Software engineering

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики и информатики

Кафедра технологий программирования

Software

Engineering

Минск 2015

Содержание

**[Глава 1. IT-проект](#_Toc433719201)** [5](#_Toc433719201)

[IT-проект и его структура 5](#_Toc433719202)

[Структура проекта 5](#_Toc433719203)

[Эффективная деятельность 6](#_Toc433719204)

[Определение термина IT-проект 7](#_Toc433719205)

[Планирование бюджета 8](#_Toc433719206)

[Реализация IT-проекта 10](#_Toc433719207)

[Прибыль в Open Source проектах 10](#_Toc433719208)

[Аналоги среди IT-проектами 13](#_Toc433719209)

[Управление IT-проектом 13](#_Toc433719210)

[Жизненный цикл IT-проекта 14](#_Toc433719211)

[Жизненный цикл IT-проекта 14](#_Toc433719212)

[Жизненный цикл проекта 14](#_Toc433719213)

[Жизненный цикл IT-проекта 14](#_Toc433719214)

[Особенности open-source проектов 17](#_Toc433719215)

[Моделирование жизненного цикла 17](#_Toc433719216)

[Мозговой штурм 17](#_Toc433719217)

[Управление жизненным циклом 17](#_Toc433719218)

[Внутренние меры контроля 17](#_Toc433719219)

[Open-Source проекты 18](#_Toc433719220)

[**Глава 2. Менеджмент IT-проекта** 18](#_Toc433719221)

[Менеджмент IT-проектов 18](#_Toc433719222)

[Роли в команде и их функции 18](#_Toc433719223)

[Для чего нужно разделение ролей 18](#_Toc433719224)

[Распределение функций между исполнителями 18](#_Toc433719225)

[Подход компании Microsoft к распределению ролей 20](#_Toc433719226)

[Основные роли, встречающиеся на проекте и их обязанности 22](#_Toc433719227)

[Возможность совмещения ролей 26](#_Toc433719228)

[Распределение ролей посредством RACI-матрицы 26](#_Toc433719229)

[Пример использования RACI-матрицы 28](#_Toc433719230)

[Управление IT-проектами 29](#_Toc433719231)

[Стратегии развития крупнейших и наиболее известных IT-компаний 29](#_Toc433719232)

[Стандартизация 29](#_Toc433719233)

[**Глава 3. Архитектура компьютера и мобильных устройств** 29](#_Toc433719234)

[Архитектура компьютера 29](#_Toc433719235)

[Операционные и вычислительные системы 29](#_Toc433719236)

[Мобильные операционные системы 29](#_Toc433719237)

[**Глава 4. Языки программирования** 29](#_Toc433719238)

[Языки программирования 29](#_Toc433719239)

[Обзор языков программирования 29](#_Toc433719240)

[JavaScript 29](#_Toc433719241)

[**Глава 5. Разработка программного обеспечения** 29](#_Toc433719242)

[Парадигмы программирования 29](#_Toc433719243)

[Стили программирования 29](#_Toc433719244)

[Чистый код и его оптимизация 29](#_Toc433719245)

[Паттерны проектирования 29](#_Toc433719246)

[Разработка мобильных приложений 29](#_Toc433719247)

[Тестирование 29](#_Toc433719248)

[**Глава 6. Сложные науки и технологии** 29](#_Toc433719249)

[Трансляторы 29](#_Toc433719250)

[Искусственный интеллект 29](#_Toc433719251)

[Виртуальная реальность 29](#_Toc433719252)

[Нейронные сети 29](#_Toc433719253)

[Облачные вычисления 29](#_Toc433719254)

[**Глава 7. Компьютерная и информационная безопасность** 29](#_Toc433719255)

[**Приложение** 29](#_Toc433719256)

[Список вопросов, возникнувших в ходе обсуждения 29](#_Toc433719257)

[IT-проект 29](#_Toc433719258)

[IT-проект и его структура 29](#_Toc433719259)

[Жизненный цикл IT-проекта и фазы разработки ПО 30](#_Toc433719260)

[Моделирование жизненного цикла 31](#_Toc433719261)

[Мозговой штурм 32](#_Toc433719262)

[Доклад Григорьева 33](#_Toc433719263)

[Менеджмент IT-проекта 34](#_Toc433719264)

[Менеджмент IT-проектов 34](#_Toc433719265)

[Роли в команде и их функции 35](#_Toc433719266)

[Архитектура компьютера и мобильных устройств 37](#_Toc433719267)

[Архитектура компьютера 37](#_Toc433719268)

[Операционные и вычислительные системы 38](#_Toc433719269)

[Мобильные операционные системы 38](#_Toc433719270)

[Языки программирования 40](#_Toc433719271)

[Языки программирования. Обзор языков программирования 40](#_Toc433719272)

[JavaScript 42](#_Toc433719273)

[Разработка программного обеспечения 43](#_Toc433719274)

[Парадигмы программирования 43](#_Toc433719275)

[Стили программирования 43](#_Toc433719276)

[Чистый код и его оптимизация 43](#_Toc433719277)

[Паттерны проектирования 43](#_Toc433719278)

[Разработка мобильных приложений 43](#_Toc433719279)

[Тестирование 43](#_Toc433719280)

[Сложные науки и технологии 43](#_Toc433719281)

[Трансляторы 43](#_Toc433719282)

[Искусственный интеллект 43](#_Toc433719283)

[Виртуальная реальность 43](#_Toc433719284)

[Нейронные сети 43](#_Toc433719285)

[Облачные вычисления 43](#_Toc433719286)

[Компьютерная и информационная безопасность 43](#_Toc433719287)

# **Глава 1. IT-проект**

## IT-проект и его структура

### Структура проекта

**Проект** – это временное действие, которое выполняется для создания уникального продукта или услуги. Временное обозначает, что каждый проект имеет свои определенные начало и конец. Уникальный обозначает, что продукт или услуга принципиально отличается от других аналогичных продуктов или услуг, так как принципиально отличаются условия создания этих продукта или услуги в каждом проекте.

**Участниками** проекта являются:

* собственник, заказчик, инвестор;
* менеджеры проекта;
* исполнители работ проекта;
* «окружающая» организация;
* службы контроля (технического, финансового и пр.)
* финансирующие организации, банки.

Несмотря на все многообразие существующих проектов, в команде можно выделить ряд более или менее стандартных ролей.

В первую очередь, это менеджер (руководитель) проекта — физическое лицо, несущее личную ответственность за успех проекта и осуществляющее оперативное руководство.

Как правило, в компаниях назначают куратора проекта — представителя высшего руководства, который хоть и не вникает в тонкости текущего положения дел в проекте, но контролирует его ход, следит, чтобы проект соответствовал стратегическим целям компании, а если у менеджера проекта не хватает полномочий, — помогает ему своим авторитетом.

Проектный комитет создается в компаниях, в которых бизнес построен по проектному типу. Это орган, задачи которого — отбирать проекты и контролировать их выполнение на высшем уровне, принимать ключевые решения.

В технически сложных проектах важна роль главного инженера проекта (ГИП, командный лидер), который порой по статусу равен менеджеру проекта.

В крупных проектах могут выделяться менеджеры по различным функциональным областям, например, по управлению финансами, персоналом, рисками и т. п.

Все вышеперечисленные роли образуют команду управления проектом, которая входит в команду проекта. Также участниками команды проекта являются исполнители как из числа штатных сотрудников компании, так и нанятые специально для реализации конкретного проекта. Иногда в нее включают подрядчиков и субподрядчиков.

Отдельно стоит выделить проектный офис. В простейшем случае это своего рода секретариат, в котором хранится вся документация по проекту. Он может состоять как из одного, так и из нескольких сотрудников. В более продвинутых компаниях проектный офис также играет роль методологического центра, обслуживающего все проекты организации.

Основная команда формируется исходя из потребностей проекта. Должны ли сотрудники:

* работать полный или неполный рабочий день?
* отчитываться менеджеру проекта?
* тратить на проект 100% своего рабочего времени?

Высшее руководство компании

Проектный комитет

Куратор проекта

Менеджер проект

Руководитель

функциональной области

(например, финансы)

Руководитель

функциональной области

(например, финансы)

Главный

инженер

проекта

Проектный офис,

администратор проекта

Команда

управления

проектом

Исполнители

Подрядчики

Субподрядчики

Команда проекта

### Эффективная деятельность

Показатели *эффективной деятельности* команды:

* Ясное понимание цели проекта и нацеленность на конечный результат;
* Четкое распределение функций и ответственности;
* Наличие плана развития команды;
* Командная солидарность;
* Взаимопонимание и бесконфликтность;
* Посещаемость рабочих совещаний и активное участие в решении проблем.

Для *обеспечения эффективного руководства* командой проектный менеджер должен:

* Определить организационную структуру команды, подобрать ее состав, распределить функции и обязанности;
* Назначить руководителей и ответственных за отдельные направления;
* Своевременно спланировать, распределить и скоординировать работу;
* Четко объяснить цели и задачи;
* Преодолевать препятствия и избегать конфликтов;
* Обеспечить трудовую активность команды силой личного авторитета, заинтересовать каждого члена команды, оказывать им помощь и проявлять участие, поддержать перспективу команды;
* Привлекать всех к решению задач;
* Обеспечить поддержку проекта со стороны руководства и регулирование отношений с окружением команды, создавать привлекательный имидж команды.

**Менеджер проекта** – лицо, отвечающее за успех проекта, а также за подбор и работу своей команды и завершение проекта. Это происходит в рамках ограничений, наложенных партнерскими и другими организациями, внешними по отношению к проектной команде.

Менеджер проекта должен быть назначен как можно раньше. Обычно менеджера проекта назначает *Заказчик*.

Искусство управления человеческими ресурсами и координация их в проекте реализуется менеджером посредством применения административных и поведенческих знаний для достижения определенных проектных целей в содержании, затратах, времени, качестве и удовлетворении участников проекта.

Менеджер проекта является ключевой фигурой в команде проекта. От его лидерских качеств, организационных способностей, харизмы, умения вовремя принимать решения и сглаживать конфликты напрямую зависит успех проекта.

### Определение термина IT-проект

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

С одной стороны, эти работы соответствуют классическому определению проекта "Проект – это комплекс усилий, предпринимаемых с целью получения конкретных уникальных результатов в рамках отведенного времени и в пределах утвержденного бюджета, который выделяется на оплату ресурсов, используемых или потребляемых в ходе проекта". С другой стороны, они обладают известными отличительными особенностями:

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

### Планирование бюджета

*Планирование бюджета* на IT-проекту можно разделить на три этапа: 

1. **Сбор информации**

Для составления (и обоснования) бюджета на ИТ нужно знать:

1. Что компания уже купила в прошлом и за что продолжает платить в настоящем, а именно:

* используемое и находящееся в резерве оборудование,
* лицензии на программное обеспечение,
* сервисные контракты,
* операторские услуги,
* расходные материалы и затраты на них.

