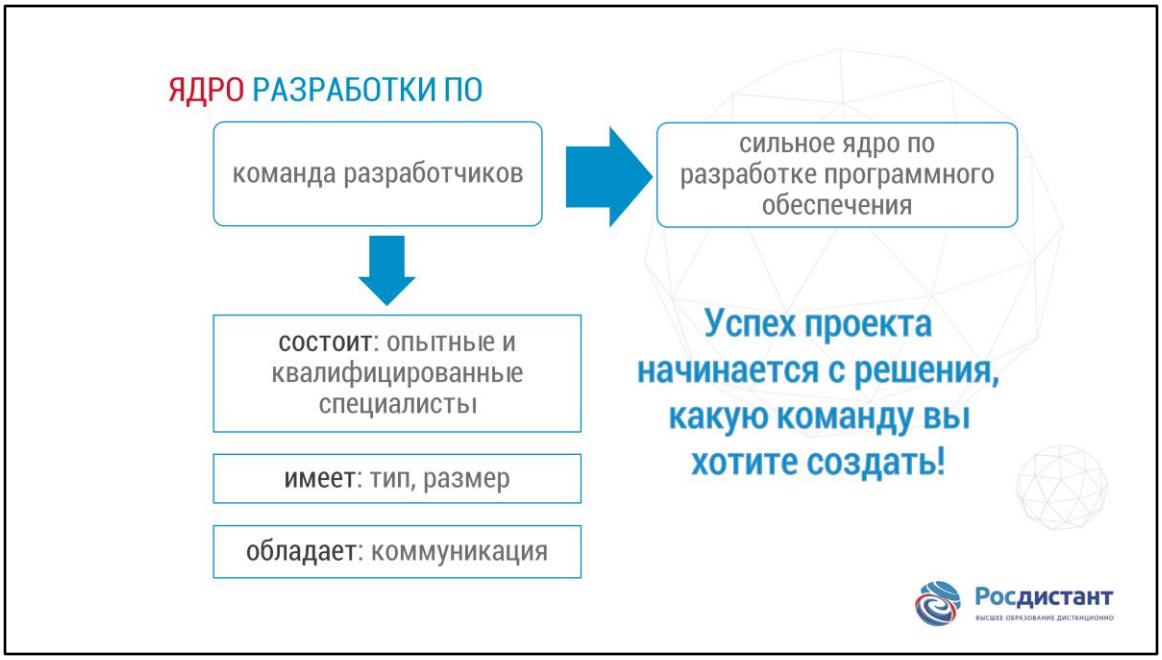

Росдистант
ВЫСШЕЕ ОБРАЗОВАНИЕ ДИСТАНЦИОННО
ВВЕДЕНИЕ В
ПРОГРАММНУЮ
ИНЖЕНЕРИЮ
6 ЧАСТЬ

Тема 6. Управление командной разработкой программного обеспечения
В ходе изучения данной темы вы узнаете, какие типы команд могут быть при разработке программного обеспечения. У вас сформируется представление о ключевых ролях командной разработки программных продуктов. Вы узнаете о функциональных обязанностях каждой роли и ее значении в общей работе над ИТ-проектом, а также какими навыками должен обладать специалист, выполняющий ту или иную роль в команде.



Начиная новый проект по разработке программного обеспечения, естественно, мы все стремимся к его успеху. Но чтобы добиться его, вам нужно полагаться на сильное ядро – вашу команду разработчиков программного обеспечения.

Все команды различаются по стилю работы и уникальной экосистеме внутри, но при этом у них есть некоторые довольно общие элементы. Мы все ожидаем, что наша команда будет состоять из очень опытных и квалифицированных специалистов. Но разве это единственное условие? Чтобы сделать группу людей разных профессий действительно эффективной командой разработчиков программного обеспечения, вам необходимо помнить о некоторых элементах, которые следует учитывать: тип, размер и коммуникация.

Очень важным в вопросе формирования команды является общение. Это последний кусок головоломки, который очень часто становится причиной успеха или неудач команды разработчиков программного обеспечения. Вы можете иметь все остальные элементы в рабочем состоянии и должным образом функционирующими, но если вашей команде не хватает последнего элемента – общения, это может нанести серьезный вред вашему продукту и всему процессу.

Хорошие коммуникативные навыки – важный мягкий навык для любого программиста. Но вы должны воспринимать их способность к общению как

данность. Ваша задача – развивать и поощрять общение внутри вашей команды разработчиков программного обеспечения. Нужно сделать его частью вашего обычного процесса, практикуя ежедневные стендапы, проверки дизайна и кода, написание документации, презентации, а также некоторые общественные мероприятия.

Успех проекта начинается с решения, какую команду вы хотите создать. Должна ли она состоять из специалистов, обладающих глубокими знаниями в своем предмете, быть разносторонней группой специалистов широкого профиля, обладающих определенными знаниями в различных областях, или быть их сочетанием. Затем вам нужно определить размер вашей команды, учитывая, что влечет за собой ваш выбор. После того как ваша команда будет организована, определите роли в команде разработчиков программного обеспечения и то, что ожидается от каждой из них. Но помните, что успешная команда – это та, которая нашла четкие каналы коммуникации и эффективный стиль работы.

