобы она могла работать результативно, требуется небольшое "перекрытие" способностей. Каждый член команды должен понимать, что делают другие люди, и какие проблемы они решают. В основном каждый член команды должен овладеть навыками продюсера, чья задача, как мы помним, – вести проект. Каждый член команды должен уметь общаться с клиентом и не отказываться от разговоров с ним, не избегать собраний и быть готовым работать в особом режиме.

Итак, примерные названия должностей в малой веб-команде будут следующие:

- Sales and Marketing Manager (менеджер по маркетингу и продажам);
- Project Manager and Office Manager (менеджер проектов и директор);
- User Experience Designer (пользовательский дизайнер);
- Front End Designer/Developer (разработчик пользовательского интерфейса);
- Back End Developer (программист).

6.9 Организация работы над проектом

Управление проектами старо как мир, тем не менее, это не значит, что кому-то удалось изобрести совершенный рецепт организации рабочего процесса. Сначала нужно собрать массу информации, в том числе определить специфические цели. Однако в этом «совете» главное – понимание последовательности действий. Несмотря на то что существует много моделей рабочего процесса проекта, самое главное – правильно ими воспользоваться для достижения необходимых целей.

Задачи рабочего процесса проекта:

- Предпроектная часть. Компания и клиент утверждают проект, назначается его менеджер, команда, определяются ответственные лица, осмысляются потребности рынка и бюджет;
- Определение задач. Команда основательно знакомится с задачами, задачи раздаются членам команды;
- Производство. Разрабатывается контент. Менеджер проекта следит за работой членов команды по всем аспектам проекта: дизайн и web-графика, HTML и другие коды, web-программирование, управление контентом и редактирование;
- Проверка качества. На этой важной фазе проект подвергается проверке. Тестирование web-страниц, проверка на доступность и простоту использования, тестирование на совместимость с различными браузерами и операционными системами и другие проверки на качество; при необходимости вносятся исправления;
- Запуск. Во время запуска проект «оживает» и вступают в силу задачи маркетинга и соответствующей поддержки;
- Сопровождение проекта. Выполняются задачи по обновлению, поддержке и маркетингу. Менеджерам проекта на этой фазе рекомендуется собрать команду и обсудить выполненную работу: что получилось хорошо, что плохо, что хорошего можно извлечь из полученного опыта?

6.10 Составление списка основных задач

Креативные задачи. Для того чтобы упорядочить план процесса работы, разделите задачи по типу – это позволит вам более организованно их выполнять. Чтобы упорядочить креативные задачи, вам необходимо выявить все задачи, связанные с дизайном и брендами.

Примерами креативных задач могут быть следующие:

- содержимое (контент). Что является голосом сайта? Как содержимое будет систематизировано, написано и представлено?

▪ дизайн (визуальное представление). Как будет выглядеть сайт? Какие цвета, логотипы и другие идентификаторы компании необходимо изучить и воспроизвести?

▪ мультимедиа. Требуется ли мультимедиа для сайта? Если да, то какие технические приемы будут использоваться?

Менеджер проекта может работать с членами своей команды, чтобы прийти к четкому перечню креативных (созидательных, изобретательных) задач, соответствующих целям проекта.

Технические задачи. Как и креативные задачи, технические задачи должны быть определены и вставлены в рамки проекта. Технические задачи включают в себя следующие:

▪ определение потребностей разметки и скриптинга в соответствии с пожеланиями клиента (HTML, XHMTL, CSS и JavaScript);

▪ выяснение, необходимы ли языки приложений, и, при наличии такой необходимости, определение языка и операционной системы;

▪ выяснение, нужна ли база данных; если да, то определение оптимальной базы данных;

▪ администрирование web-сервера.

Административные задачи. Существуют многочисленные административные задачи, которыми, в основном, занимается менеджер проекта, даже несмотря на то что иногда ему или ей приходится поручать часть своих функций кому-то другому.

Административные задачи включают в себя следующее:

- четкое определение проекта;
- составление бюджета проекта;
- назначение членов команды и распределение задач;
- наблюдение за выполнением задач в рамках рабочего процесса проекта;
- ведение дел с клиентом, выполнение его требований;
- обеспечение доступности ресурсов;
- управление временными рамками проекта и решение потенциальных проблем.

Задачи маркетинга. Еще одна область, где потенциальные результаты зависят от работы всей команды - это маркетинг. Задачи маркетинга и задачи создания великолепно выглядящего продукта иногда бывают несбалансированными.

Определение этих задач до начала работы может обеспечить хорошие результаты в процессе работы над проектом, а также помочь

менеджеру проекта в предотвращении задержек, связанных с разногласиями в команде.

Задачи маркетинга следующие:

- анализ потенциальных посетителей (целевой аудитории);
- поиск наиболее подходящих методов маркетинга для продукта или услуги (рекламный маркетинг, размещение в поисковых машинах, рекламные мероприятия, а также взаимные ссылки на дружественных сайтах);
- организация публикаций, в том числе написание и доставка пресс-релизов, подготовка и планирование особых мероприятий при запуске сайта;
- оценка потребностей рынка на долгий срок, в том числе планирование распространения рекламных писем после запуска сайта и, при необходимости, постпроизводства.

Задача проверки качества. Когда начинается производство сайта и все принимаются за работу, внимание переключается на эффективное управление проектом, где нельзя обойтись без проверки качества и этапов тестирования.

Если менеджер проекта с самого начала не занимается проверкой качества или если эти проверки не планируются должным образом, это может привести к значительным сбоям в рабочем процессе.

Этап проверки качества должен включать в себя следующие пункты:

- проверка разметки и визуализации;
- тестирование всех возможностей разрабатываемого программного обеспечения;
- проверки скорости загрузки (особенно, если ожидается, что у сайта будет большой трафик);
- тестирование на совместимость с разными браузерами и операционными системами;
- тестирование на доступность;
- тестирование на простоту использования (юзабилити);
- проверка редактором всего содержания.

Способы проверки качества зависят от проекта, его объема и содержания. Необходимо определить способы тестирования сайта на простоту использования, так как от нее во многом зависит дальнейшее существование проекта. Например, если вы собираетесь привлекать посторонних экспертов для тестирования на юзабилити, будьте готовы к возможным задержкам в работе над проектом – это поможет вам избежать срыва сроков его сдачи.

6.11 Поощрение сотрудничества

Как говорилось ранее, некоторые эксперты считают, что поощрение сотрудничества – не очень хорошее дело. Их аргументация такова: если слишком разным людям позволяется сотрудничать, то на достижение цели уходит гораздо больше времени. Несмотря на то, что обычно так оно и есть, а степень сотрудничества при работе над проектом должна быть продиктована определенной ситуацией, хороший менеджер проекта все же должен включать определенные моменты коллективного решения проблем в работу.