1. Какие информационные технологии используются в компании и для чего.  
   Полный список услуг (ИТ-сервисов), которые использует бизнес в своей работе, начиная от банальных «электронная почта», «печать документов», «телефонная связь», заканчивая системами управления, безопасности и специфическими бизнес-приложениями.
2. Какие цели, планы и задачи у компании на ближайший финансовый период, какие проблемы должны быть решены? Реорганизация подразделений, увеличение персонала, появление новых задач, изменение требований по производительности, надежности и безопасности работы информационных систем. На данном этапе полезно пообщаться как с руководством компании, так и с руководителями структурных подразделений, чтобы понять их потребности по изменению качества обслуживания, надежности, удобства работы с системами, а также уточнить список используемых ими ИТ-сервисов. Дополнительно можно провести анкетирование пользователей, собрать полученные за последнее время обращения.
3. **Анализ**

Задача данного этапа – найти те точки, которые мешают в текущий момент достичь поставленных целей или могут помешать в будущем. Для этого рекомендуется проанализировать следующие показатели:

1. **Производительность**

Хватает ли производительности систем в настоящий момент всем пользователям? Соответствует ли текущий уровень производительности систем ожиданиям бизнеса? Что сейчас является слабым звеном? Хватит ли производительности, если нагрузка вырастет в соответствии с планами по развитию?

1. **Надежность**

Достаточные ли меры приняты для обеспечения сохранности данных? Допустимо ли менять оборудование по мере выхода из строя или стоит его обновить заранее? Сможете ли вы в случае сбоя восстановить работоспособность в требуемые сроки или требуется заранее приобрести дополнительное оборудование или программное обеспечение для этого? Есть ли какие-то известные проблемы, которые влияют на безотказность работы ИТ-систем?

1. **Функциональность**

Решают ли существующие приложения задачи пользователей и бизнеса? Решают ли они их эффективно? Что компания недополучает сейчас? Что нужно изменить, чтобы соответствовать будущим требованиям? Актуальны ли существующие бизнес-приложения вообще?

1. **Безопасность**

Защищены ли данные компании от внешних угроз? А от внутренних? Соответствует ли система защиты уровню угроз? Как изменятся требования к информационной безопасности в обозримом будущем?

1. **Удобство**

Создает ли что-нибудь дискомфорт в работе пользователей с компьютерной техникой? Удобно ли расположены принтеры в офисе, сильно ли шумят компьютеры, все ли интерфейсы и системы понятны для пользователей, жалуются ли они еще на что-то? Можно ли это улучшить?

1. **Операционные расходы**

Оптимальны ли существующие операционные расходы? Соответствуют ли они рыночной стоимости? В какие суммы ежемесячно обходится компании тот или иной сервис? Из чего состоят эти расходы? Можно ли их уменьшить без ущерба для компании?

1. **Запасы**

Есть ли необходимые расходные материалы? Сколько нужно будет дополнительного оборудования и лицензий в случае расширения? Нужны ли будут дополнительные разовые или постоянные услуги в случае планируемого развития бизнеса?

1. **Формирование бюджета и обоснование**

Собственно, вся основная работа выполнена на прошлых этапах. Последняя задача — представить полученные выводы руководству в понятном и удобном для принятия решений виде. Обычно разделяют все расходы по следующим категориям:

1. Операционные расходы на поддержание деятельности: расходные материалы, сервисные контракты, услуги, оплата труда специалистов.
2. Необходимые капитальные вложения, в случае отсутствия которых возможны серьезные потери для бизнеса. Сюда относятся расходы, которых компании не избежать и вопрос лишь в том, будет она инвестировать деньги в это заранее, или, когда уже понесет обозначенные потери.
3. Рекомендуемые инвестиции – в сочетании с необходимыми позволяют значительно повысить показатели работы, а также устранить риски, которые могут оказать негативное влияние на бизнес.
4. Расходы, связанные с развитием – необходимый объем инвестиций для обеспечения работы систем и поддержания качественных показателей в случае реализации планов по росту бизнеса.
5. Возможные инвестиции, позволяющие улучшить функциональные возможности и/или удобство работы сотрудников с ИТ-системами. Данный пункт является красной тряпкой для финансистов, позволяя им при согласовании бюджета отказать вам в этой части и тем самым с честью выполнить свой долг, не опасаясь за последствия.
6. Обоснование каждой из статей бюджета. Это самый важный подпункт. Бизнес, к сожалению, не оперирует понятиями «шестилетний сервер» и ничего не понимает в технике — он оперирует только категориями потребностей в ИТ-сервисах, возможностями, рисками и их стоимостью для бизнеса. Каталог услуг (ИТ-сервисов), который вы делали в самом начале, нужен вам для того, чтобы общаться с руководством компании на одном языке – это ваша точка взаимопонимания. Выявив потребность в оборудовании, программном обеспечении, персонале и пр. обосновывайте необходимость в них конкретными показателями работы конечных ИТ-сервисов (риски, качество, скорость реакции и пр.), которые получает или получит бизнес.

### Реализация IT-проекта

В **реализации ИТ-проектов** следует обратить внимание на следующие особенности:

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

### Прибыль в Open Source проектах

Зачастую в ИТ проектах заказчик оплачивает всю его стоимость, однако следует отметить, что в **Open Source** проектах *финансирование* может происходить и другими способами:

* Продажа профессиональных услуг

Финансовая отдача от затрат на программное обеспечение с открытым исходным кодом может исходить от продажи услуг, таких как обучение, техническая поддержка, или консультации, а не самого программного обеспечения.

Другая возможность предлагает open source программное обеспечение только в виде исходного кода, при этом предоставляя исполняемые бинарные файлы только платным клиентам, предлагая коммерческие услуги по компиляции и созданию инсталляционных пакетов программного обеспечения. Кроме того, предоставление open source  как коммерческий товар на физическом носителе (например, DVD).

Успешные Open source компании, использующие эту бизнес-модель: RedHat и IBM; более специализированным примером является Revolution Analytics.

* Продажа фирменных товаров

Некоторые FOSS организации, например, Mozilla Foundation или Wikimedia Foundation, пытаются продавать фирменные товары: футболки, кофейные кружки. Это может рассматриваться в качестве дополнительной услуги для сообщества пользователей.

* Продажа программного обеспечения как услуги

Платная подписка на онлайн-аккаунты и доступ к серверу для клиентов является способом получения прибыли на базе программного обеспечения с открытым исходным кодом. Кроме того, комбинация настольных ПК с сервисом, называется программное обеспечение плюс услуги. Предоставление услуг облачных вычислений и программного обеспечения как услуги (SaaS) без предоставления самого программного обеспечения с открытым исходным кодом, ни в двоичной ни в исходной форме соответствует большинстве лицензий с открытым исходным кодом (за исключением AGPL).

* Партнерство с финансирующими организациями

Прочие финансовые ситуации включают партнерские отношения с другими компаниями. Правительства, университеты, компании или другие неправительственные организации могут разрабатывать у себя или нанять подрядчика для внутренних пользовательских модификаций программного обеспечения, а затем выпустить этот код под открытой open source лицензией. Некоторые организации поддерживают разработку программного обеспечения с открытым исходным кодом грантами или стипендиями, например, Google Summer of Code initiative основанную в 2005.

* Добровольные пожертвования

Появление систем Интернет микроплатежей в 2000-х годах таких, как PayPal, Flattr и Bitcoin помогает этому.

* Денежное вознаграждение за выполнение задачи (bounty)

Пользователи конкретного программного обеспечения могут объединиться вместе и собрать 1еньги для open source проекта для разработки желаемого функционала.

* Предварительный заказ / Crowdfunding / модель обратная bounty

Новая возможность финансирования проектов СПО –  Crowdfunding, модель похожа на пред-заказ, а также на перевернутую модель bounty. Как правило, организуется на базе веб-платформ, таких как Kickstarter, Indiegogo, Catincan или Bountysource. Пример успешного финансирования: австралийской программист Timothy Arceri, который предложил за $2500 реализовать в течение двух недель расширение OpenGL 4.3 для библиотеки Mesa.

* Программное обеспечение, содержащее рекламу

С целью коммерциализации FOSS, многие компании (в том числе Google, Mozilla и Canonical) перешли к экономической модели заработка на рекламе в программном обеспечении.

Например, приложение с открытым исходным кодом AdBlock Plus получает деньги от Google за расширение белого списка разрешенных рекламных блоков в обход блокировщика рекламы в браузере.

* Продажа дополнительных проприетарных расширений

Некоторые компании продают собственные дополнительные расширения, модули, плагины к open source программному продукту. Это может соответствовать свободным лицензиям, если сделано технически достаточно тщательно.

* Продажа необходимых проприетарных частей программного продукта

Вариант подхода выше, заключается в хранении нужного контента (например, аудио, видео для игр, графику или другие художественные активы) в закрытом программном продукте, выпуская сам исходный код под открытой лицензией. Хотя такой подход вполне совместим с большинством open source лицензий, клиенты должны купить контент, чтобы иметь полную и работающую версию программного продукта.

Похожий на этот прием привязывания open source программного продукта к проприетарной аппаратной части называется «*тивоизация*» и проходит с большинством open source лицензий за исключением GPLv3, которая прямо запрещает подобное использование.

* Пере-лицензирование под проприетарной лицензией

Если программный продукт использует только собственное программное обеспечение и программное обеспечение с открытым исходным кодом под разрешительной свободной лицензией, то компания может повторно лицензировать конечный программный продукт под проприетарной лицензией и продать продукт без исходного кода и софтверных свобод. Например, [Apple](http://tim4dev.com/2014/11/iphone-apple-budet-otklyuchat-prilozheniya/) Inc. является активным эксплуататором этого подхода, используя исходный код и программное обеспечение из различных open source проектов, например, ядро ​​операционной системы BSD Unix под лицензией BSD было использовано в компьютерах Mac, которые продаются как патентованных продукты.

* Обфускация исходного кода

Подход состоящий в запутывании исходного кода, для коммерциализации с некоторыми открытыми лицензиями, защищая при этом важные коммерческие тайны, интеллектуальную собственность и технические ноу-хау. Этот подход был использован в ряде случаев, например, Nvidia в своих драйверах для графических карт.

* Задержка с выпуском open-source software

Некоторые компании предоставляют самую свежую версию только для платных клиентов. Далее вендор делает ответвление от программного проекта без авторского права, добавляет к нему дополнения с закрытым кодом и продает конечный программный продукт. После некоторого периода времени патчи интегрируются обратно в приложение под той же лицензией, что и остальной часть кода. Эта бизнес-модель называется версия с отставанием.

Экстремальный вариант такой модели является бизнес-практика, которую популяризировали Id Software и 3D Realms, которые выпустили несколько своих программных продуктов под свободной лицензией после долгого коммерческого периода, в течение которого произошел возврат инвестиций.

* FOSS и экономика

По правовым исследованиям предпринимательства в Гарвардской школе права, свободное программное обеспечение является наиболее видимой частью новой экономики на основе общего равного производства информации, знаний и культуры. В качестве примеров они приводит ряд проектов FOSS, которые включают как free, так и open source ПО.

Эта новая экономика уже на стадии разработки. С целью коммерциализации FOSS, многие компании, Google является самым успешным, движутся в направлении экономической модели программного обеспечения, содержащего рекламу (AdWare).