Создание четкой структуры команды разработчиков программного обеспечения – важный первый шаг к общему успеху вашего проекта.

Здесь вам предстоит выбрать один из трех основных типов.

- Универсалы – так сказать, мастера на все руки. Универсалы обладают широким спектром знаний и опыта. Как правило, эти типы команд предназначены для обработки комплексных решений. Преимущество универсалов в том, что они могут предоставить полное решение проблемы. Однако у них также есть некоторые недостатки – если ваш проект требует более высокого уровня знаний в какой-либо области, универсалы окажутся в затруднительном положении, поскольку им может не хватать знаний и навыков.
- Специалисты – этот тип команд состоит из членов, обладающих высокой квалификацией в определенной области. Преимущество специалистов очевидно – они могут решить конкретный вопрос со всеми своими знаниями и опытом, что приведет к более эффективной и результативной работе. С другой стороны, общение – не самая сильная сторона команды разработчиков программного обеспечения такого типа. Часто, будучи очень узкими специалистами, члены команды могут не иметь общего понимания ролей других членов команды, что делает общение между ними несколько

неэффективным.

- Гибридная команда – этот тип команды представляет собой смесь двух предыдущих. Кажется, что этот подход сочетает в себе лучшее из двух миров, где специалисты сосредотачиваются на функциональных частях, а универсалы несут ответственность за общение и сотрудничество внутри команды. Однако эта мечта о команде сопровождается ограничением – обычно команду разработчиков программного обеспечения такого типа сложнее собрать с точки зрения временны́х и финансовых ресурсов.

В идеале нужно стремиться к сбалансированности универсалов и специалистов в команде для достижения лучших результатов.

СРАВНЕНИЕ ПОДХОДОВ К СТРУКТУРЕ КОМАНДЫ

Тип подхода	+	-
Универсалы	Хорошо разбирается в продукте Компетентен	Необходимость привлечь нового члена команды в середине проекта
Специалисты	Глубокое знание каждого элемента проекта Быстро построить сложные высококачественные системы	Индивидуальность Возможны пробелы в общении из-за отсутствия общих знаний
Гибридная команда	Есть специалисты и универсалы Процесс разработки максимально эффективен	Сложность скоординировать людей Требует много времени и очень дорого

Росдистант
ВЫСШЕЕ ОБРАЗОВАНИЕ ДИСТАНЦИОННО

Сравним три подхода к структуре продуктовой команды, выделив плюсы и минусы.

Плюсы универсального подхода

- Каждый член команды хорошо разбирается в продукте, поэтому может сосредоточиться на его улучшении в целом.
- Каждый человек достаточно компетентен, чтобы выполнять свою работу без зависимости от других.

Минусы универсального подхода

- Поскольку никто не обладает конкретными знаниями, иногда бывает необходимо привлечь нового члена команды в середине проекта.

Плюсы подхода специалиста

- Глубокое знание каждого элемента проекта.
- Команда может очень быстро построить сложные высококачественные системы.

Минусы подхода специалиста

- Поскольку каждый работает индивидуально, есть вероятность, что компоненты не подходят с первых итераций.
- Возможны пробелы в общении из-за отсутствия общих знаний.

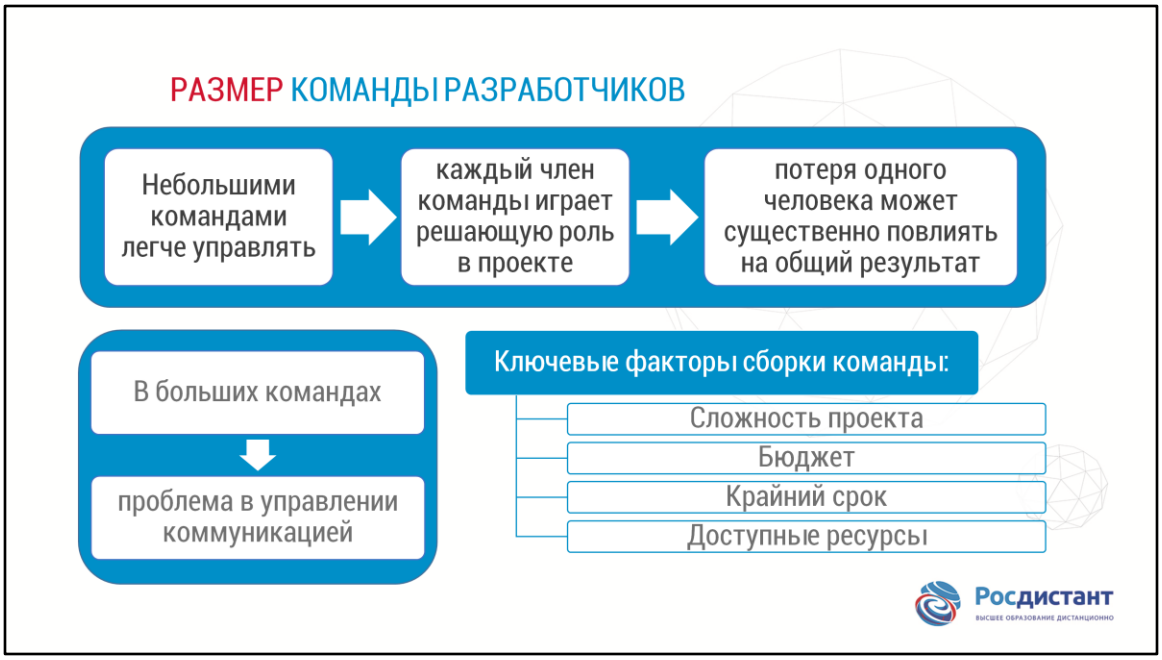
Плюсы гибридного подхода

- Есть как специалисты, которые создают отдельные компоненты, так и универсалы, которые следят за интеграцией системы.
- Процесс разработки максимально эффективен.