Без сотрудничества может возникнуть огромное количество проблем. Необходимо разработать хорошую стратегию для поощрения эффективных коллективных встреч сотрудников.

Ниже приводятся некоторые советы по проведению таких эффективных встреч:

- установите определенную цель для встречи, например: «В конце данной встречи, мы определим, какие браузеры должен поддерживать наш web-сайт»;
- убедитесь, что каждый перед встречей получил план, где выделены темы, цели, время начала и конца;
- попросите людей подготовиться к встрече заранее. Дизайнера можно попросить подготовить макеты внешнего вида (layout) сайта в различных web-браузерах, администратора сервера - собрать журнальные данные (log files), сообщающие о том, какие браузеры используют посетители данного сайта, а маркетолог может принести статистические данные о посетителях данного или аналогичного проектируемому сайта;
- каждому участнику необходимо предоставить определенное время для обсуждения темы в начале дискуссии и в конце встречи;
- после каждой встречи подводите итоги с перечнем действий и результатов. Чтобы эффективно управлять командой (даже если вы работаете одни с привлечением посторонних специалистов), необходимо помочь каждому участнику проекта лучше ознакомиться с ним, а также почувствовать свое личное участие в успехе проекта. Тогда люди будут стараться прикладывать максимум своих усилий, что положительно скажется на результатах проекта.

Если вы поддерживаете сотрудничество, необходимо включить встречи-обсуждения в план процесса работы над проектом, а менеджеру нужно в данном случае быть настойчивым, но дипломатичным.

6.12 Управление изменениями объема

Изменение объема - это выход проекта за рамки установленного вами плана. При необходимости вы можете увеличить сроки выполнения проекта, но помните, что это последняя мера.

Само собой, изменения объема часто происходят даже при очень хорошей работе менеджеров и управляемых ими команд.

Итак, что же делать, если вы понимаете, что не укладываетесь в изначально установленные сроки. Сядьте и произведите повторную оценку:

- выявите проблемы;
- убеждайте участников проекта заняться их решением, при этом не жертвуя выполнением важных заданий;
- подумайте об использовании дополнительных ресурсов (человеческих или технических), если вам кажется, что это больше поможет в преодолении увеличения объема, нежели более эффективное использование имеющихся в наличии ресурсов;
- если ситуация становится неуправляемой, подумайте о привлечении постороннего менеджера, который сможет помочь решить возникшие проблемы.

В web-дизайне и разработке существует еще масса трудноопределимых факторов, которые подвергают риску работу над web-проектом.

Хорошие менеджеры проектов стараются использовать все свои знания и навыки, чтобы избежать изменений объема. Но стоит помнить, что качество всегда должно соответствовать потраченному времени и материальным ресурсам.

В заключение отметим, что управление web-проектами все еще находится на стадии становления, прежде всего из-за проблем, упомянутых выше. Тем не менее, используя естественный инстинкт и опыт других проектов, причем, не только в сети Интернет, разумные менеджеры проектов или их участники смогут отыскать полезные инновационные способы, которые помогут в решении возникающих проблем.

Вопросы для самоконтроля:

1. Чем сайт может помочь бизнесу?
2. Какова роль концепции в успехе или провале веб-проекта?
2. Что общего и в чем отличие дизайна от контента?
3. Как устранять разногласия при возникновении проблем с разработкой и продвижением сайта?
4. Назовите оптимальный состав малой веб-команды.

7 Проблемы и ошибки при разработке веб-проектов

7.1 Распространенные проблемы при управлении веб-проектами

Общее отношение

Большинство менеджеров веб-проектов вырастает из разработчиков. При этом часто возникает ряд проблем:

- не у всякого исполнителя есть нацеленность на результат. Порой, работа воспринимается как процесс. Тогда как, у менеджера, должна присутствовать личная заинтересованность в своевременной сдаче проекта. И это должно заключаться не столько в получении процентов, сколько в определенном настрое, мировоззрении;
- чрезмерный интерес к реализации. То есть вместо управленческих функций, менеджер начинает диктовать правила написания кода или формирования архитектуры;
- можно привести в пример реальный случай, когда разработчика повысили до ПМ. Через некоторое время, он заглянул в код, управляемого им проекта, и ужаснулся. “Если хочешь, чтобы что-то было сделано хорошо – сделай это сам” - сказал он и взялся за кодирование. При этом он был и остается отличным специалистом. Возможно, это была его личная неудача, так как когда ПМ начинает заниматься кодом, то отвлекается и теряет контроль над проектом;
- характер человека так же играет немаловажную роль. Довольно часто можно встретить ситуацию, когда проблемы возникают из-за не сделанного в нужное время телефонного звонка или из-за неотправленного письма;
- стоит отметить и общее отношение к ошибкам. Бывает и так, что из-за страха перед начальством некоторые проблемы остаются не озвученными. Что через некоторое время может привести к более серьезным последствиям, чем, если бы проблема была решена на ранней стадии;
- и самый тяжелый случай: “Менеджеры меня гоняли, теперь уж погоняю я”. То есть речь не идет об управлении, а скорее о самоутверждении. При этом забывается, что основная задача менеджера это не гонять сотрудников, а создать людям условия для работы и контролировать процесс (чтобы в заданные сроки, за заданный бюджет была реализована данная функциональность). Недостаточный контроль – так же частая проблема;

• очень тяжело бывает работать с руководителями, которые считают, что люди – это ресурс. И это притом, что и менеджеры, и разработчики все те же наемные работники, просто у них различные роли в проекте.

Сроки

Наверное, одна из самых больных тем. Рынок веб-девелопмента насыщен. И за каждого клиента приходится биться, придумывая и сроки поменьше и цены пониже. Это сильно сказывается на состоянии команды разработчиков, которым часто приходится работать в состоянии аврала.

Иногда встречается и другой подход. Сроки выполнения задач уточняются у исполнителей. Однако об адекватности подобных оценок мало кто задумывается.

Бывает и так, что сроки навязываются заказчиком. Это не такой уж и плохой вариант как кажется. Но только при том условии, если заказчик готов согласовать объем работ и принять бюджет (или же наоборот).