### Аналоги среди IT-проектами

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

### Управление IT-проектом

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

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

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

### Жизненный цикл IT-проекта

**Жизненный цикл** – это последовательность фаз проекта, через которые он должен пройти для гарантированного достижения целей проекта, в нашем случае – для реализации некоторой информационной технологии.

Организационная структура подразумевает выделение ролей исполнителей, которые необходимы для реализации проекта, определение взаимоотношений между ними и распределение ответственности за выполнение задач.

## Жизненный цикл IT-проекта

### Жизненный цикл проекта

Любой проект имеет ограниченный отрезок времени существования. Наличие этого отрезка времени означает, что у проектов есть жизненный цикл. Жизненный цикл последовательно проходить через четыре стадии:

1. Определение
2. Планирование
3. Выполнение
4. Сдача

Все начинается, когда появляется идея проекта. Происходит его определение. Здесь поднимается вопрос о цели проекта, о его целесообразности. Далее проект переходит в стадию планирования, где планируется бюджет, принимаются ключевые решения. Бюджет проекта имеет большое значение. Так, например, на воплощение проекта с большим бюджетом, в среднем затрачивается меньше времени, чем на проект с меньшим бюджетом. К этой стадии необходимо относится ответственно, потому что если на этом этапе совершается архитектурная ошибка, то в случае, если ошибку невозможно исправить, проект либо меняет свою структуру (а это тянет за собой дополнительные расходы ресурсов), либо прекращает свое существование. На стадии выполнения происходит непосредственное воплощение проекта. Здесь проект может претерпевать значительные изменения. После стадии выполнения проект сдается. Под сдачей подразумевается введение проекта в эксплуатацию.

### Жизненный цикл IT-проекта

В IT-проектах можно выделить следующие особенности:

* по мере реализации проектов выполняется большое количество уточнение и корректировка требований и содержания;
* крайне велико влияние человеческого фактора на сроки выполнения;
* из-за трудности планирования творческой деятельности, отсутствуют единые нормативы и стандарты;
* сохраняется повышенный уровень риска, вплоть до непредсказуемости результатов;
* происходит постоянное совершенствование технологии выполнения работ, что тянет за собой постоянное изменение структуры проекта.

Что касается стандартизации, то современные стандарты не предписывают четких и однозначных схем построения структуры жизненного цикла проекта. Это сделано намеренно, поскольку достаточно жесткие схемы препятствуют использованию более прогрессивных технологий разработки, которых появилось очень много и которые продолжают интенсивно развиваться.

Исходя из этих особенностей, можно построить структуру жизненного цикла it-проекта:

* **Начальная стадия** - цель**:** определить границы системы и собрать требования высокого уровня;
* **Стадия уточнения -** цель**:** создать архитектурную основу системы;
* **Стадия конструирования** - цель**:** создание финального продукта;
* **Стадия передачи и сопровождения** - цель**:** внедрение продукта на предприятии заказчика, обучение персонала, сопровождение и обновление установленной информационной системы.

**Начальная стадия жизненного** цикла it-проекта не отличается по структуре от любого другого проекта. На стадии уточнения производится выбор технологий, определение необходимых ресурсов (как денежных, так и человеческих), построение команды. Также здесь происходит подписание необходимых документов. Таких как техническое задание и документов, которые призваны защитить компанию-разработчика от внезапного отказа заказчика от проекта. Уже на стадии уточнения может начинаться тестирование. Такой подход называется *test-driven development* (TDD, Разработка через тестирование). Он заключается в написании тестов раньше, чем написание кода. Далее следует **стадия конструирования**, которую можно условно разделить на фазы части:

* **Проектирование** - определение характеристик архитектуры системы, компонентов, составляющих, интерфейсов и других частей. Созданное описание, в свою очередь, является фундаментом для реализации;
* **Реализация** - создание программного продукта, исходя из созданного на этапе проектирования программного проекта;
* **Интеграция** - объединение программных компонентов и интегрирование их в среду.

Основная часть стадии конструирования, **реализация**, может быть поделена еще на 6 частей:

* **Пре-Альфа** - период времени со старта разработки до выхода стадии Альфа (или до любой другой, если стадии Альфа нет). Также так называются программы, не вышедшие еще в стадию альфа или бета, но прошедшие стадию разработки, для первичной оценки функциональных возможностей в действии. В отличие от альфа- и бета-версий, пре-альфа может включать в себя не весь спектр функциональных возможностей программы. В этом случае, подразумеваются все действия, выполняемые во время проектирования и разработки программы вплоть до тестирования. К таким действиям относятся — разработка дизайна, анализ требований, собственно разработка приложения, а также отладка отдельных модулей.
* **Альфа** - внутреннее тестирование — стадия начала тестирования программы в целом специалистами-тестерами, обычно не разработчиками программного продукта, но, как правило, внутри организации или сообществе разрабатывающих продукт. Также это может быть стадия добавления новых функциональных возможностей. Программы на данной стадии могут применяться только для ознакомления с будущими возможностями.
* **Бета** - публичное тестирование — Стадия активного бета-тестирования и отладки программы. Программы этого уровня могут быть использованы другими разработчиками программного обеспечения для испытания совместимости. Тем не менее, программы этого этапа могут содержать достаточно большое количество ошибок.
* **Релиз-кандидат** - иногда «гамма-версия» — стадия-кандидат на то, чтобы стать стабильной. Программы этой стадии прошли комплексное [тестирование](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F), благодаря чему были исправлены все найденные критические ошибки. Но в то же время существует вероятность выявления ещё некоторого числа ошибок, не замеченных при тестировании.
* **Релиз** - издание продукта, готового к тиражированию. Это стабильная версия программы, прошедшая все предыдущие стадии, в которых исправлены основные ошибки, но существует вероятность появления новых, ранее не замеченных, ошибок.
* **Пост-релиз** - издание продукта, у которого есть несколько отличий от релизного и помечается как самая первая стадия разработки следующего продукта. Такие релизы не выпускаются на продажу, а раздаются бета-тестировщикам. Эта стадия встречается редко и присуща проектам, которые делятся на отдельно реализуемые версии.

Третья по счету фаза конструирования - **интеграция**. На этой фазе происходит интеграция разрабатываемого проекта с уже созданным окружением, с которым этот проект будет функционировать.

И последняя, четвертая, стадия жизненного цикла it-проекта - **стадия передачи и сопровождения**. Передача проекта заказчику - очень сложный процесс и требует тщательного подхода и ответственности. Какие здесь возникают проблемы? Часто возникают спорные вопросы из-за того, что требования к системе были сформулированы абстрактно либо недостаточно хорошо. Качественное техническое задание - оружие компании-разработчика. Однако наличие технического задания не является ни необходимым, ни достаточным основанием для полноценного закрытия работ. Если исполнитель придерживается ГОСТов, к приемочным испытаниям на основании технического задания должен быть разработан и согласован с заказчиком дополнительный документ "Программа и методика испытаний". В нем должны быть прописаны принципы оценки реализации требований технического задания. Также на этой стадии производится поддержка и сопровождение проекта. Длительность сопровождения зависит от сложности, целесообразности, выгодности и области применения проекта. Так, например, социальные сети поддерживаются и сопровождаются на протяжении всего времени использования, а проект, который перестал приносить прибыль, быстро теряет поддержку со стороны производителя.

### Особенности open-source проектов

У open-source проектов выделяется ряд специфических особенностей:

* техническое задание либо размытое, либо и вовсе отсутствует;
* большое количество исполнителей (contributors);
* отсутствие четко сформированного будущего проекта;
* непредсказуемость продолжительности жизни.

## Моделирование жизненного цикла

## Мозговой штурм

## Управление жизненным циклом

Хотя управление и соответствие нормативным требованиям встречаются на протяжении всего жизненного цикла, однако их представление, область действия и цели зависят от конкретного этапа. Например, действия по управлению изменениями на этапах «Планирование» и «Эксплуатация» будут иметь другую значимость и отличаться по составу участников и используемым факторам.

### Внутренние меры контроля

Процедуры и меры контроля следует разделить между несколькими людьми, каждый из которых будет выполнять свою часть. В этом случае внутренние меры контроля должны обеспечить надлежащее объединение результатов, полученных разными людьми, и гарантировать, что никому не удалось уклониться от выполнения. В финансовых вопросах проблемы контроля являются еще более важными. Отсутствие эффективного контроля может привести к ошибкам в бухгалтерском учете или даже мошенничеству и хищению.

Внутренние меры контроля представлены во всех областях, с которыми работает IT-подразделение. Одни меры контроля предназначены для физической среды, в которой находится инфраструктура центров данных, а другие используются непосредственно для технологий (например, определяют конфигурацию и перечень лиц, которым предоставлен доступ к административным функциям). Некоторые меры контроля используются при доступе к данным и применяются на различных этапах жизненного цикла данных — от шифрования до авторизации, восстановления и защиты данных.

Подтверждением того, что IT-услуга фактически контролируется на протяжении всего жизненного цикла, является следующее:

* Определение общих целей для каждого этапа жизненного цикла
* Определение рисков, связанных с достижением этих целей
* Определение методов управления рисками в виде мер внутреннего контроля по смягчению последствий рисков

Руководство несет ответственность за выработку целей, оценку хода работ и достижение результатов. В частности, управление включает процессы принятия решений (меры контроля), помогающие руководству выполнять эти требования. Каждый этап жизненного цикла IT-услуги содержит одну или несколько процедур управленческого анализа, функционирующих как управленческие меры контроля. Это означает, что нужные люди будут собраны вместе в надлежащее время и обеспечены информацией, необходимой для принятия управленческих решений.

*Контроль исполнения проекта* - процесс сравнения показателей плановых и фактических показателей выполнения проекта, анализ отклонений и их причин, оценка возможных альтернатив и принятие, в случае необходимости, решений о корректирующих действиях для ликвидации нежелательных отклонений.

Контроль проекта может включать следующие процедуры:

* Сбор отчетности о ходе работ по проекту
* Анализ текущего состояния проекта относительно основных базовых показателей (результаты, стоимость, время)
* Прогнозирование достижения целей проекта
* Подготовка и анализ последствий корректирующих воздействий
* Принятие решений о воздействиях и изменениях

Организация контроля может следить за:

* Качеством работ
* Ходом и темпом работ
* Стоимостью и сроками

## Open-Source проекты

# **Глава 2. Менеджмент IT-проекта**

## Менеджмент IT-проектов

## Роли в команде и их функции

### Для чего нужно разделение ролей

Каждый IT-специалист идет по пути наименьшего сопротивления, как правило, пытаясь сбросить часть работы на коллегу. Для избегания возможных конфликтов в команде нужно четко разграничить участок работ каждого участника, в том числе заказчика. 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.