Минусы гибридного подхода

- Может быть сложно скоординировать людей с разными подходами к рабочему процессу.
- Создание гибридной команды требует много времени и очень дорого.

В идеальном мире каждый хотел бы иметь в штате специалистов разных профилей. Но реальность такова, что у каждого бизнеса есть ограничения – время и бюджет. Вот почему большинство команд разработки программного обеспечения являются универсальными.



Следующий шаг в формировании команды разработчиков программного обеспечения – это определение ее размеров.

Небольшими командами легче управлять, с одной стороны, но в этом случае каждый член команды играет решающую роль в проекте, и потеря даже одного человека может существенно повлиять на общий результат. В больших командах проблема заключается в управлении коммуникацией.

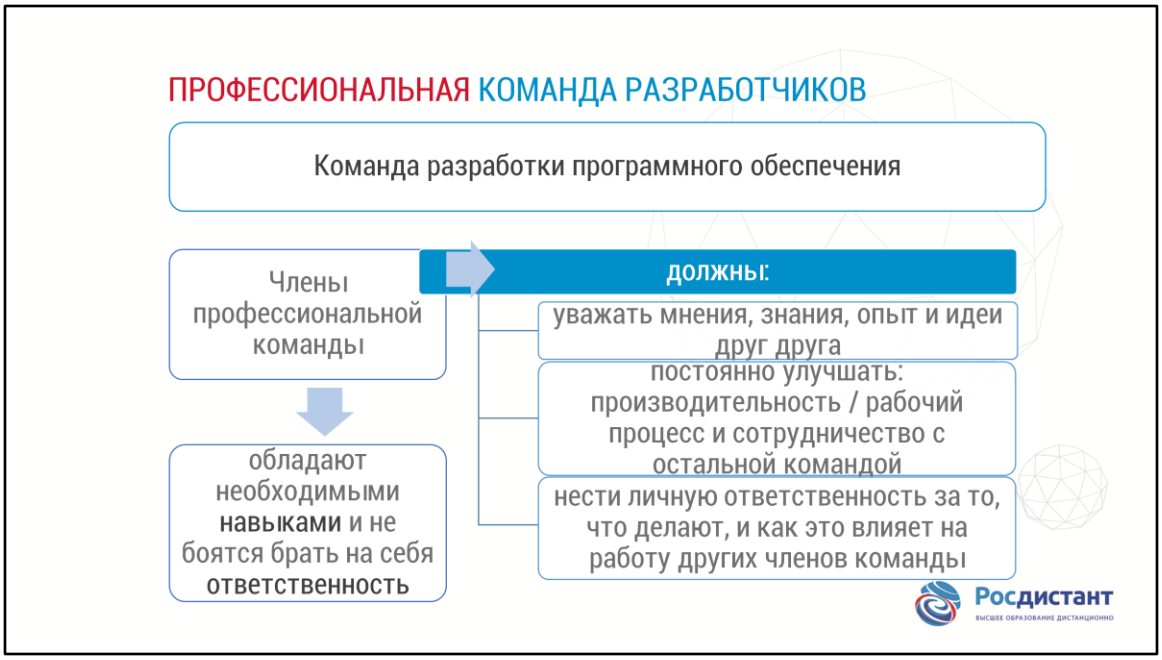
Но когда дело доходит до сборки команды, все сводится к следующим ключевым факторам:

- сложность вашего проекта;
- бюджет;
- крайний срок;
- доступные ресурсы.

На основе этих важных элементов вы можете решить, какой размер команды вам подходит.

Согласно методологии SCRUM оптимальный размер команды составляет от 3 до 9 человек, из которых 7 являются наиболее подходящими. Но это не значит, что вы должны строго следовать этому правилу. Если вашему программному проекту требуется большая команда, это не значит, что у вас обязательно возникнут проблемы с управлением или установлением связи. Ключевым

моментом здесь является тщательное управление вашей командой в соответствии с требованиями вашего проекта.



И наконец, надо установить четкие роли и цели в команде разработчиков. Кажется очевидным – роли внутри вашей команды ясны. Есть дизайнеры, разработчики и, наверное, тестировщик, и другие. На самом деле роли эффективной команды разработчиков программного обеспечения более разнообразны и сложны.

Команды разработки программного обеспечения – это не просто разработчики и технический директор. Их можно определить как тесные узлы различных навыков, критически важных для данной фазы проекта, которыми обладают разные специалисты.

Каждый человек в ИТ-проекте играет определенную важную роль, чтобы обеспечить максимальную производительность.