Отсутствие плана проекта

Конечно же, план есть. Только в голове менеджера и не документирован. Хотя, обычно это совсем не план, а общее понимание необходимых работ для завершения проекта. В управлении проектами основная задача – это создание предсказуемости. Что к данному моменту времени можно с высокой вероятностью назвать выполненный (и не выполненный) объем работ. В случае экстренных ситуаций план может помочь определить текущее состояние проекта, быстрее сориентироваться, и принять верное решение. Наличие плана дает еще возможность контроля над ходом работ.

Бессистемность при управление требованиями

Любой проект должен начинаться со сбора и анализа требований. Как иначе понять, что действительно нужно клиенту? Проблема в том, что не всякий заказчик четко понимает, что именно он хочет от данного проекта. При создании сайтов это довольно распространенная ситуация. В подобной ситуации есть два основных решения.

Первое – провести переговоры, и определить реальные требования и желаемый результат от создания проекта. Да, это не всегда просто. Порой одних переговоров может быть не достаточно.

Однако с подобным подходом повышается вероятность, что заказчик получит действительно то, на что он хотел потратить свои деньги.

Второе решение более популярно. Основано оно на установке “Мы знаем, что Вам нужно”. При выборе данного подхода продается продукт, составленный из готовых модулей. Для веб – разработок, пожалуй, это довольно важно. Действительно, регулярное создание сайтов “с нуля” (если такое вообще возможно при большом потоке заказов) – это излишнее расточительство ресурсов. Но в таком случае, заказчику навязывается ряд ограничений, имеющихся у данного готового продукта. Соответственно чаще возникает ситуация, когда проект готов, а заказчик говорит: “Но получилось совсем не то, что мы хотели”.

Бессистемность сбора требований ведет к следующим проблемам:

- объем работ растет, причем не контролируемо;
- сроки изменяются произвольно;
- невозможно провести полноценное проектирование при нечетких требованиях. Это может отразиться на качестве конечного продукта;
- в любой момент времени заказчик может отказаться от своих слов. Как итог – потерянное время (в лучшем случае);
- общий список требований не согласовывается. Соответственно, конечный продукт может не удовлетворять заказчика.

Отсутствие управления рисками

В веб-проектах, как правило, об управлении рисками никто не думает. Понятно, что они, в общем-то, одни и те же: неадекватная оценка объема работ и сроков, задержки при согласовании ТЗ или макетов, несвоевременное предоставление оборудования и так далее. Эти риски игнорируются, хотя правильнее было все-таки указать в договоре о выполнении работ варианты действий в этих случаях. А регулярно сорванные сроки – это риск испортить репутацию.

Отсутствие совещаний

Проведение совещаний по веб-проекту – довольно большая редкость. Во всяком случае, если они и проводятся, то подготовить и заблаговременно разослать повестку обычно забывают. Естественно, что с таким подходом, про составление протокола вообще речи не идет. Сами же совещания частенько сводятся к парным обсуждениям технических вопросов. В итоге, получается пустая трата времени: что

обсуждали – не понятно, какие решения приняты – никто не запомнил, что дальше делать – тоже не ясно.

Работа над ошибками

Менеджеры веб-проектов обычно даже не запоминают, сколько проектов у них было. Получается, что опыт ведения проектов не накапливается. Так как при отсутствии привычки составлять соответствующие документы, с менеджерами из компании будут уходить и их знания, которыми не всегда можно поделиться при сдаче-приеме дел. Соответственно, процесс развития в управлении может двигаться крайне медленно.

Подведение итога

Возможно, основная проблема веб-девелопмента в незрелости. В этой сфере очень много молодых специалистов без опыта или специального образования. Многие, получив этот самый начальный опыт, переходят к более серьезным софтверным проектам, освобождая место новичкам.

Иногда возникает вопрос, а целесообразно ли усложнять процесс разработки сайтов, внедряя серьезный менеджмент проектов. Можно встретить убеждение: “Не думай как лучше, просто сделай этот сайт.” Это целесообразно, если это позволит выдавать более качественный продукт за меньшее время. Что, соответственно, приведет к снижению расходов. Плюс к тому – бонус в качестве более спокойной, предсказуемой работы – а это один из ключей к снижению текучести кадров. А может быть, дело и в том, что заказчики еще не готовы платить по нескольку десятков (сотен) тысяч долларов за качественную разработку. И правда, зачем, если знакомые студенты сделают ту же работу, только за несколько сотен долларов.

7.2 Типичные менеджерские ошибки заказчика сайта

Отсутствие четкого осознания целей проекта

Очень часто при создании сайта у заказчика отсутствует четкое понимание целей, которые стоят перед проектом. И в итоге результат может оказаться плачевным: вы “что-то” попросили, мы “что-то” и сделали. Четкое понимание, что сайт нужен только для галочки и указания на визитках (а для многих компаний это так и есть) избавило бы множество заказчиков от головной боли и лишних трат.

Помните, что сайт – это не набор красивых изображений в Интернете, а, прежде всего, маркетинговый инструмент, который должен выполнять свое предназначение, а не “просто быть”.

Крайне рекомендуется перед началом проекта провести предпроектное исследование и заказать разработку веб-стратегии, которая позволит сформировать некое видение проекта на всех этапах его жизненного пути.

Неправильная оценка и распределение бюджета проекта

Довольно часто заказчик не может правильно оценить бюджет проекта и правильно расставить приоритеты по аспектам работ. Очень часто заказчик исходит из личных предпочтений и устоявшихся стереотипов (например, “дизайн сайта не важен, главное - информация”). Это неверно, бюджет должен формироваться исходя из четкого понимания целей и задач проекта. Если вы определили, что основная задача сайта – поддержание бренда или определенной марки продукта в Сети, глупо экономить на разработке дизайн-концепции.

Типичная ошибка – большой бюджет на продвижение проекта (допустим, сотни тысяч долларов в год) - и мизерный на сам сайт. Подумайте, вкладывая такие деньги в рекламу, логично потратить немного больше денег на разработку, чтобы посетители сайта не были разочарованы. Верно и обратное: не имеет смысла раздувать бюджет на сайт, если никаких целей, кроме, собственно, его наличия нет.

Если у вас есть сомнения в собственных возможностях определения общего бюджета или распределения его по составным частям, обратитесь за консультацией во внешнюю экспертную организацию (только не к подрядчику). Желательно, чтобы эта организация не занималась разработкой сайтов, а специализировалась именно на консалтинге, иначе ее анализ может быть предвзятым.