### Распределение функций между исполнителями

Функции, выполняемые разработчиками*,* — понятие неформализованное. В разных проектах оно может обретать свое содержание. Заметим, что в рамках деятельности менеджера любого проекта необходимо организовать распределение функций проекта между исполнителями. В результате ее выполнения члены команды, выполняющей проект, начинают играть соответствующие роли.  
Обычно роль объединяет родственные функции. Также принято обозначать роли их главными функциями. Продолжая только что приведенную иллюстрацию функций, выполняемых разработчиками проекта, укажем на следующие роли: кодировщик — действующее лицо, главной функцией которого является кодирование, аналитик — тот, кто занимается анализом требований. Подобную характеристику можно дать и тестировщику. Что же касается функции отладки, то в реальных проектах она обычно подразделяется на несколько видов: отладка компонентов, которой занимаются разработчики компонентов (например, те же кодировщики) и комплексная отладка, которая может поручаться специально выделенным сотрудникам или рассматриваться в качестве задачи группы разработчиков. Часто выполняются такие работы, как отладка спецификаций, декомпозиция проекта и др. Иными словами, функция отладки обычно не рассматривается как образующая роль, а распределяется по нескольким ролям в соответствии с принятой стратегией развития проекта. Роли назначаются на начальной стадии жизненного цикла проекта.

Не следует путать функции, которые предписано выполнять разработчику как исполнителю определенной роли, с поручениями в проекте. Поручения — это разовые или систематические задания, из которых обычно и складываются действия, необходимые для выполнения той или иной функции. Если для функции определен регламент выполнения поручений, т. е. последовательность выполнения составляющих ее поручений не требует дополнительных разъяснений для исполнителя, то такая функция называется технологической. В случае нетехнологической функции сотруднику, который выполняет соответствующую роль, приходится самому выстраивать нужную последовательность. Иными словами, технология здесь уступает место ремеслу.

При разработке любых проектов естественно стремление к повышению их технологичности, к установлению регламента для как можно большего числа функций. И одним из способов достижения этого для менеджера является использование работников с нужной квалификацией, для которых поручаемые им роли оказываются технологичными, т.е. состоящими только из технологических функций. К сожалению, реальность такова, что менеджеру приходится работать в условиях ограниченных возможностей в подборе кадров, а потому уровень технологичности выполнения проекта снижается по сравнению с идеальной ситуацией. 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.

Функции, выполняемые разработчиками в проекте, подразделяются на:

* Организационные
* Производственные

Первые создают условия для выполнения проектных заданий, вторые непосредственно связаны с этими заданиями. Часто неудачи проекта возникают из-за того, что менеджер не учитывает важность выполнения организационных функций. Так, обычно проектное задание фиксирует лишь то, что нужно предъявлять заказчику в качестве результатов. С точки зрения результатов просто не требуется знать, например, как организована передача проектных материалов между разработчиками, какая процедура отчетности предусматривается, но игнорирование задач реализации информационных потоков в проекте может привести к хаосу, что в конечном итоге отразится и на результатах.

Понятно, что как состав, так и значимость ролей разработчиков и тех, кто с ними связан, различаются в зависимости от выполняемого проекта, от коллектива исполнителей, принятой технологии, от других факторов. Неоднозначность выбора ролей приводит к тому, что их список часто становится непомерно велик. Чрезмерное увеличение числа есть следствие отождествления роли и функции и, соответственно, игнорирования понятия родственности функций. В то же время, если ролей выбрано недостаточно, есть опасность, что не все нужные в проекте функции будут охвачены планированием и управлением.

Также при выборе состава нужно учитывать направление того или иного человека. В случае крупных проектов необходимо нанимать узкопрофильным специалистов, так как они лучше знают свою сферу. Если у проекта небольшой бюджет, то можно сэкономить, наняв многопрофильных специалистов. К слову, есть такой класс разработчиков, как full-stack разработчики — это еще одна попытка «работодателей» получить задешево, то, что никогда дешевым быть не могло. Это такой же психологический прием, как и «вы же профессионал! Вы же профессионал?»

Цепочку распределений ролей можно описать в следующем виде:

* спонсор (куратор) проекта (это сотрудник (как правило, руководитель высшего звена) организации, реализующей проект, который курирует проект со стороны организации (владельца проекта), обеспечивает общий контроль и поддержку проекта) назначает менеджера проекта и обеспечивает ему необходимую поддержку.
* менеджер проекта выбирает команду управления проектом, среди которых есть командный лидер (team leader).
* командный лидер назначает разработчиков.

### Подход компании Microsoft к распределению ролей

Как конкретный разработчик может получать одновременно несколько ролей, так и роль может быть распределена между несколькими исполнителями. Когда менеджеру в конкретных условиях руководства коллективом придется распределять роли, он неизбежно столкнется с тем, что эта задача зависит и от специфики проекта, и от контингента исполнителей. В связи с этим уместно упомянуть об одном из ее решений, которое предлагается специалистами Microsoft в качестве универсального подхода.

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 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:

* **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).

### Основные роли, встречающиеся на проекте и их обязанности

Существуют следующие роли на IT-проектах:

* **Заказчик (Customer)** — отвечает за:
  + своевременный просмотр спецификаций и других присылаемых документов (с целью утвердить документ, дать комментарии, исправить неточности и т.п.);
  + внесение замечаний, дефектов, пожеланий в багтрекинговую систему;
  + своевременный просмотр каждого выпуска и предоставление комментариев.
* **Планировщик ресурсов (Planner)**:
  + выдвигает и координирует требования к проектам в организации, осуществляющей данную разработку;
  + развивает и направляет план выполнения проекта с точки зрения организации.
* **Менеджер проекта (Project Manager)** — отвечает за:
* проектная документация;
* составление плана проекта;
* согласование сроков;
* анализ возможных рисков;
* участие в подборе и утверждении проектной команды;
* разбивка продукта на компоненты и раздача их исполнителям;
* определение требуемых ресурсов и рабочей среды, их распределение внутри команды;
* постановка рабочего процесса в команде (разработка, тестирование, работа с требованиями);
* определение приоритетности задач;
* организация работы команды вокруг требуемой задачи;
* отслеживание состояния проекта, хода выполнения задач;
* отслеживание должной приоритетности выполнения задач;
* отслеживание нагрузки задачами и прогресса по задачам каждого разработчика;
* отслеживание сроков выполнения задач;
* удерживание команды в рабочем состоянии, мотивация команды;
* создание прозрачной среды общения между всеми участниками процесса;
* отслеживание удовлетворенности проектом со стороны команды;
* решение всевозможных конфликтных ситуаций внутри команды и в связке заказчик-команда;
* общение с заказчиком, управление его ожиданиями;
* предоставление заказчику отчетности о ходе выполнения задач и проекта в целом;
* презентация заказчику готовых решений, демоверсий, прототипов;
* интервьюирование новых членов команды.
* **Руководитель команды (Team Leader)**

Руководитель команды — это нечто среднее между проектным менеджером и квалифицированным разработчиком. Командный лидер обязан перевести бизнес-задачу в понятную техническую для разработчиков и сказать не только то, что нужно сделать, но и зачем это нужно.

На проектах есть две роли: менеджерская — PM, и техническая — System Architect. Командный лидер отчасти выполняет обе роли, но акцент его обязанностей направлен на менеджмент (акцент на техническую часть — это tech lead).

*Под управленческую роль* TL попадают такие обязанности, как:

* менеджмент;
* распределение и делегирование задач:
* всевозможные оценки и составление рабочего графика;
* контроль состояния проекта;
* проведение митингов;
* коммуникации с заказчиком, руководством и всеми членами команды (разработчиками, архитекторами, тестировщиками, менеджерами).

*Под техническую роль* TL попадают**:**

* участие в написании технической документации;
* выбор технологий для проекта;
* разработка архитектуры;
* обзор и анализ кода (code review);
* контроль и наставление молодых разработчиков;
* проведение технических собеседований;
* грамотное вовлечение новых членов команды в рабочий процесс;
* ответственность за техническую часть проекта.
* **Системный аналитик (Technical Leader) —** отвечает за:
* координацию и контроль деятельности по дизайну, архитектуре и кодированию;
* поддержку контроля версий;
* настройку скрипта для авто-билдера и своевременную сборку версий.
* **Архитектор (Architect)** — отвечает за:
  + проектирование архитектуры системы;
  + согласование развития работ, связанных с проектом.

Архитектор — это человек, который решает, как в конечном итоге будет выглядеть информационная система организации в целом и в деталях. Основная цель архитектора в компании заключается в том, чтобы обеспечить решение задач бизнеса при помощи информационных технологий. Причем, он должен не только сформировать решение, но и контролировать правильность его реализации.

Внутри профессии существуют специализации: *функциональная* и *техническая*. В первом случае архитектор в большей степени отвечает за общение с бизнесом и по результатам контактов определяет конструкцию системы, которая нужна заказчику. Во втором ИТ-архитектор в основном общается с разработчиками и конструирует систему изнутри.

Не всякой компании нужен ИТ-архитектор. На небольших предприятиях или там, где информационные проекты не слишком масштабны, функции ИТ-архитектора может выполнять опытный менеджер проекта, разработчик или иной технический специалист в сфере ИТ.

Иметь собственного ИТ-архитектора необходимо, в первую очередь, крупным компаниям с развитой функциональностью унаследованных систем, разветвленной региональной оргструктурой и имеющим согласованные руководством планы развития ИТ.

* **Эксперт предметной области (Domain Expert)** — отвечает за:
  + изучение сферы приложения;
  + поддержку направленности проекта на решение задач данной области.
* **Разработчик (Developer)**:
  + разработку качественного кода;
  + проведение модульного тестирования;
  + поддержку контроля версий;
  + написание пользовательской документации, относящейся к инсталляции и администрированию.

Это широкое понятие, которое может подразделяться на специальные роли (например, разработчик классов). В зависимости от сложности проекта команда может включать различное число разработчиков.

* **Бизнес аналитик (Business Analyst)** отвечает за:
  + выяснение и анализ всех требований заказчика;
  + фиксирование всех требований заказчика (в багтрекинговой системе и в функциональных спецификациях), прослеживание всех изменений в требованиях;
  + написание и поддержка спецификаций.
* **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.
* **QA менеджер (Quality Assurance manager) —** отвечает за:
* организацию и контроль процесса тестирования в проекте;
* планирование тестирования;
* участие в адаптации процесса разработки под проект, анализ его качества;
* анализ результатов тестирования и качества продукта;
* участие в управлении требованиями;
* участие в настройке багтрекинговой системы, полное прослеживание багов;
* контроль готовности нового выпуска для QA.
* **QA аналитик (QA Analyst)** — отвечает за:
* подготовку тест дизайна;
* написание тест кейс спецификаций;
* проведение тестирования;
* регистрацию багов;
* прослеживание и проверку багов;
* написание документации пользователя.

### Возможность совмещения 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 | Эффективно 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 | Оправданно |

### Распределение ролей посредством RACI-матрицы

Модель RACI — средство для выявления активностей и распределения их по ролям и зонам ответственности. Использование матрицы RACI позволяет избежать непонимания в том, кого необходимо привлечь к проекту, а также кто и что должен делать.