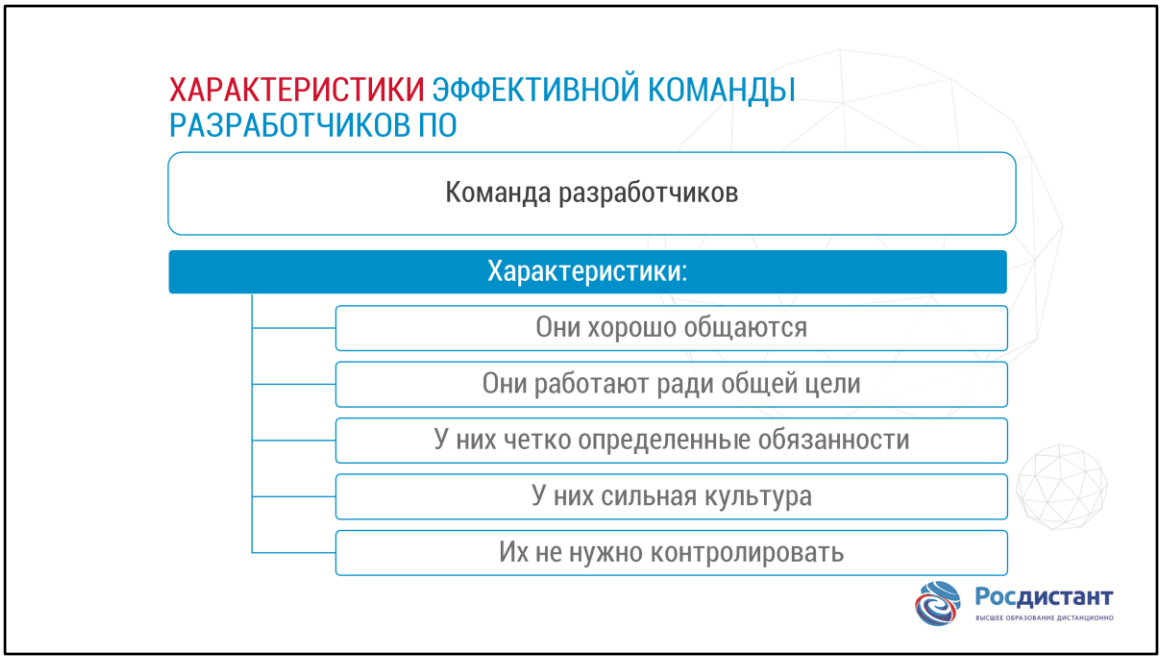
Безупречных проектов не бывает, но с правильной командой людей «на борту» проекты могут быть максимально приближены к совершенству, проходить гладко и в соответствии с планом.

Самое главное, люди в профессиональной команде разработчиков программного обеспечения обладают необходимыми **навыками** и не боятся брать на себя **ответственность**. Также важно, чтобы они:

- уважали мнения, знания, опыт и идеи друг друга;
- смело бросали вызов тому, как работать, чтобы постоянно улучшать:

производительность / рабочий процесс и сотрудничество с остальной командой.

- А также несли личную ответственность за то, что они делают, и за то, как это влияет на работу других членов команды.



Команда разработчиков – это группа штатных или преданных разработчиков, которые вместе работают над проектом. Они работают над продуктом в тесном сотрудничестве.

Самая большая задача каждой организации – обеспечить, чтобы их люди были мотивированы работать с максимальной отдачей. И хотя практически каждый может найти квалифицированных сотрудников, не каждой организации удается создать благоприятную среду для совместной работы, позволяющую процветать своим командам.

- Эффективная команда разработчиков программного обеспечения обладает следующими характеристиками.
- Они хорошо общаются. Коммуникация всегда лежит в основе командной работы независимо от отрасли, и разработка программного обеспечения не исключение. В отличных командах у людей есть все необходимые инструменты и процессы для регулярного здорового общения.
- Они работают ради общей цели. Хорошие команды не нуждаются в строгом управлении сверху вниз. У них четкие цели и общая миссия. В такой среде успех команды воспринимается как успех каждого человека.
- У них четко определенные обязанности. В то время как команда разделяет общую цель, каждый человек точно знает, что ему нужно сделать, чтобы все

это работало. Ожидания, роли и зоны ответственности определены с самого начала, и люди несут ответственность друг за друга за достижение прогресса.

- У них сильная культура. Наличие сильной культуры означает развитие профессиональных связей, поддержку и уважение друг к другу, а также чувство комфорта в компании друг друга. Такие команды с удовольствием проводят время вместе как на работе, так и вне офиса.
- Их не нужно контролировать. Управление сверху вниз постепенно уходит в прошлое, потому что великие команды с общими целями и общим видением не нуждаются в продвижении. Они хорошо выполняют свою работу, потому что хотят, а не потому, что их заставляют.