Затягивание согласования работ

Главная причина затягивания сроков со стороны заказчика – мучительные согласования результатов работ. Особенно это касается разработки дизайн-концепции сайта. Чем больше людей со стороны заказчика участвуют в этом процессе (а по дизайну каждый считает себя специалистом) – тем больше хаоса привносится в проект. Это необходимо исключить – у проекта должен быть менеджер, курирующий все вопросы по проекту, а так же четко выделенный человек, принимающий окончательные решения. Разработчику обязательно нужно обеспечить доступ как к менеджеру, так и принимающему решения лицу, как бы трудно это ни было. Остальных “советчиков” необходимо исключить из процесса, потому что выполнения всех их пожеланий и комментариев (зачастую необоснованных и противоречащих друг другу) затянет проект на

неопределенный срок. Это ни в коей мере не касается этапа тестирования сайта и выявления ошибок – тут нужно привлечь максимально широкий круг людей, чтобы вычистить все огрехи. Нужно четко построить модель взаимодействия по проекту с разработчиком именно на уровне менеджмента компании.

При принятии решения по дизайну необходимо максимально абстрагироваться от собственных предпочтений и мыслить в категориях пользователя и общей адекватности предлагаемых вариантов.

Очень часто заказчик начинает “придираться” к мелочам в дизайне, хотя на самом деле ему не нравится предложенный концепт в принципе, но он просто не может (или опасается) это выразить. Важно отследить этот момент и предложить разработать новый вариант. Разработчику это будет проще, чем потратить месяцы на согласование не играющих роли мелочей, которые все равно не добавят удовлетворенности заказчику. Не теряйте за мелочами общего видения проекта!

Еще один важный аспект – не пускайте проект на самотек. Очень часто после первого срыва сроков по этапам, заказчик расслабляется и перестает оперативно реагировать на ситуацию. Это может привести к огромным задержкам по срокам, поскольку разработчик часто склонен сделать то же самое. Проект может попасть в “мертвую точку”, когда все потеряли к нему интерес. Сдвинуть ситуацию с этой точки может стоить огромных усилий.

Неправильная организация этапа подготовки материалов на сайт

Второй по важности аспект, влияющий на затягивание сроков – неправильная организация процесса подготовки материалов на сайт. Очень часто заказчик подходит к этому процессу слишком оптимистично, обещая предоставить материалы в кратчайший срок. Это иллюзия. Как правило, подготовка таких материалов – дополнительная обязанность людей, занимающихся чем-то внутри компании.

Постарайтесь выделить какое-то время сотрудников непосредственно на подготовку материалов на сайт. Установите жесткий дедлайн и примите важное решение – если контент не будет готов к обозначенному сроку, то он не будет готов никогда. Со стороны разработчика будет также абсолютно оправдано установить некий срок, после которого он не будет принимать от вас никаких

материалов, и их отсутствие не будет влиять на процесс согласования его работы.

Соберите готовый материал и закажите копирайтинг недостающего, если это необходимо (крупные студии-разработчики с радостью предложат вам свою помощь). Не забывайте предоставлять разработчику максимально полную информацию, он всегда сможет отмести лишнее.

“Хорошая ситуация” – когда на момент старта проекта в студию приезжает менеджер со стороны клиента и привозит кипу бумаг и электронные материалы (все-все, что он смог собрать). Разработчик не обманывает вас, когда говорит, что материалы крайне важны для его работы. Это действительно так. Идеальная ситуация – 100%-ая готовность материалов на момент начала работ по дизайну. Наличие материалов к этому моменты позволит вам существенно ускорить процесс, а также лишить разработчика типичной отговорки “не было материалов, поэтому задержали сроки”, хотя, как правило, это не является отговоркой, а действительной причиной.

Отсутствие дедлайна работ

Жесткий дедлайн, о котором известно вам и разработчику – отличное средство мотивации деятельности по проекту с обеих сторон. Его наличие позволяет отмести мелочи и долгое согласование незначительных моментов, сформировать нацеленность на результат. Если жесткого дедлайна нет – придумайте его (вплоть до указания финансовых санкций в случае его несоблюдения). Отсутствие дедлайна расслабит как вас, так и разработчика.

Оптимизм исполнителя при составлении календарного плана

Довольно часто при утверждении календарного плана разработчик рассматривает “идеальную ситуацию” и указывает слишком малые сроки. Идеальных проектов не бывает, это надо помнить. Календарный план нужно оценить на общую адекватность, обязательно проверить на наличие этапов на согласование работ и подписание документов, получение оплаты и пр.

Кроме того, нужно помнить, что малые сроки в плане – конкурентное преимущество разработчика, и он мог намеренно занизить их на стадии коммерческого предложения. С этим можно бороться прописыванием жестких санкций в договоре, а также предложением пересмотреть первичный план, например, по итогам составления ТЗ.

Нарушение типового цикла проведения работ

Очень часто при работе над сайтом нарушается типовой цикл разработки проекта. Это может происходить как по инициативе разработчика, так и заказчика. Процесс по возможности должен идти последовательно по циклу. Поскольку очень часто результат по предыдущему этапу влияет на следующий, необходимо продолжать работы, только согласовав и утвердив предыдущий этап.

Например, заказчик часто требует начать параллельно с этапом дизайна этап сборки и верстки сайта. Это в корне неверно, поскольку при внесении изменений в дизайн-макеты может поменяться вся концепция ресурса, и все выполненные работы по дальнейшим этапам пойдут насмарку.

Необходимо помнить, что процесс разработки сайтов подчиняется общепринятым принципам ведения проекта и разработки ПО. Согласно этим принципам, рекомендуется по возможности вести процесс итерационно, корректируя собственные действия по мере приобретения опыта и новых знаний по проекту. Работая над большим проектом, крайне полезно создать и отладить работу облегченной версии, постепенно дорабатывая новый функционал.

Создание “сайта-памятника”

Еще одна типовая ошибка заказчика – считать, что после создания сайта его работа по развитию Интернет-направления заканчивается. Это в корне не верно. Создание сайта – точка “ноль” для его существования. Чтобы сайт стал эффективным маркетинговым инструментом, необходимо разработать стратегию его продвижения, а также на постоянной основе заниматься его обновлениями, мониторингом состояния, произведением улучшений и доработок, развитием.

В сети существует множество примеров красивых “сайтов-памятников”, которым не уделяется внимания со стороны их владельцев. Они висят “мертвым грузом”, не принося никакой пользы и сводя к нулю все усилия, потраченные на их разработку.

Небольшой пример – большинство заказчиков хотят видеть на своем сайте ленту новостей, которая публикуется на главной странице. При этом у большинства компаний не хватает сил поддерживать ее в актуальном состоянии (или просто нет должного количества информационных поводов). Определите этот момент еще на стадии проектирования, ведь нет ничего хуже сайта с новостями за прошлый год на первой странице – он моментально создает у

пользователя ощущение “заброшенности ресурса”. Лучше не публикуйте новости вообще, они, как правило, не очень интересны пользователю.