RACI — сокращение от основных ролей участников проекта:

* **Responsible (Исполнитель)**: Тот кому назначена эта роль отвечает за выполнение работы и достижение целей проекта. На каждом этапе может быть несколько исполнителей.
* **Accountable (Ответственный)**:  Исполнитель этой роли отвечает за качество и результаты процесса. Обладатель этой роли обеспечивается полномочиями для обратной связи с исполнителями. На каждом этапе может быть только один ответственный.
* **Consulted (Консультант, Эксперт)**: Тот кому назначена эта роль привлекается, как носитель уникальных знаний или информации. Часто в этой роли выступают эксперты в предметной области.
* **Informed (Информируемый)**: Это лицо, которого необходимо держать в курсе о ходе и результатах процесса, чаще всего в одностороннем порядке, т.к. у него нет полномочий напрямую влиять на ход проекта.

Иногда в эту модель добавляются и другие роли, например, S — supported (Оказывающий поддержку).



Для того, чтобы понимать, по какому принципу такая табличка должна рисоваться, а также как ее использовать на практике (в реальных проектах), рекомендуется уделить должное внимание следующему порядку действий при построении матрицы:

* Определяется список необходимых активностей/процессов в поставленной задаче (проекте)
* Определяется и указывается функциональные роли (людей, которые заинтересованы или которых тем или иным образом касается данная задача)
* Собирается митинг и назначаются RACI коды (собственно — буквы) конкретным ролям, непосредственно разграничиваются ответственности
* Определяются несоответствия (например, слишком много ответственных либо отсутствие таковых)
* Описывается таблица и собираются отзывы
* Контролируется выполнение назначенных ролей

По функциональным ролям, анализировать можем, отвечая на такие вопросы:

* *Много «А»* — правильно ли распределены обязанности? Есть ли в наличии «узкие места»?
* *Много «R»* — не многовато ли ответственности для одной роли?
* *Отсутствие пустых ячеек в таблице* – действительно ли эта роль должна быть вовлечена в такое количество задач?

Также, проводится анализ по выполняемым активностям:

* *Более одного «А»* — только одна роль должна быть подотчетной
* *Отсутствие «А»* — необходимо найти подотчетного
* *Более одного «R» или отсутствие такового* – кто-то должен быть ответственный, однако нетребуется, чтобы ответственность была широко распределена – есть риск того, что задача не будет выполнена
* *Много «С»* — стоит ли консультироваться с многими ролями и будет ли это эффективно?
* *Отсутствие «С» и «I»* — правильно ли установлены коммуникации?

### Пример использования RACI-матрицы

Допустим, есть авиакомпания, которая на своем сайте собирается внедрить систему online check-in. Глобальные активности, необходимы к выполнению в контексте задачи, будут приблизительно следующие: сбор требований к системе; дизайн решения; непосредственная разработка решения (development); внедрение; собственно – стадия “production”; оптимизация решения.  
  
Далее — определяется список функциональных ролей, в данной задачи возможны такие варианты: внутренний сервис провайдер (IT отдел авиакомпании) или же внешний сервис провайдер в случае отсутствия первого; ISP – компания предоставляющая хостинг для сайта авиакомпании; бизнес подразделение авиакомпании (представляющее интересы заказчика); финансовое подразделение (бухгалтерия); сервис менеджер (в зависимости от размеров организации, может входить во внутренний IT отдел); команда разработчиков (в зависимости от размеров организации, может входить во внутренний IT отдел).

Попробуем расставить RACI коды соответственно ролям и выполняемым ими активностям (ясно, что данный процесс проходит при участии всех сторон).



## Управление IT-проектами

## Стратегии развития крупнейших и наиболее известных IT-компаний

## Стандартизация

# **Глава 3. Архитектура компьютера и мобильных устройств**

## Архитектура компьютера

## Операционные и вычислительные системы

## Мобильные операционные системы

# **Глава 4. Языки программирования**

## Языки программирования

## Обзор языков программирования

### JavaScript

# **Глава 5. Разработка программного обеспечения**

## Парадигмы программирования

## Стили программирования

## Чистый код и его оптимизация

## Паттерны проектирования

## Разработка мобильных приложений

## Тестирование

# **Глава 6. Сложные науки и технологии**

## Трансляторы

## Искусственный интеллект

## Виртуальная реальность

## Нейронные сети

## Облачные вычисления

# **Глава 7. Компьютерная и информационная безопасность**

# **Приложение**

## Список вопросов, возникнувших в ходе обсуждения

### IT-проект

#### IT-проект и его структура

* Каким образом заказчик может вносить изменения в проект? (Белый А.А.)
* Чем отличается куратор проекта от менеджера? (Белый А.А.)
* Какими навыками должен обладать менеджер? (Белый А.А.)
* Что обязана делать группа людей, занимающаяся внедрением готового продукта? (Борисевич П.И.)
* Кто контролирует аудит проекта? (Гетьман С.И.)
* Как планируется бюджет проекта, и кто является инвестором? (Гетьман С.И.)
* Что означает "развертывание IT-инфраструктуры"? (Гетьман С.И.)
* Кто и зачем занимается документацией проекта? (Гетьман С.И.)
* Как влияет изменение состава исполнителей на развитие проекта? (Гетьман С.И.)
* Кто такие подрядчики и субподрядчики, их роль и структура? (Григорьев А.В.)
* Если какие-нибудь дополнения к схеме участников проекта? (Грушевский А.А.)
* Отличия open source проекта от коммерческого? (Грушевский А.А.)
* Какая роль на проекте является самой важной, а какая самой незначительной? (Ипатов А.Е.)
* Какие роли требуют высокой квалификации? (Ипатов А.Е.)
* Какие существуют части планирования? (Ипатов А.Е.)
* Какие роли могут совмещаться, а какие нет? (Ипатов А.Е.)
* Как организуется взаимодействие между командой и руководством? (Лебедев Н.А.)
* Бизнес-аналитики в структуре проекта? (Лебедев Н.А.)
* Какова роль менеджера проекта? (Лебедев Н.А.)
* Какова роль руководителей компании в структуре проекта? (Лебедев Н.А.)
* Самая ответственная должность на проекте? (Лебедев Н.А.)
* В чем заключается принципиальное различие в работе куратора и руководителя проекта? (Михальцова А.Ю.)
* Что означает фраза "проект структурирован"? (Михальцова А.Ю.)
* Какими способами можно контролировать выполнение проекта? (Ровдо Д.И.)
* Какие должности в структуре проекта обязательны, а какие нет? (Ровдо Д.И.)
* Насколько важную роль в проекте играют подрядчики? (Ровдо Д.И.)
* Как можно разрешить ситуацию, когда кто-то из структуры (руководство, исполнители) не может выполнять свою работу из-за нехватки знаний, опыта? (Трубач Г.Г.)
* Как происходит общение бизнес-аналитика с заказчиком? (Трубач Г.Г.)
* Как разработчики получают прибыль из open source проектов? (Трубач Г.Г.)
* Какова структура open source и фриланс проектов? (Трубач Г.Г.)
* Есть ли отличия между структурами IT-проектов в Беларуси и проектов других стран? Если есть, то какие? (Ярошевич Я.О.)
* Как осуществляется взаимодействие в команде между собой, если проект является интернациональным? (Ярошевич Я.О.)
* Как происходит выбор подходящей команды для проекта? (Ярошевич Я.О.)
* Как организуется структура проекта, создающийся не в IT –компании?

(Ярошевич Я.О.)

#### Жизненный цикл IT-проекта и фазы разработки ПО

* На каком этапе реализуется поддержка проекта? (Белый А.А.)
* Существуют ли отличия между жизненным циклом обычного проекта и IT-проекта? (Белый А.А.)
* Как осуществляется интеграция проекта? (Белый А.А.)
* С какой целью созданы международные стандарты жизненного цикла и в каких случаях их требуют к выполнению? (Борисевич П.И.)
* Насколько тщательно подходят к планированию жизненного цикла? (Гетьман С.И.)
* Как справиться с постоянным обновлением и улучшением различных платформ разработки? (Гетьман С.И.)
* Как долго длится сопровождение продукта? (Гетьман С.И.)
* Влияет ли бюджет проекта на его жизненный цикл? (Гетьман С.И.)
* Суть и назначение пост-релизного этапа разработки. (Григорьев А.В.)
* Роль планирования в процессе разработки проекта (Григорьев А.В.)
* Почему планирование проекта требует малых затрат? (Лебедев Н.А.)
* На какой стадии жизненного цикла определяется архитектура проекта?

(Лебедев Н.А.)

* Что может произойти с проектом, если при планировании допущена архитектурная ошибка? (Лебедев Н.А.)
* Можно ли выпускать проект на альфа-стадии его разработки? (Михальцова А.Ю.)
* Чем отличаются стадии разработки "релиз кандидат" от "публичной реализации"? (Михальцова А.Ю.)
* Означает ли фаза "завершение", что проект уже окончательно готов к использованию? (Михальцова А.Ю.)
* Когда занимаются поддержкой проекта, если в жизненном цикле отсутствует фаза post-release? (Ровдо Д.И)
* На какой стадии лучше всего начинать тестирование продукта? (Ровдо Д.И)
* Какие существуют подэтапы этапа планирования? Какие из них являются наиболее важными? (Ровдо Д.И)
* Каков жизненный цикл open source проектов и стартапов? (Трубач Г.Г.)
* Какие существуют стандарты жизненного цикла? (Трубач Г.Г.)
* Какие существуют варианты выхода из ситуации, когда заказчик отказывается от проекта и возможен ли такой случай? (Трубач Г.Г.)
* Возможно ли возвращение от более позднего этапа жизненного цикла к более раннему? Если возможно, то по каким причинам? (Ярошевич Я.О.)
* Как соотносятся между собой этапы жизненного цикла по времени? (Ярошевич Я.О.)
* Осуществляется ли поддержка проекта после релиза и в каких случаях? (Ярошевич Я.О.)

#### Моделирование жизненного цикла

* Как в Scrum методологии осуществляется контроль рисков? (Белый А.А.)
* Какая методология наиболее универсальна? (Белый А.А.)
* Можно ли менять методологию в процессе разработки IT-проекта? (Борисевич П.И.)
* Когда используется модель водопада? (Борисевич П.И.)
* Кто выбирает методологию? (Гетьман С.И.)
* Какая методология наименее затратная? (Гетьман С.И.)
* Какие методологии используют наиболее успешные компании? (Гетьман С.И.)
* Какие основные отличия между спиральной и итерационной моделью? (Григорьев А.В.)
* В чем суть экстремального программирования? (Григорьев А.В.)
* Суть Scrum методологии и ее виды. (Григорьев А.В.)
* Какая модель наиболее эффективна? (Ипатов А.Е.)
* Какая методология наиболее универсальна? (Ипатов А.Е.)
* В чем заключается простота Scrum методологии? (Лебедев Н.А.)
* Как происходит переход между спринтами в Scrum методологии? (Лебедев Н.А.)
* В чем преимущества и недостатки каждой из методологий? Какая из них наиболее оптимальна? (Лебедев Н.А.)
* Каковы минусы SCRUM-a, если они есть. (Ровдо Д.И.)
* Какая модель самая оптимальная? (Ровдо Д.И.)
* Почему модель "спираль" довольно известна? Ведь, по сути, она почти не отличается от итерационной. Какие есть особенности этой модели, которые отличают ее от других? (Ровдо Д.И.)
* Какая методология сейчас используется наиболее часто? (Трубач Г.Г.)
* От чего зависит выбор той или иной модели? (Трубач Г.Г.)
* Можно ли изменять спринт при Scrum методологии? (Трубач Г.Г.)
* Чем грозит срыв спринта при Scrum методологии? (Трубач Г.Г.)
* Можно ли изменять сроки спринта при Scrum методологии? (Трубач Г.Г.)
* Каковы основные принципы экстремального программирования? Как происходит работа с заказчиком в данной методологии? (Трубач Г.Г.)
* Какая методология сейчас наиболее популярна? (Щавровский С.А.)
* Каковы особенности Scrum методологии? (Щавровский С.А.)
* Примеры использования тех или иных методологий? (Щавровский С.А.)
* Какая модель используется чаще других? (Ярошевич Я.О.)
* В чем разница между итеративной и спиральной моделью? (Ярошевич Я.О.)
* Как выбрать лучшую модель для конкретного проекта? (Ярошевич Я.О.)
* Проводят ли компании тренинги по изучению той или иной модели? (Ярошевич Я.О.)
* Какие существуют методологии, придуманные самими компаниями? (Ярошевич Я.О.)