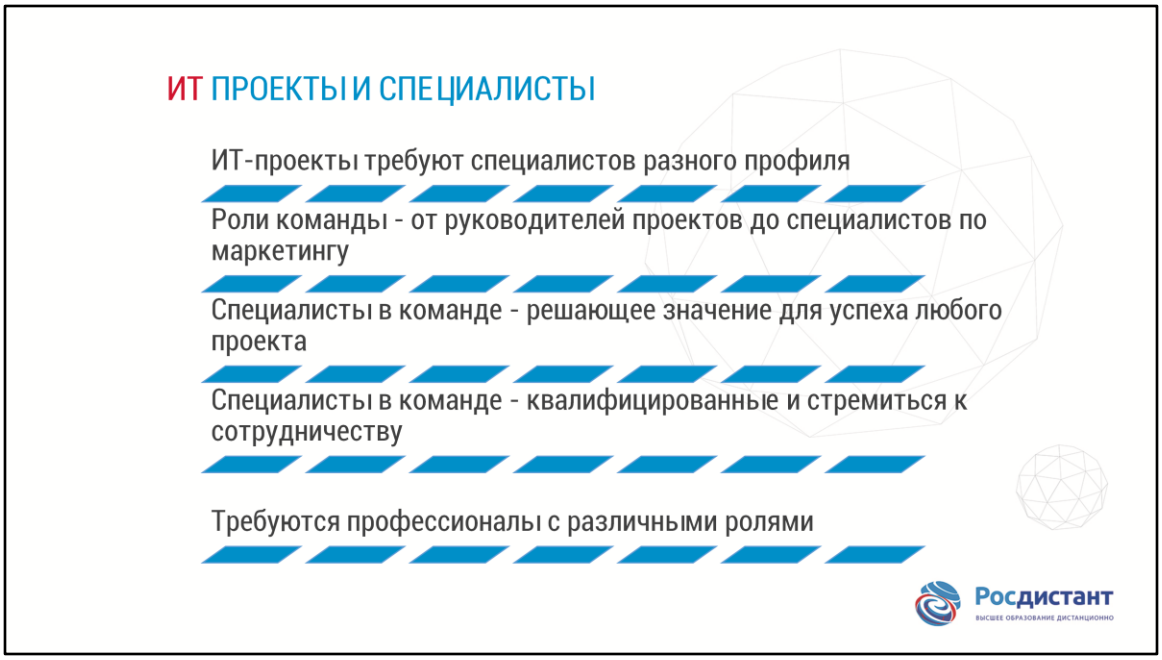
В общем, недостаточно нанять людей и разместить их в одной комнате. Работа в команде сложна. Это требует знания структуры команды разработчиков программного обеспечения и определенного понимания того, как осуществляется разработка программного обеспечения.

Общая структура команды разработчиков включает следующие роли.

- Владелец продукта – это клиент с видением того, как должен выглядеть конечный продукт, кто такие конечные пользователи и что он должен делать.
- Менеджер проекта – это человек, ответственный за управление и руководство всей командой. Его роль заключается в эффективной оптимизации работы команды, обеспечении соответствия продукта требованиям и определении целей для команды.
- Архитектор программного обеспечения – это высококвалифицированный разработчик программного обеспечения, который должен продумать все аспекты проекта и отвечает за выбор дизайна высокого уровня, а также за выбор технических стандартов. Например, определяет стек технологий для использования.
- Разработчики или инженеры-программисты – это члены команды, которые применяют свои знания в области инженерии и языков программирования при разработке программного обеспечения.
- Опытные дизайнеры гарантируют, что продукт будет простым и приятным в использовании. Они проводят интервью с пользователями, исследуют рынок и разрабатывают продукт с учетом потребностей конечных пользователей.
- QA или тестировщик несет ответственность за обеспечение качества и следит

за тем, чтобы продукт был готов к использованию.

- Бизнес-аналитик – ищет способы улучшить продукт. Он взаимодействует с заинтересованными сторонами, чтобы понять их проблемы и потребности, а затем документирует и анализирует их, чтобы найти решение.



Образ людей, работающих в сфере информационных технологий, изменился за последние годы. Смешные стереотипы о программистах были сломаны, и в настоящее время программирование считается крутым, так же, как и сами программисты стали элитой.

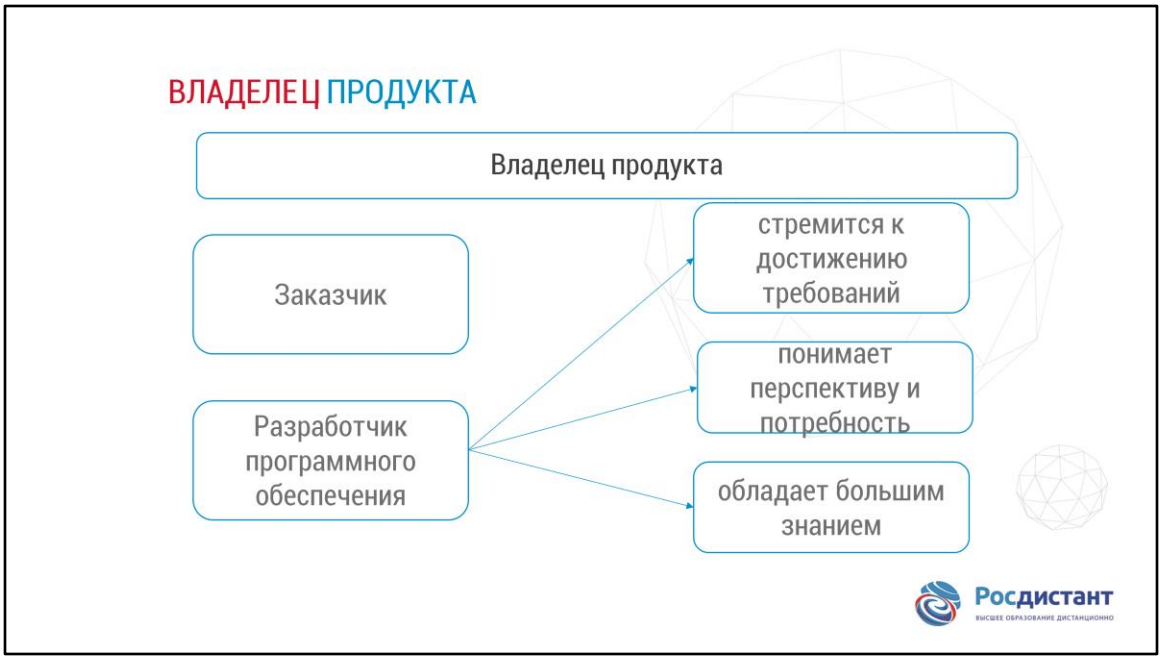
Однако, когда мы говорим об ответственности группы разработчиков, то речь идет не только о программистах, которые пишут код с использованием множества технологий и языков программирования. ИТ-проекты требуют специалистов разного профиля, не только с техническим образованием. Роли команды разработчиков варьируются от руководителей проектов до специалистов по маркетингу.