Регистрация домена студией или лично менеджером проекта

Зачастую такой вопрос, как регистрация домена, отдается на откуп разработчику, и разработчик регистрирует домен на себя. В случае возникновения конфликтной ситуации (особенно на стадии, когда проект уже запущен и имеет какой-то индекс цитируемости, а почта и адрес сайта указаны в рекламной продукции). Компания подрядчик может разыграть этот козырь, угрожая снять сайт и почтовый сервер с вашего домена. Это же относится и к регистрации домена лично на имя менеджера с вашей стороны. Если отношения не сложатся, он сможет забрать его себе, оставив вас ни с чем. Поэтому надо обязательно проконтролировать, чтобы домен был зарегистрирован именно на вашу компанию как на юрлицо.

Большая предоплата фрилансеру

Работая с фрилансером, необходимо помнить. Что получение большой предоплаты крайне расслабляет человека и сводит его мотивацию практически к нулю. Этот грустный факт подтвержден многолетним опытом. Не давайте предоплату частнику более 10-20% процентов, даже если вы очень в нем уверены. Это не касается работы с крупными студиями – получение предоплаты в 40-60% процентов от бюджета проекта является нормой для рынка.

Отсутствие контроля откатов

Не надо забывать о существующей повсеместно практике откатов. Если ваш менеджер, которому вы поручили создание сайта, настойчиво рекомендует конкретного исполнителя, и у вас появляются сомнения в разумности представленной сметы, ее необходимо проверить. Отправьте запрос в 2-3 компании, работающие в диапазоне вашего общего бюджета, и сравните их с предлагаемой сметой. Возможно, вы будете неприятно удивлены.

Статика или динамика: что выбрать с учетом развития сайта

Все сайты в сети можно разделить на две большие группы: статические и динамические сайты. С точки зрения посетителя сайта порой не важно, на какой странице он находится, статической или динамической, иногда даже трудно точно это определить. Но я хочу

рассмотреть такое разделение с точки зрения разработчика - создателя сайта.

Статическими являются страницы, которые целиком хранятся на сервере и показываются посетителю в своем неизменном виде (следует учесть, что статическая страница может содержать некоторые изменяемые элементы, например баннеры, однако она все равно остается статической).

Статическим называется сайт, большинство или все страницы которого являются статическими. Таких сайтов довольно много, с них, можно сказать, начинался интернет.

Динамическими являются страницы, формируемые сервером из нескольких частей или получаемые путем внесения либо изменения данных в страницу.

Она собирается несколькими различными способами из данных, хранящихся на сервере, и только после этого показывается посетителю.

Первым вариантом может быть объединение нескольких (двух и более) отдельных частей в одну страницу - это самый простой способ генерации.

Вторым вариантом является заполнение шаблонной страницы какой-либо информацией, хранящейся отдельно или получаемой в результате работы алгоритма (например, в результате вычислений).

Третьим, и, пожалуй, самым распространенным вариантом является сочетание первых двух во всевозможных вариациях, т.е. страница собирается из нескольких кусочков, в которые при этом вносятся различные изменения.

Динамический сайт похож на компьютерную игру. В ней есть определенный сценарий, свои персонажи и интерьеры, но финальная картинка получается только после совмещения всех этих частей, причем не без участия пользователя.

7.3 Преимущества и недостатки динамических сайтов

Напрашивается вопрос, а зачем вообще нужно делать страницу динамической, разбивать ее на части, хранить информацию где-то отдельно и т.д.? Не проще ли создать сайт таким, как его должен видеть посетитель?

Как говорилось ранее, со статических сайтов начинался интернет, динамические страницы и сайты появились позднее, но начали теснить своих прародителей, а это значит, что они имеют свои

преимущества. Вот давайте и рассмотрим, какие преимущества имеют динамические сайты по сравнению со статическими, ну а чтобы обзор был полным, уделим время и недостаткам.

Хочу сразу заметить, что, давая определения, я начинал с описания отдельных страниц. Это было сделано для того, чтобы вам было понятнее, о чем идет речь. Теперь же, при сравнении, я буду рассматривать целые сайты: статические и динамические. Соответственно и преимущества будут касаться именно готовых сайтов, т.к. подчас для страниц они просто не подходят.

Итак, с появлением языков программирования, выполняемых на стороне сервера, появилась возможность вносить изменения в данные отправляемые посетителю. В итоге это дает:

- разделение информации и дизайна;
- упрощение модификации и обновления страниц;
- возможность изменять контент, реагируя на действия посетителя.

Рассмотрим каждый пункт поподробнее.

Разделение информации и дизайна сайта

Использование динамических страниц позволяет хранить некий шаблон дизайна, в который, в зависимости от страницы, на которую зашел посетитель, помещается необходимое наполнение. Такой вариант очень удобен, ведь для всего сайта создается один или несколько шаблонов, и все изменения дизайна, которые требуется сделать на сайте, производятся только с ними.

В качестве примера представьте себе корпоративный сайт, на котором несколько сотен страниц (это не так уж много по нынешним меркам). И предположим, организация решила поменять свой логотип, нет ничего проще - если сайт статический, нужно внести изменения в каждую страницу. А теперь представьте, что сайт состоит из тысяч страниц, а изменения приходится делать постоянно. Сколько временных и человеческих ресурсов для этого потребуется?

Если же сайт динамический, все гораздо проще. Изменения вносятся в один или несколько шаблонных файлов, и все страницы сайта автоматически изменяются.

Данный подход также позволяет разграничить полномочия людей, занимающихся наполнением и модификацией сайта. Т.е. одни сотрудники могут заниматься дизайном, другие наполнением страниц. В идеале они даже не будут пересекаться, только посетитель

сайта будет видеть результат их совместной работы на экране своего монитора.

С точки зрения организации работы и разделения труда, вариант создания сайта на основании шаблонов практически идеален. Человек, ответственный за дизайн сайта не сможет вмешаться в процесс наполнения страниц контентом, и за все недочеты в дизайне будет нести ответственность только он. И наоборот, тот, кто занимается информационным наполнением сайта, не сможет нарушить дизайн сайта. Каждый занимается своим делом, не мешая другим. Это ускоряет работу и снижает затраты.