#### Мозговой штурм

* Может ли мозговой штурм проводится удаленно (например, по Skype)? (Гетьман С.И.)
* Что произойдет, если при мозговом штурме, в команде находятся люди, которые друг с другом в плохих отношениях? (Гетьман С.И.)
* Как можно сорвать мозговой штурм? (Гетьман С.И.)
* Проводятся ли мозговые штурмы в крупных компаниях? (Григорьев А.В.)
* Кто оценивает и отсеивает идеи: вся группа или кто-то конкретный? (Григорьев А.В.)
* Почему 6-12 человек при мозговом штурме является оптимальным количеством? Уложится ли команда в отведенное время? (Лебедев Н.А.)
* Можно ли провести мозговой штурм с самим собой? (Лебедев Н.А.)
* Почему для мозговых штурмов так важно отсутствие критики? (Михальцова А.Ю.)
* Кто отвечает за анализ идей/решений? (Михальцова А.Ю.)
* Есть ли у мозговых штурмов чёткие временные ограничения? (Михальцова А.Ю.)
* Какого рода вопросы/проблемы решаются во время мозговых штурмов? (Михальцова А.Ю.)
* Как часто стоит устраивать мозговые штурмы? (Михальцова А.Ю.)
* Почему важнее количество высказанных идей, чем их качество? (Михальцова А.Ю.)
* Существует ли разделение ролей при мозговом штурме? (Трубач Г.Г.)
* Какова продолжительность рабочего дня при мозговом штурме? (Трубач Г.Г.)
* Часто ли используется методика мозгового штурма? (Щавровский С.А.)
* Какие известные команды используют методику мозгового штурма? (Щавровский С.А.)
* Будете ли вы продвигать методику мозгового штурма в своем рабочем окружении? (Щавровский С.А.)
* Где мозговой штурм может быть применен? (Ярошевич Я.О.)
* Как предотвратить хаос во время мозгового штурма (вечные споры, ругательство и т.д.)? (Ярошевич Я.О.)
* Как из практик мозгового штурма наиболее эффективна? (Ярошевич Я.О.)

#### Доклад Григорьева

* Что мешает использовать свободное по с GPL, для своих приложений, вместо того, чтобы платить за "снятие лицензии"? (Как именно снимается лицензия) (Белый А.А.)
* Какие альтернативен свободные ОС есть для обеспечения независимости, а также удобства для пользователя (Белый А.А.)
* Какая типовая лицензия для Open Source приложений подходит для использования в коммерческих целях? (Борисевич П.И.)
* Что могут сделать нарушителю GPL? (Борисевич П.И.)
* Как выбирать лицензию для свободного софта? (Борисевич П.И.)
* В чём особенность MIT License? В чём её основные преимущества и недостатки в сравнении с GPL? (Гетьман С.И.)
* В чём особенности лицензирования GitHub проектов? (Гетьман С.И.)
* Как свободное ПО влияет на рынок и на зарплаты программистов, менеджеров?

(Гетьман С.И.)

* На какие хлеба живёт Ричард Столман? (Гетьман С.И.)
* Как Open Source / free лицензии борются с вредителями? (Гетьман С.И.)
* Насколько важен выбор лицензии для разработки ПО? Бывали ли случаи судебных тяжб из-за неправильно выбранной лицензии? (Гетьман С.И.)
* Какие существуют типы лицензий Open Source? (Ипатов А.Е.)
* На твой взгляд, есть ли перспектива развития направления Open Source? (Ипатов А.Е.)
* Как происходит тестирование Open Source проектов? (Ипатов А.Е.)
* Назови какие-нибудь популярные Open Source проекты. (Ипатов А.Е.)
* Существует ли свободное ПО, которые нарушает 4 свободы? (Лебедев Н.А.)
* В чем выгода для организации, которая работает под Open Source? (Лебедев Н.А.)
* Как сложилось, что образовались такие кластеры как Freeware, FreeSoft, ComSoft? (Лебедев Н.А.)
* Как заработать на рынке Open Source? (Лебедев Н.А.)
* На ваш взгляд, какое будущее у Open Source? (Лебедев Н.А.)
* Существуют другие лицензии, кроме GPL? Если есть, то почему нет единого стандарта? (Лебедев Н.А.)
* Какие проблемы с законом могут возникнуть при использовании Open Source?

(Михальцова А.Ю.)

* Любой ли желающий может использовать Open Source? (Михальцова А.Ю.)
* Считаешь ли ты эффективным использование Open Source и почему? (Михальцова А.Ю.)
* Возможно ли в дальнейшем переведение всего ПО на открытое? (Михальцова А.Ю.)
* Как именно владельцы свободного ПО с лицензией GPL обходят ее и продают возможность пользоваться своим продуктом, не "заражая" покупателей? (Ровдо Д.И.)
* Почему все вокруг так увлечены Open Source проектами? (Ровдо Д.И.)
* В чем смысл подхода владельцев софта с открытым исходным кодом, но запретом его использовать? (Ровдо Д.И.)
* Как в проектах отслеживается тип лицензии? (Трубач Г.Г.)
* Возможно ли в открытых приложениях использовать проприетарный софт и наоборот? (Трубач Г.Г.)
* Какие самые известные Open Source проекты существуют? (Трубач Г.Г.)
* Какая ответственность предполагается за использование Open Source проектов в коммерческих целях? (Трубач Г.Г.)
* Примеры свободного программного обеспечения? (Ярошевич Я.О.)
* Можно ли отойти от одного или нескольких пунктов из десяти пунктов в определении Open Source программного обеспечения, и всё равно считать его таковым? (Ярошевич Я.О.)
* Примеры Open Source программного обеспечения? (Ярошевич Я.О.)

### Менеджмент IT-проекта

#### Менеджмент IT-проектов

* Имеет ли место в практике, что один менеджер контролирует несколько проектов? (Белый А.А.)
* Какие существую меры контроля рисками? (Белый А.А.)
* Чем отличается работа менеджера на крупном проекте и на стартапе? (Белый А.А.)
* Какой человек может стать менеджером? (Белый А.А.)
* Можно ли рассчитывать на успех команды менеджмента проекта в новом проекте, если она хорошо справилась с предыдущим? (Борисевич П.И.)
* Как можно оценить деятельность команды менеджмента проекта? (Борисевич П.И.)
* Насколько глубокими должны быть технические знания менеджера в IT-проекте? (Григорьев А.В.)
* Может ли менеджер быть одновременно разработчиком в том же проекте? (Григорьев А.В.)
* Может ли в проекте быть одновременно несколько менеджеров? Если да, то могут ли пересекаться их сферы деятельности? (Григорьев А.В.)
* Чем отличаются полномочия менеджера от его полномочий его помощников? Где они пересекаются и в чем друг друга дополняют? (Грушевский А.А.)
* Должен ли менеджер знать некоторую техническую составляющую проекта? (Ипатов А.Е.)
* Может ли менеджер принимать решение об отказе от того или иного заказа? (Ипатов А.Е.)
* Может ли менеджер выполнять еще какие-либо роли или ему стоит сосредоточиться на своей роли? (Ипатов А.Е.)
* С помощью каких программных или аналитических средств менеджер строит бизнес-модель? (Лебедев Н.А.)
* Направление развития проекта определяется только бизнес-моделью менеджера или существуют другие определяющие факторы? (Лебедев Н.А.)
* Почему менеджер не учитывает тенденций рынка на этапе анализа? (Лебедев Н.А.)
* В чем заключается работа менеджера? (Михальцова А.Ю.)
* Разработка проекта считается непрерывным процессом? (Михальцова А.Ю.)
* Что включает check-лист? (Михальцова А.Ю.)
* Может ли заказчик выступать в качестве менеджера? (Михальцова А.Ю.)
* Какая группа успешнее выполняет проект, поставленную задачу? (Михальцова А.Ю.)
* Может ли менеджер устанавливать сроки проекта? (Михальцова А.Ю.)
* Менеджер не обязательно должен владеть техническими данными, но в его обязанности входит пункт разбиения задачи на подзадачи. Тогда получается, что у менеджера должны быть помощники? (Ровдо Д.И.)
* Какие инструменты влияния есть для обычных работников (не фрилансеров)? (Ровдо Д.И.)
* Должен ли менеджер интересоваться ситуацией и разговаривать с командой или достаточно действий командного лидера? (Ровдо Д.И.)
* Является ли проблема разногласий в команде проблемой менеджера? (Ровдо Д.И.)
* Каким образом определяются и строятся бизнес-модели проекта? (Трубач Г.Г.)
* Как происходит обсуждение вопросов менеджмента и бизнес-части проекта? (Трубач Г.Г.)
* Каковы особенности метода менеджмента проекта — делегирования? (Трубач Г.Г.)
* Чем характерен метод менеджмента — «разделяй и властвуй»? (Трубач Г.Г.)
* Полезно ли записывать (логирование) все события менеджмента (разногласия, соглашения, задачи и т.д.)? (Трубач Г.Г.)
* Почему среди перечисленных ресурсов не было ресурса времени? (Щавровский С.А.)
* Как происходит менеджмент проекта в open-source проектах? (Щавровский С.А.)
* Какие есть решения по автоматизации менеджмента проекта? (Щавровский С.А.)
* Какое образование должен иметь менеджер и его команда? (Ярошевич Я.О.)
* Кто несет наказание в случае нарушения сроков? Что произойдет, если не учтутся некоторые факторы и сроки затянутся? (Ярошевич Я.О.)
* Общается ли менеджер непосредственно с заказчиком? (Ярошевич Я.О.)
* Может ли менеджер работать удаленно? (Ярошевич Я.О.)