Программные проекты сегодня используются в разных отраслях, таких как здравоохранение, логистика, электронное обучение и так далее. Для таких проектов требуются разные подходы, разные инструментарии, но основные аспекты процесса разработки приложений обычно остаются неизменными.

Специалисты в команде разработчиков программного обеспечения имеют решающее значение для успеха любого проекта. Они должны быть квалифицированными, разносторонними и стремиться к сотрудничеству, обмениваться знаниями и видениями.

Поэтому для реализации ИТ-проектов требуются профессионалы с различными

ролями, которые работают над достижением наилучшего интеллектуального решения, отвечающего целям и требованиям клиента.
Мы должны разобраться с основными ключевыми ролями в команде разработчиков программного обеспечения. При этом необходимо четко понимать, кого и почему мы должны иметь в команде, чтобы создать хороший проект. Мы должны понимать, какие роли в проекте могут быть совмещены, а какие роли требуют особых умений и навыков.



Первая роль, которую мы рассмотрим, – это владелец продукта. Это лицо, принимающее окончательные решения. Обычно им является заказчик, как ключевой участник проекта.

Владелец продукта – это роль разработчика программного обеспечения, который представляет бизнес или конечных пользователей и отвечает за работу с группой пользователей, чтобы определить, какие функции будут в выпуске продукта.

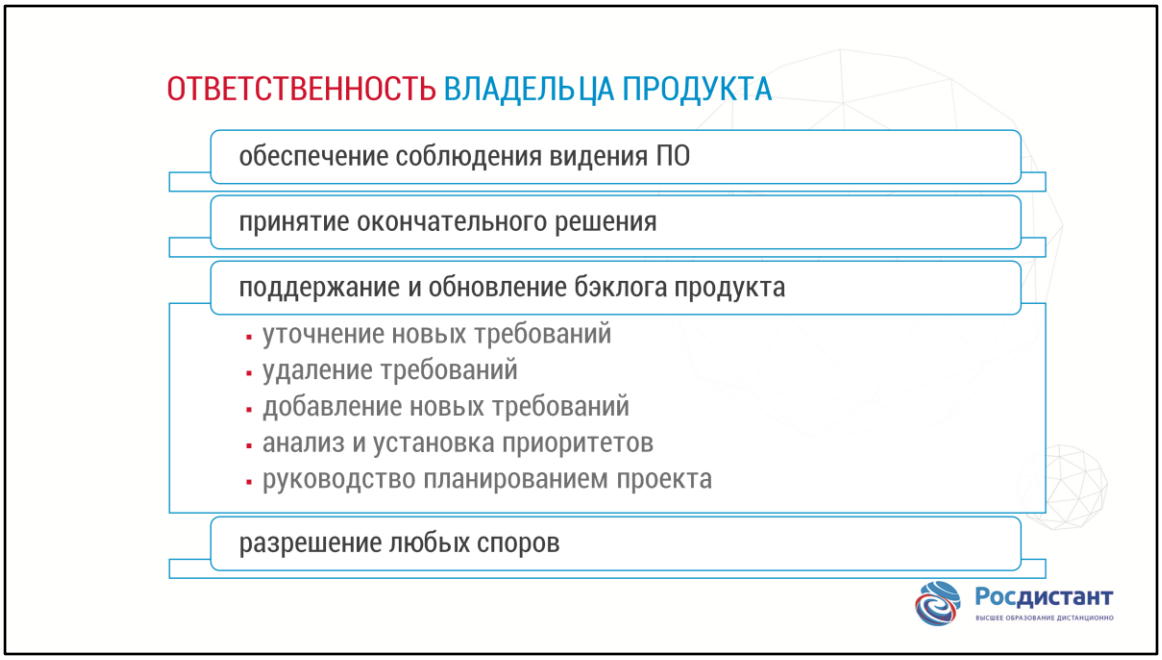
Владелец продукта – это тот, кто глубоко знает пользователя и продукт и отвечает за внутреннюю сторону разработки. Его задача – убедиться, что конечный продукт / услуга соответствует потребностям клиента. Владелец продукта следит за командой, поддерживает и координирует ее работу, а также обеспечивает выполнение всех требований к продукту.