Упрощение модификации и обновления страниц на сайте

Разделение информации и дизайна на сайте позволяет ускорить процесс обновления и наполнения сайта, т.к. не требует от людей, выполняющих эти операции, знаний в областях html-разметки, графики и т.д. Т.е. можно даже подготовить дизайн сайта (заказать у фирмы, предоставляющей такие услуги), а потом производить наполнение сайта собственными силами, не затрачивая средства на постоянное привлечение сторонних или включение в свой штат специалистов по дизайну, что потребовалось бы при статической организации сайта. Да и временные затраты на обновление значительно сокращаются.

Возможность изменять контент сайта, реагируя на действия посетителя

Третье достижение стало прорывом на пути развития глобальной сети. Посудите сами, все преимущества, описанные в первых двух пунктах, облегчали работу создателей сайтов и снижали их расходы, но для посетителей это было не очень важно. А вот третье преимущество отразилось непосредственно на них. Только благодаря ему вы можете видеть огромное количество электронных магазинов, виртуальных клубов, интернет-игр и т.п. Только возможность изменять наполнение сайта под конкретного посетителя позволило воплотиться в жизнь этим проектам.

Еще один пример – это форумы и чаты, коих на просторах сети развелось бесчисленное множество. Все они создаются динамически, без поддержки этой технологии ни один из них не смог бы существовать. Доски объявлений, клубы по интересам, виртуальные игры и соревнования - ничего этого бы не смогли увидеть.

Недостатки динамических сайтов

Как и у всего остального в нашем неидеальном мире, у динамических сайтов есть свои недостатки.

Первым недостатком является необходимость использования дополнительных программных средств для построения динамического сайта. На статическом сайте все страницы уже готовы, серверу остается только показать их посетителю, а на динамическом сайте необходимо вносить в них какие-то изменения, для этого требуется соответствующие программные решения.

В зависимости от сложности сайта, трудоемкость и стоимость разработки таких программ может очень сильно варьироваться. Сейчас существует множество готовых решений для создания сайта, в том числе и бесплатных. В сети можно найти не один десяток всевозможных бесплатных скриптов, которые позволят вам создать на сайте форум, доску объявлений, клуб знакомств, магазин и т.д. Но если требуется что-то специфическое, здесь не обойтись без дополнительных разработок.

Вторым недостатком является повышение требований к аппаратным мощностям серверных систем. Этот недостаток непосредственно следует из предыдущего, т.к. теперь серверу требуется еще выполнить какую-то программу для модификации страницы сайта, а только потом выдать ее посетителю. Особенно заметной эта проблема становится на сайтах с большой посещаемостью. Часто в таких случаях приходится производить дополнительные оптимизации кода для нормальной работы сайта.

Следовательно, стоимость услуг, по содержанию такого сайта, намного выше, нежели статического. Хотя сейчас, даже многие бесплатные хостинги поддерживают возможность создания динамических сайтов, не говоря уже о платных, где все необходимое входит в стандартный набор услуг.

Третьим недостатком, также вытекающим из первого, является сложность больших структурных изменений сайта. Представьте, у вас на сайте размещен форум, а вам захотелось, чтобы был еще и чат. Если этой возможности изначально не было, то, как бы вы не меняли дизайн или наполнение форума, он не сможет превратиться в чат. Вам для этого придется изменить программу.

Несмотря на эти недостатки, динамических сайтов в сети становится все больше и больше, видимо, описанные мной ранее преимущества перекрывают все недостатки. Теперь давайте

рассмотрим, что же на сегодняшний момент имеется в сети, какие возможности может получить человек, желающий создать сайт.

Системы управления контентом

Технологии не стоят на месте, развиваются и возможности, предлагаемые для создания сайтов. Сейчас все большую популярность завоевывают системы управления контентом. Что это такое? Давайте разберемся.

CMS - это аббревиатура от **Content Management System**, что в дословном переводе - Система Управления контентом сайта. Проще говоря, это тот самый программный комплекс, который позволяет вам изменять дизайн и наполнение сайта таким образом, как вам требуется.

Сейчас в сети можно найти множество таких систем, какие-то из них бесплатные, какие-то платные. Часто фирмы-разработчики предоставляют своим клиентам такие системы. Каждая система индивидуальна и обладает своими достоинствами и недостатками.

Преимуществом визуальной системы редактирования является то, что вы сразу видите, как будет выглядеть та или иная страница на сайте. Вам не нужно задумываться о том, как данные хранятся в системе, как собирается страница, достаточно наполнить ее информацией и сайт готов.

Такое решение очень выгодно для небольших компаний, которые не могут позволить себе держать в штате программиста, дизайнера, верстальщика, достаточно только оператора для ввода и обновления информации. Такой вариант выгоден и частным лицам, желающим создать сайт, но не имеющим в этом большого опыта.

Также данная технология предусматривает возможность создания интернет-магазина, на базе конструктора для создания сайта

Не стоит считать, что динамические сайты - это всегда правильно решение. Всегда следует исходить из ситуации. Если нет необходимости динамического изменения данных, а сайт предполагается не очень большим, можно обойтись и статическими страницами. Ведь бывают одностраничные сайты, так зачем под них писать программы, когда проще создать статическую страницу. Но не во всех случаях это возможно.

Следует исходить из целесообразности использования того или иного средства для создания сайта. Здесь следует учитывать как первоначальные, так и последующие финансовые и трудовые затраты, необходимые на поддержание сайта.

Вопросы для самоконтроля:

1. Назовите распространенные проблемы, возникающие при управлении веб-проектами.
2. Назовите типичные менеджерские ошибки, совершаемые заказчиком при разработке сайта
3. Назовите достоинства и недостатки динамических сайтов.
4. Что такое система управления контентом?

8 Работа в среде Microsoft Project

Предлагаемые Microsoft инструменты управления проектами предусматривают наличие одного из вариантов конфигурации MS Project, в частности, MS Project 2007.

Необходимо отметить, что все программные инструменты управления проектами (в том числе и MS Project) следует рассматривать и использовать как средства поддержки принятия решений менеджером проекта: с помощью MS Project менеджер может оценить эффективность нескольких альтернативных вариантов реализации проекта и выбрать стратегию, в наибольшей степени отвечающую интересам компании и целям проекта.