#### Роли в команде и их функции

* Кто распределяет роли в команде? (Белый А.А.)
* Кто такой проектный архитектор? (Белый А.А.)
* Могут ли совмещаться роли разработчика и QA? Применимо ли это на практике? (Белый А.А.)
* Как происходит повышение должности работника? (Белый А.А.)
* Почему нежелательно, чтобы командный лидер и менеджер проекта был одним человеком? (Борисевич П.И.)
* На каком этапе жизненного цикла распределяются роли в проекте? (Борисевич П.И.)
* Может ли заказчик сам назначать роли? (Гетьман С.И.)
* Сколько ролей оптимально использовать на проекте? (Гетьман С.И.)
* Сколько ролей может иметь один человек в команде? (Гетьман С.И.)
* С чем может быть связано изменение ролей? (Гетьман С.И.)
* Сколько ролей оптимально использовать на проекте? (Гетьман С.И.)
* В чем разница бизнес компетенции в кластере управление продуктом и кластере управления выпуском? (Гетьман С.И.)
* Кого лучше взять в команду: узкопрофильного или многопрофильного специалиста? (Гетьман С.И.)
* Может ли меняться роль работника на протяжении проекта? (Григорьев А.В.)
* Может ли один человек выполнять сразу несколько ролей? (Григорьев А.В.)
* Является ли выделение родственных функций в роли обязательным или оно носит формальный характер? (Григорьев А.В.)
* Зависит ли наличие определенных ролей от выбранной модели разработки? И от чего зависит наличие той или иной роли? (Григорьев А.В.)
* Кто занимается консультациями при разработке? (Грушевский А.А.)
* Кто помогает заказчику понимать документацию по отчетности разработки, технических возможностей проекта и т.д.? (Грушевский А.А.)
* Кому подчиняется архитектор? (Грушевский А.А.)
* Что входит в документацию пользователя? (Грушевский А.А.)
* На каком этапе происходит распределение ролей? (Ипатов А.Е.)
* На всех ли проектах у одной роли одинаковые функции? (Ипатов А.Е.)
* Какие роли являются ключевыми, а какие нет? (Ипатов А.Е.)
* Чем отличается кластер управление программой от кластера управление продуктом? (Лебедев Н.А.)
* Как командный лидер оценивает компетентность своей команды? (Лебедев Н.А.)
* Какие роли можно совмещать? (Лебедев Н.А.)
* Какая ситуация наиболее приемлема: сотрудник работает по конкретному плану или сотрудник сам определяет свои обязанности? (Лебедев Н.А.)
* Самые важные роли, с которыми можно начинать разработку? (Лебедев Н.А.)
* Кто распределяет роли в команде? (Михальцова А.Ю.)
* Четко ли соблюдается распределение ролей в команде или возможны совмещения? (Михальцова А.Ю.)
* По каким принципам распределяются роли в различных моделях жизненного цикла? (Михальцова А.Ю.)
* Участвует ли командный лидер в непосредственной разработке проекта? (Михальцова А.Ю.)
* Если командный лидер разъясняет требования заказчика разработчикам, то каковы обязанности бизнес аналитика? (Михальцова А.Ю.)
* Какие роли совмещать нежелательно? (Михальцова А.Ю.)
* Какая из ролей находится выше, а как ниже в структуре проекта? (Ровдо Д.И.)
* Некоторые роли имеют некоторые схожие функции. Получается, что некоторые функции пересекаются между разными ролями? (Ровдо Д.И.)
* Кто такой product owner? (Ровдо Д.И.)
* Как можно сопоставить качества человека и роли? (Щавровский С.А.)
* Возможно ли создать команду без разделения ролей? (Щавровский С.А.)
* Кто такие full-stack разработчики? (Щавровский С.А.)
* Какие роли лучше всего сочетать? (Щавровский С.А.)
* Кто занимается обучением сотрудников в IT-компаниях? (Ярошевич Я.О.)
* Какие навыки нужны для роли командного лидера? (Ярошевич Я.О.)
* Бывают ли такие проекты, на которых отсутствует QA-менеджер? (Ярошевич Я.О.)
* Какими ролями можно пренебречь? (Ярошевич Я.О.)

### Архитектура компьютера и мобильных устройств

#### Архитектура компьютера

* Какие существуют способы для ускорения передачи данных между ПЗУ (ОЗУ) и регистрами? (Белый А.А.)
* В чем существенное различие между поколениями ОЗУ (DDR2, DDR3, DDR4)? (Белый А.А.)
* Почему не используют закрытую архитектуру? (Гетьман С.И.)
* Какова максимальная пропускная способность системной шины? (Гетьман С.И.)
* Что означает разрядность процессора? (Григорьев А.В.)
* Может ли CD-диск выступать в качестве системного диска? (Григорьев А.В.)
* Существуют и используются ли компьютеры с закрытой архитектурой? (Григорьев А.В.)
* Все ли компьютеры используют BIOS? (Григорьев А.В.)
* В чем различия распределения ресурсов в одно- и мультипроцессорных системах? (Грушевский А.А.)
* Как разрешаются конфликты доступа к ресурсам мультипользовательских UNIX-системах? (Грушевский А.А.)
* Какие существуют способы расширения возможностей компьютера? (Ипатов А.Е.)
* Особенности cash-памяти и ее уровни (L1, L2 и т.д.)? (Ипатов А.Е.)
* В чем отличия системной и адресной шин? (Ипатов А.Е.)
* Какие существуют аналоги центральной системной шины? Почему остановились на использовании магистралей? (Лебедев Н.А.)
* Можно ли жесткий диск отнести к модулям? (Михальцова А.Ю.)
* От каких параметров зависит частота импульсов и что можно сделать, чтобы ее увеличить? (Михальцова А.Ю.)
* Без каких компонентов архитектуры компьютер может работать (запускаться)? (Ровдо Д.И.)
* В чем заключается разница между HDD и SSD дисками? (Трубач Г.Г.)
* Как производится обмен данными внутри компьютера? (Трубач Г.Г.)
* Как устроен и какие функции выполняет сопроцессор? (Трубач Г.Г.)
* Существуют ли процессоры с разрядностью большей чем х64? (Трубач Г.Г.)
* Что произойдет, если вставить в нужный разъем на материнской плате планку оперативной памяти большего объема, чем это допустимо платой? (Трубач Г.Г.)
* Какие бывают типы жестких дисков и в чем их различия? (Щавровский С.А.)
* В чем структурны различия между графическим процессором и центральным процессором? (Щавровский С.А.)
* Каковы структурные различия ядра UNIX-систем и GNU/Linux? (Щавровский С.А.)
* Какие есть примеры универсальных компьютерных шин? (Ярошевич Я.О.)
* А какие процессы выполняются, когда компьютер выключается? (Ярошевич Я.О.)

#### Операционные и вычислительные системы

* Какие ОС преимущественно ставят на суперкомпьютеры и почему? (Белый А.А.)
* Как понять, что ОС построена некорректно? (Гетьман С.И.)
* Как обеспечивается на аппаратном уровне возможность работы на одном компьютере с несколькими ОС? (Гетьман С.И.)
* В чем заключается ненадежность ОС Windows? (Гетьман С.И.)
* В чем заключаются недостатки ОС X? (Гетьман С.И.)
* Есть ли будущее у индивидуальных вычислительных систем? (Гетьман С.И.)
* В чем недостатки тех или иных ОС? (Ипатов А.Е.)
* Как ОС наиболее популярна? (Ипатов А.Е.)
* Какие ОС лучше использовать при решении определенных задач и почему? (Лебедев Н.А.)
* Что следует учитывать при разработке ОС? (Лебедев Н.А.)
* Какая ОС наиболее популярна? (Михальцова А.Ю.)
* Каковы преимущества и недостатки наиболее известных и используемых ОС? (Михальцова А.Ю.)
* Что относят к системному ПО? (Михальцова А.Ю.)
* Какая ОС используется в сложных системах, где требуется сильная и стабильная безопасность? (Ровдо Д.И.)
* Какие существуют нестандартные ОС? (Ровдо Д.И.)
* Какие ОС используются в управлении самолетами? (Трубач Г.Г.)
* В чем заключаются проблемы ОС Windows? (Трубач Г.Г.)
* Каков прогноз на разработку игр на свободные ОС? (Трубач Г.Г.)
* Какие ОС входят в ТОП-5 по различным показателям (количеству пользователей, дружественности GUI и т.д.)? (Ярошевич Я.О.)
* Какой есть самый простой путь, чтобы написать свою ОС? (Ярошевич Я.О.)

#### Мобильные операционные системы

* Какие главные недостатки мобильных операционных систем Android, IOS?

(Борисевич П.И.)

* Может ли мобильное устройство использовать несколько операционных систем? (Борисевич П.И.)
* Какая операционная система наименее энергозатратна? (Борисевич П.И.)
* Почему не делают общий магазин для различных мобильных ОС? (Гетьман С.И.)
* Почему Windows Phone не имеет такой популярности среди мобильных ОС, как Windows для ПК? (Гетьман С.И.)
* Что можно сказать о безопасности Android, iOS, а также их взлома (перепрошивки)? (Гетьман С.И.)
* На кого рассчитана Android? IOS? BlackBerry? Windows Phone? Sailfish? Ubuntu Touch? (Гетьман С.И.)
* Зачем нужны альтернативные магазины для Android? Одного мало? (Гетьман С.И.)
* Вероятно ли появление на рынке и обретение популярности мобильной ОС, созданной не гигантом вроде Microsoft, Google, Apple? (Гетьман С.И.)
* А создаются и используются ли ОС, которые обеспечивают максимально минималистичный интерфейс и функционал (чтобы сделать телефон прочным телефоном вроде Nokia 3310)? (Гетьман С.И.)
* Почему не производят мобильные устройства, на которые можно поставить и Windows Phone, IOS или Android? Как скоро нас ждут подобные решения? (Гетьман С.И.)
* Какие ещё существуют мобильные операционные системы и какие из них способны стать одними из лидеров? (Григорьев А.В.)
* Перспективно ли создание собственной мобильной операционной системы?

(Григорьев А.В.)