Владелец продукта в команде разработчиков программного обеспечения:

- обладает бóльшим знанием проекта и пользователя;
- понимает перспективу и потребность клиента;
- стремится к достижению видения и требований в конечном продукте / услуге.

Заказчик должен быть гибким, творческим, усердным, а также обладать аналитическими способностями, поскольку его решения должны основываться на текущем бизнес-анализе и следовании рыночным тенденциям. Заказчик

делает всё, чтобы помочь команде разработчиков выполнить требования в быстро меняющейся среде.



Владелец продукта обычно сосредоточен на предоставлении максимально возможной ценности. Успех проекта находится в его руках. Он полагается на приоритеты, установленные на основе хорошего понимания бизнес-целей и потребностей клиентов.

Владелец продукта также несет ответственность за приоритетное отставание и максимизацию рентабельности инвестиций в программный проект. В обязанности этой роли входит документирование пользовательских историй или требований для программного проекта.

Он выступает в качестве основного контактного лица для всех решений, касающихся проекта. Он должен быть уполномочен выполнять свои обязанности без необходимости запрашивать предварительное разрешение от спонсоров проекта.

Назначение на эту роль подходящего человека с соответствующими делегированными полномочиями для продвижения проекта имеет фундаментальное значение для успеха проекта, особенно если применяется гибкий методологический подход.

В частности, владелец продукта несет ответственность за следующее.

- Обеспечение того, чтобы заявление о видении программного продукта соблюдалось.

- Принятие окончательного решения по всем вопросам, связанным с объемом работ.
- Поддержание и обновление бэклога продукта на постоянной основе.
- Уточнение новых требований.
- Удаление требований, выходящих за рамки.
- Добавление новых требований, определенных как необходимые для достижения заявления о видении программного продукта.
- Анализ и установка приоритетов.
- Руководство всеми встречами по планированию проекта.
- Разрешение любых споров либо с командой разработчиков программного обеспечения, либо внутри компании.

Отсутствие владельца продукта обычно означает, что программный проект будет выполняться урывками, в то время как разработчики программного обеспечения ждут важных отзывов.

Менеджер проекта PM – это человек, который отвечает за планирование и исполнение. PM несет ответственность за выполнение работы. Он также заботится о построении отношений между клиентом и различными подразделениями организации. Менеджеры проекта контролируют все процессы, делегируют задачи другим членам команды и следят за тем, чтобы все не сбились с пути.

Менеджер проекта отвечает за знание того, «кто, что, где, когда и почему» составляет проект программного обеспечения. Это означает знание заинтересованных сторон проекта и возможность эффективно общаться с каждой из них.

Менеджер проекта также отвечает за создание и управление бюджетом и графиком проекта, а также за процессы, включая управление объемом, проблемами и рисками.

Основные обязанности менеджера проекта могут включать следующее.

- Разработка плана программного проекта;
- управление результатами в соответствии с планом проекта программного обеспечения;
- подбор персонала для программных проектов;
- руководство и управление командой разработчиков программного

обеспечения.

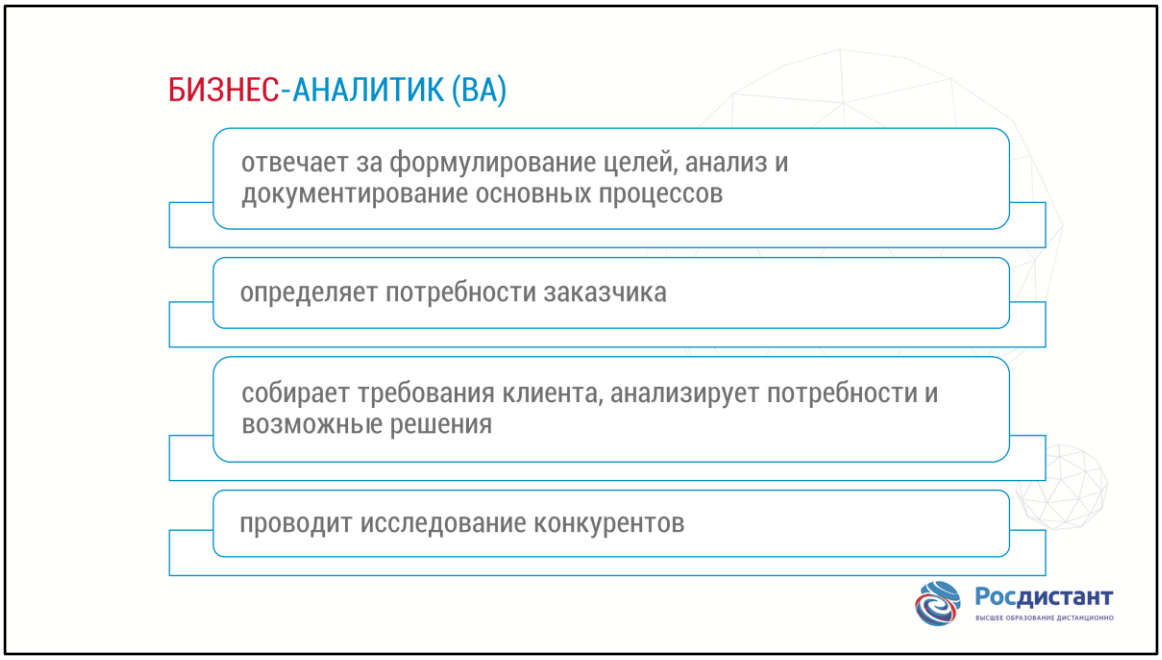
- А также определение методологии, используемой в проекте;
- составление графика проекта и определение каждой фазы;
- назначение задач членам проектной команды;
- предоставление регулярных обновлений высшему руководству.