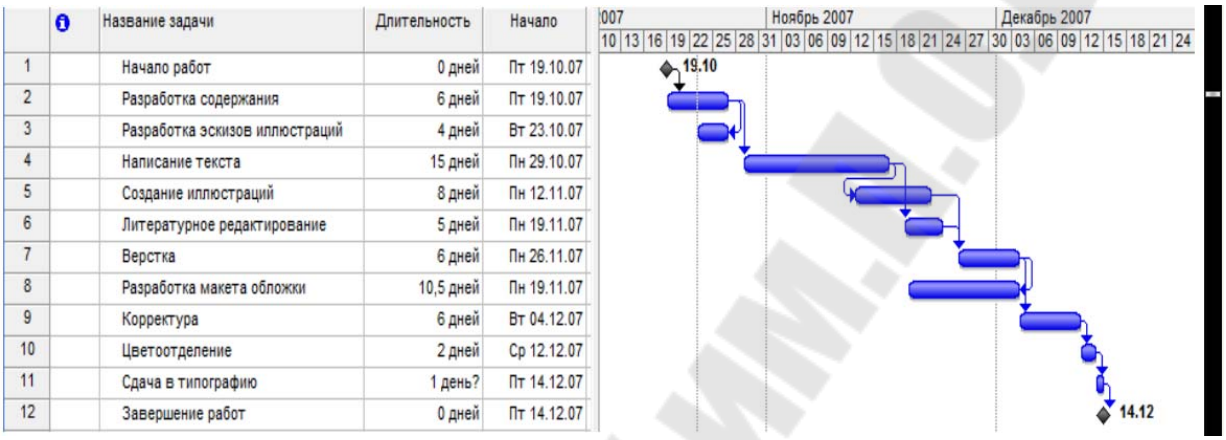
Применение MS Project на стадии планирования поможет руководителю ответить на следующие вопросы:

- насколько вообще реально воплощение в жизнь данного проекта?
- какие конкретно работы необходимо выполнить для достижения целей проекта?
- какой состав исполнителей, соисполнителей и какие виды материальных ресурсов потребуются для реализации проекта?
- какова стоимость проекта и как наиболее выгодно распределить во времени финансовые затраты на реализацию проекта? Кто должен отвечать за те или иные виды работ?
- насколько велик риск и каков возможный ущерб при завершении проекта на той или иной стадии?

Для ответа на первый вопрос требуется провести полный анализ проекта по методу критического пути с использованием ресурсного планирования, однако без излишней детализации. В этом отношении весьма большую помощь могут оказать шаблоны, входящие в состав

стандартной конфигурации MS Project. Каждый из шаблонов относится к определенной сфере, и может считаться своеобразным стандартом соответствующего плана проекта. Внеся в него необходимые коррективы в соответствии с особенностями конкретного проекта, можно получить вполне реалистичную оценку возможного развития событий и требуемых затрат.

Например, на рисунке 8.1 приведен фрагмент диаграммы Ганта.



	Название задачи	Длительность	Начало
1	Начало работ	0 дней	Пт 19.10.07
2	Разработка содержания	6 дней	Пт 19.10.07
3	Разработка эскизов иллюстраций	4 дней	Вт 23.10.07
4	Написание текста	15 дней	Пн 29.10.07
5	Создание иллюстраций	8 дней	Пн 12.11.07
6	Литературное редактирование	5 дней	Пн 19.11.07
7	Верстка	6 дней	Пн 26.11.07
8	Разработка макета обложки	10,5 дней	Пн 19.11.07
9	Корректура	6 дней	Вт 04.12.07
10	Цветоотделение	2 дней	Ср 12.12.07
11	Сдача в типографию	1 день?	Пт 14.12.07
12	Завершение работ	0 дней	Пт 14.12.07

Рисунок. 8.1 – Диаграмма Ганта

Ответ на второй вопрос также может быть получен с помощью одного из стандартных расписаний. Если структуру проекта приходится создавать вручную (нет шаблона0, то MS Project 2007 способен оказать существенную помощь благодаря средствам построения сетевого графика (Network Diagram). Для работ проекта автоматически устанавливаются параметры, заданные по умолчанию (такие как длительность, календарные даты начала и окончания и т. д.). На рисунке 8.2 показан один из возможных вариантов представления сетевого графика, наиболее близкий его «бумажному» аналогу.

На основе сетевого графика автоматически формируется календарный план в виде диаграммы Ганта. Определив структуру расписания в виде сетевого графика, вы получаете «заготовку» календарного графика с привязкой сроков выполнения работ к реальным датам (Рисунок 8.3).