* Отличия и преимущество открытых и закрытых операционных систем. (Григорьев А.В.)
* Для какой из озвученных операционных систем легче/выгоднее создавать свои приложения? (Григорьев А.В.)
* Возможность и способы заработка на своих приложениях для мобильных операционных систем. (Григорьев А.В.)
* Какие существуют альтернативные разработчики ОС Android (кроме официальных)? (Грушевский А.А.)
* Какие есть отличия и нововведения в разных версиях мобильных ОС? (Грушевский А.А.)
* В каких еще сферах возможно внедрение той или иной мобильной ОС? (Ипатов А.Е.)
* С каждым годом IOS все больше и больше набирает обороты, а Android наоборот постепенно начинает уступать. С чем это связано? (Ипатов А.Е.)
* Есть ли смысл создания единой ОС, включающей все преимущества той или иной ОС? Возможно ли это вообще? (Ипатов А.Е.)
* Какие существуют глобальные проблемы современных OC помимо энергосбережения? (Ипатов А.Е.)
* За счет чего добиваются эффективного энергоснабжения? Чем приходится жертвовать? (Ипатов А.Е.)
* Что прежде всего стоит учесть при создании мобильного приложения? Есть ли отличия от разработки desktop приложений? (Лебедев Н.А.)
* В чем секрет успеха Android? (Лебедев Н.А.)
* Какие отличительные черты у каждой из мобильных ОС? (Лебедев Н.А.)
* Известно, что любая мобильная ОС часто обновляется. С чем это связано? (Лебедев Н.А.)
* Если ли отличия в качестве и производительности открытой от коммерческой мобильной ОС? (Лебедев Н.А.)
* Microsoft в свое время упустили мобильный рынок. Правда, что мобильный Windows — качественный продукт или это хороший маркетинг Microsoft? (Лебедев Н.А.)
* Сильно ли схожа функционально Мобильных ОС с ОС для ПК? (Михальцова А.Ю.)
* Если все Мобильные ОС функционально схожи, зачем тогда их такое большое количество? (Михальцова А.Ю.)
* Что можно предпринять для улучшения энергопотребления? (Михальцова А.Ю.)
* Бесплатна ли Мобильная ОС? (Михальцова А.Ю.)
* Удобна ли синхронизация (данных) мобильного телефона с ПК? (Михальцова А.Ю.)
* Преимущества Android для управления автомобилем. (Михальцова А.Ю.)
* Как ОС влияет на производительность мобильного устройства? (Трубач Г.Г.)
* В чем преимущество ОС IOS? Зачем люди скупают продукцию Apple, которая стоит неоправданно дорого? (Трубач Г.Г.)
* Что из себя представляет ОС Firefox? Это ОС, или какой-нибудь веб-браузер, работающий на движке Gecko? Если второе, то как работает ОС? (Трубач Г.Г.)
* Какая ОС самая (менее) энергосберегательная? (Трубач Г.Г.)
* Есть ли будущее у Windows Phone? (Щавровский С.А.)
* Почему у Android такой большой отрыв в количестве пользователей? (Щавровский С.А.)
* Почему Windows Phone не может завоевать рынок? (Щавровский С.А.)
* Что является основным критерием для вас при выборе мобильной ОС? (Щавровский С.А.)
* Каким образом мобильная ОС "понимает", что следует перейти в другой режим для энергосбережения? Какие есть режимы, и как это организовано? (Ярошевич Я.О.)
* Есть ли возможность эмуляции мобильной ОС? К примеру, специальный софт, который позволяет на устройстве c Android отображать ОС, подобную iOS? И для других аналогично? (Ярошевич Я.О.)
* Какую последовательность действий нужно совершить, чтобы выложить свое приложение в маркет на каждой из мобильных ОС, чтобы пользоваться им в дальнейшем?

(Ярошевич Я.О.)

* Есть такое наблюдение, что одни и те же приложения на разных ОС стоят по-разному. Как правило, на iOS дороже. С чем это связано? В пример можно привести приложение maps.me, которое является бесплатным на ОС Android, но платным на ОС iOS.

(Ярошевич Я.О.)

### Языки программирования

#### Языки программирования. Обзор языков программирования

* Чем конкретно "вреден" оператор goto по Дейкстре? (Белый А.А.)
* Какие основные принципы функциональных языков программирования? (Какие уникальные возможности есть?) (Белый А.А.)
* Какие тенденции развития языков программирования? (Что в дальнейшем будет популярнее: языки типа Java или языки типа Python?) (Белый А.А.)
* Для каких программ использовали первые языки программирования? (Борисевич П.И.)
* Какие есть известные языки программирования без обязательной типизации данных? (Борисевич П.И.)
* Какие устройства могли запускать программы на первых языках программирования? (Борисевич П.И.)
* Почему интерпретатор выполняет код программ медленнее компилятора? (Борисевич П.И.)
* Есть ли среди языков программирования "мёртвые"? Какие атрибуты попадают под это определение? (Гетьман С.И.)
* Возможно было бы определение и создание "структурного программирования" раньше, чем в конце 60-х с выходом статьи Дейкстры? (Гетьман С.И.)
* Какие языки лучше и удобнее для разработки: со строгой или динамической типизацией? А для учёбы? (Гетьман С.И.)
* Как происходили развитие и эволюция компиляторов и интерпретаторов? (Гетьман С.И.)
* Что должен иметь язык программирования, чтобы стать используемым и популярным для разработки? (Гетьман С.И.)
* В чём смысл создания таких экзотических языков как Brainfuck (код посредством рисунка из слешей), Cook (код состоит из одной инструкции, повторяющейся различное число раз в зависимости от вызова необходимой функции)? (Гетьман С.И.)
* Нужны ли языки программирования, в которых даже структур нет (я уже молчу об классах и основах ООП)? (Гетьман С.И.)
* Нужны ли визуальные языки программирования? В чём их преимущества? (Гетьман С.И.)
* Что влияет на рост/падение популярности языка программирования? (Григорьев А.В.)
* Перспективно ли создание собственно языка программирования? И что для этого необходимо? (Григорьев А.В.)
* Где именно применяются низкоуровневые языки программирования? (Григорьев А.В.)
* Что относят к "нормальным алгоритмам"? (Михальцова А.Ю.)
* Если использовать оператор goto вредно, почему тогда на языке Assembler он один из основных при условном переходе? (Михальцова А.Ю.)
* Какие языки программирования относят к функциональным? (Михальцова А.Ю.)
* Что из себя представляют функциональные языки программирования?

(Михальцова А.Ю.)

* Язык С++ к какому способу реализации относят? (Михальцова А.Ю.)
* Чем отличаются языки программирования низкого и высокого уровня?

(Михальцова А.Ю.)

* Какие языки программирования относят к языкам программирования высокого уровня? (Михальцова А.Ю.)
* Если Java такая "тяжелая", почему она так распространена? Ведь ее используют не только для программирования под Android, но и в веб-приложениях. (Ровдо Д.И.)
* В чем плюсы таких языков как Python и Ruby, раз на них переходят? (Ровдо Д.И.)
* С чем вообще связана популярность тех или иных языков программирования? Что случается такого, что заставляет людей переходить на другие? (Ровдо Д.И.)
* Какова зависимость языков программирования от ОС? (Ровдо Д.И.)
* Присутствует ли оператор goto в языке Java? Почему этот оператор нежелательно использовать? (Трубач Г.Г.)
* Каковы плюсы и минусы ООП? Чем плохи "висячие" методы в коде? (Трубач Г.Г.)
* Приложения под ОС Android пишутся только на Java или еще на каких-нибудь языках? (Трубач Г.Г.)
* Что из себя представляет язык программирования Swift? (Трубач Г.Г.)
* На каких языках программирования можно разрабатывать desktop-приложения?

(Трубач Г.Г.)

* Что означает встраиваемый язык программирования? (Щавровский С.А.)
* В какой сфере наиболее применимы функциональные языки программирования? (Щавровский С.А.)
* Чего на ваш взгляд не хватает современным языкам программирования?

(Щавровский С.А.)

* Есть такая байка: все бородатые программисты в свободное время пишут свой язык программирования (в частности так появился новый язык компании Apple «Swift»). Вопрос: собираетесь ли вы отращивать бороду и писать свой язык программирования? (Щавровский С.А.)
* Какие существуют языки программирования, написанные не на английском языке? (Команды на каком-либо другом языке) (Ярошевич Я.О.)
* Какие есть языки с динамической типизацией? (Ярошевич Я.О.)
* Почему программы, написанные на интерпретируемом языке, работают медленнее, чем программы, написанные на компилируемом языке? (Ярошевич Я.О.)

#### JavaScript

* В чем заключается неработоспособность функции isNan? (Белый А.А.)
* Что является самой интересной возможностью сейчас в JavaScript, и в будущих версиях, что обещают разработчики? (Белый А.А.)
* Что может вызвать утечку памяти в JavaScript? (Борисевич П.И.)
* Какие есть преимущества и недостатки TypeScript в сравнении с JavaScript, или кроме типизации никаких отличий между ними нет? (Борисевич П.И.)
* Является ли JavaScript удобным языком в разработке приложений для мобильных устройств? (Борисевич П.И.)
* В каком состоянии CEnvy на 2015 год? И кем он и для чего использовался? (Гетьман С.И.)
* Как осуществляется связь между JavaScript и основным языком разработки приложения (к примеру, C++)? (Гетьман С.И.)
* Лучше ли CoffeeScript чем JavaScript? (Гетьман С.И.)
* Как обходить null в JavaScript? (Гетьман С.И.)
* Как с удобством проводить отладку JavaScript кода без помощи браузера? А как отлаживать библиотеки JavaScript, вроде jQuery? (Гетьман С.И.)
* Используется ли JavaScript в сферах кроме web-разработки? (Григорьев А.В.)
* С чем связан рост популярности JavaScript? (Григорьев А.В.)
* Какие конкуренты существуют у JavaScript? (Григорьев А.В.)
* Существуют ли языки альтернативные JavaScript? (Ипатов А.Е.)
* Какой главный недостаток и главный существенный плюс JavaScript? (Ипатов А.Е.)
* Каковы дальнейшие перспективы развития данного языка? Не превратится ли он в "мертвый" язык? (Ипатов А.Е.)
* Говорят, что JavaScript очень похож на Python, Ruby, но все же он сам по себе. В чем его уникальность? (Ипатов А.Е.)
* Какие недостатки есть у JavaScript? (Лебедев Н.А.)
* Как устроен интерпретатор JavaScript? (Лебедев Н.А.)
* Лучшая IDE для разработки на JavaScript? (Лебедев Н.А.)
* Эффективно было бы сделать JavaScript типизированным? (Михальцова А.Ю.)
* Какой сценарный язык может конкурировать с JS при разработке веб-приложений на стороне клиента? (Михальцова А.Ю.)
* Какие библиотеки JavaScript наиболее распространены? (Михальцова А.Ю.)
* Что представляет собой отладчик в JavaScript? (Михальцова А.Ю.)
* Как происходит тестирование страниц, написанных на JavaScript? (Михальцова А.Ю.)
* В каких случаях имеет смысл писать сервер на JavaScript, а, например, не на Java?

(Ровдо Д.И.)

* Какой самый популярный JavaScript фреймворк и почему? (Ровдо Д.И.)
* В чем преимущество нетипизированного языка программирования, как JavaScript? Всего лишь в том, что программист может не думать о типах? То есть, для программистов, которые не хотят слишком много думать? (Ровдо Д.И.)
* Существует ли возможность разработки desktop-приложений на языке JavaScript?

(Трубач Г.Г.)

* Как объяснить следующее "3" -+-+-+ "1" + "1" / "3" \* "6" + "2" = "42"? (Трубач Г.Г.)
* Возможна ли разработка приложения на JavaScript в связке с C++? (Трубач Г.Г.)

### Разработка программного обеспечения

#### Парадигмы программирования

#### Стили программирования

#### Чистый код и его оптимизация

#### Паттерны проектирования

#### Разработка мобильных приложений

#### Тестирование

### Сложные науки и технологии

#### Трансляторы

#### Искусственный интеллект

#### Виртуальная реальность

#### Нейронные сети

#### Облачные вычисления

### Компьютерная и информационная безопасность