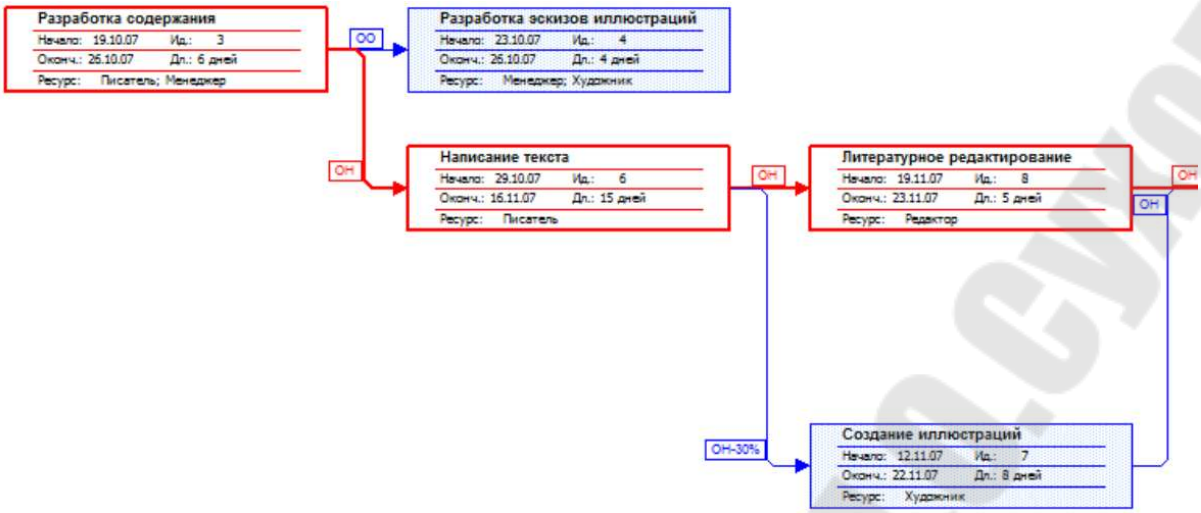


Рисунок 8.2 – Сетевой график

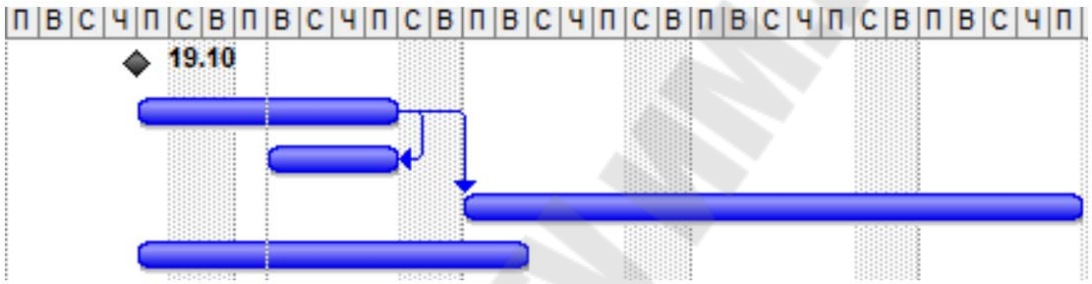


Рисунок 8.3 – Разные типы связей между задачами

Чтобы получить ответ на третий по счету из перечисленных выше вопросов, требуется выполнить назначение ресурсов. В качестве ресурсов проекта могут быть заданы либо уникальные для него исполнители и материалы, либо назначены виды ресурсов, использовавшихся в предыдущих проектах (или взятые из шаблонов). Обобщенную информацию об используемых в проекте ресурсах можно получить с помощью таблицы ресурсов (Рисунок 8.4).

	Название ресурса	Тип	Единицы измерения материалов	Краткое название	Группа	Макс. единиц	Стандартная ставка	Ставка сверхурочных	Затраты на использ.	Начисление	Базовый календарь
1	Писатель	Трудовой		П	Люди	100%	0,00р./ч	0,00р./ч	7 000,00р.	По окончании	Стандартный
2	Редактор	Трудовой		Р	Люди	100%	50,00р./ч	100,00р./ч	0,00р.	Пропорциональ	Стандартный
3	Художник	Трудовой		Х	Люди	100%	70,00р./ч	140,00р./ч	0,00р.	Пропорциональ	Стандартный
4	Верстальщик	Трудовой		В	Люди	100%	50,00р./ч	100,00р./ч	0,00р.	Пропорциональ	Стандартный
5	Корректор	Трудовой		К	Люди	100%	50,00р./ч	100,00р./ч	0,00р.	Пропорциональ	Стандартный
6	Менеджер	Трудовой		М	Люди	100%	100,00р./ч	200,00р./ч	0,00р.	Пропорциональ	Стандартный
7	Компьютер	Трудовой		К	Техника	400%	0,00р./ч	0,00р./ч	0,00р.	Пропорциональ	Стандартный

Рисунок 8.4 –Таблица ресурсов

А более детальную информацию – на основе анализа назначений. Для каждого ресурса могут быть построены гистограммы его загрузки и стоимости. После назначения очередного ресурса (с указанием его стоимости и объема) выполняется автоматический пересчет стоимости проекта, благодаря чему очень легко получить сравнительную оценку различных вариантов назначений.

Для проведения стоимостного анализа проекта в MS Project 2007 используется так называемый «метод освоенного объема» (Earned Value Analysis), с помощью которого может быть проведен анализ затрат либо на текущую дату, либо на заданную календарную дату.

Любой, даже самый хороший план не застрахован от случайностей.

Чтобы адекватно анализировать риски проекта, необходимо иметь его детализированный план. Как правило, выбор методов и средств для анализа рисков зависит от специфики проекта, состава и уровня подготовки группы проекта. В частности, простым и вместе с тем эффективным средством является сравнение нескольких версий (сценариев) расписания проекта.

При использовании **анализа по** методу **PERT** таких сценариев должно быть три: **наиболее вероятный (ожидаемый), оптимистичный и пессимистичный**. Для сравнительной оценки длительности проекта по этим трем сценариям в составе MS Project 2007 имеется специальный инструмент – процедура *PERT,* С се помощью вы можете описать и сравнить между собой параметры проекта для трех альтернативных сценариев развития событий.

Для более сложных ситуаций могут быть созданы соответствующие макросы, реализованные с помощью VBA (язык программирования Visual Basic Application).

Завершая короткий обзор основных возможностей MS Project 2007, отметим, что на любой стадии работы над проектом вы всегда будете чувствовать поддержку со стороны разработчиков. Либо в виде всплывающих окон с подсказками, либо в форме смарт-тегов, либо в какой-то другой форме.

Вопросы для самоконтроля:

1. На какие попросы планирования может помочь ответить модель, созданная в Microsoft Project?
2. Что такое сетевой график?

3. Какие типы связей между задачами существуют в Microsoft Project?

4. Какие виды ресурсов существуют в Microsoft Project?

5. Что такое анализ по методу PERT?

6. Что такое фазы, вехи и завершающие задачи в Microsoft Project?

7. Что такое назначения в Microsoft Project и какие задачи планирования они решают?

Список использованных источников

1. Шафер Д., Фатрел Р., Шафер Л. Управление программными проектами: достижение оптимального качества при минимуме затрат. – М.: Вильямс, 2003. – 1136 с.

2. Хелдман К. Профессиональное управление проектом – М.: БИНОМ. Лаборатория знаний, 2005. – 517 с.

3. Ройс У. Управление проектами по созданий программного обеспечения. – М.: Издательство «Лори», 2000. – 431 с.

4. Брауде Э. Технологии разработки программного обеспечения. – СПб.: Питер, 2004. – 655 с.

5. Беркун С. Искусство управления IT-проектами. – СПб.: Питер, 2007. – 400 с.

6. Microsoft Office Project 2007. Библия пользователя. М.: Вильямс, 2008. – 800 с.

7. Microsoft Office Project Professional 2007. Управление проектами: Практическое пособие. – СПб.: Корона-Век, 2008. – 480 с.

8. Богданов В. Управление проектами в Microsoft Office Project 2007. Учебный курс. – СПб.: Питер, 2008. – 592 с.

9. Милошевич Д. Набор инструментов для управления проектами. – М.: Компания АиТи; ДМК Пресс, 2008. – 729 с.

Осипенко Александр Николаевич

УПРАВЛЕНИЕ WEB-ПРОЕКТАМИ

Учебно-методическое пособие по одноименному курсу для слушателей специальности 1-40 01 74 «Web-дизайн и компьютерная графика» заочной формы обучения

Подписано в печать 30.10.16.
Формат 60x84/16. Бумага офсетная. Гарнитура «Таймс».
Ризография. Усл. печ. л. 4,42. Уч.-изд. л. 4,69.
Изд. № 15.
http://www.gstu.by

Отпечатано на цифровом дуплекаторе
с макета оригинала авторского для внутреннего использования.
Учреждение образования «Гомельский государственный
технический университет имени П.О. Сухого».
246746, г. Гомель, пр. Октября, 48.