Менеджер проекта также наблюдает за тестированием программного обеспечения, доставкой и официальной приемкой заказчиком. Затем менеджер проекта проводит его обзор с командой разработчиков программного обеспечения, чтобы задокументировать всё в процессе разработки программного обеспечения.

Свод знаний по управлению программными проектами PMBOK определяет пять методов управления командой проекта.

1. Управление конфликтами. В проектной среде, где люди должны работать вместе в условиях ограниченного расписания и бюджетных ограничений, лимитированных ресурсов и приоритетов заинтересованных сторон, конфликты неизбежны. В результате разрешение конфликтов может быть одной из самых сильных сторон руководителя проекта.
2. Принятие решений. Менеджер проекта должен принимать решения, чтобы продвигать команду и проект вперед. Это требует постоянного внимания к конечным целям проекта с учетом рисков, творческого подхода команды и факторов окружающей среды.
3. Эмоциональный интеллект. Важно понимать эмоции членов проектной группы, чтобы удовлетворить их потребности и одновременно гарантировать достижение целей проекта.
4. Влияние. Менеджер проекта должен оказывать влияние на команду проекта, организацию и других участников проекта, чтобы гарантировать, что все будут удовлетворены результатом проекта.
5. Лидерство. Менеджер проекта должен быть лидером проекта. Это означает, что он должен вдохновлять людей следовать своему видению проекта и

заручаться их поддержкой. Успех проекта требует сильного руководства. Управление командой проекта – это навык, которым менеджеры проектов пользуются каждый день. Вот почему успех проекта требует сильного управления командой проекта, и необходимо усвоить все эти ключевые моменты.



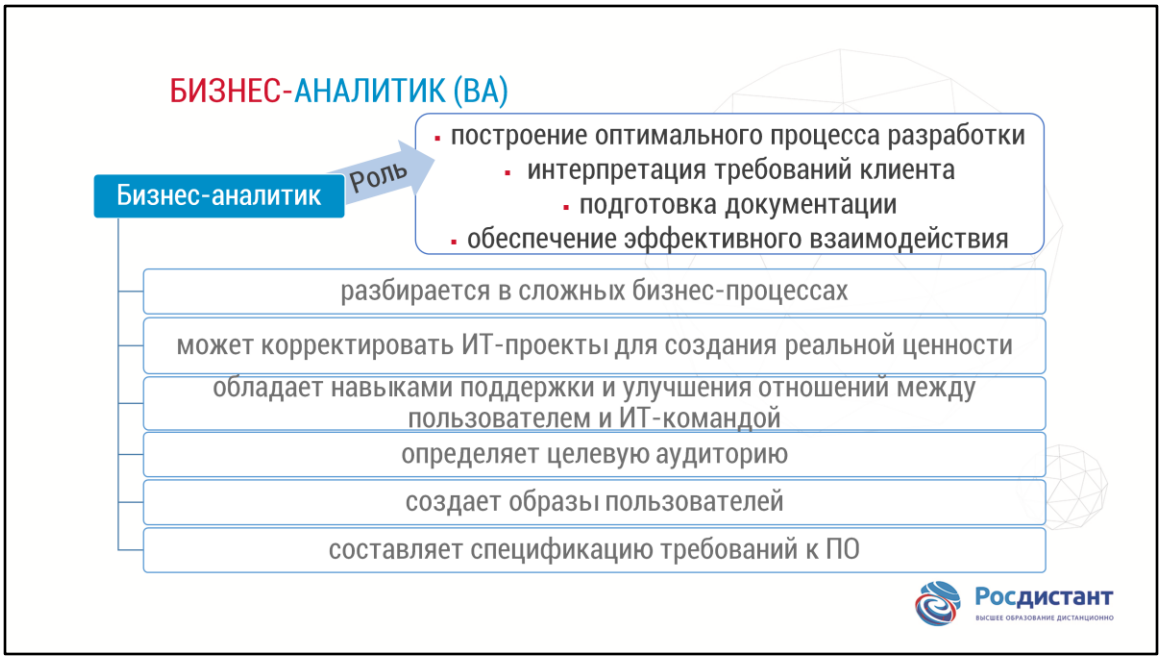
Бизнес-аналитик BA – это тот, кто отвечает за формулирование целей, анализ и документирование основных процессов и систем, а также за обеспечение согласования бизнес-модели и технологий. Он оценивает, что работает, а что нет, и задает направление развития бизнеса.

Бизнес-аналитики несут ответственность за преобразование бизнес-потребностей в требования. Они помогают компаниям формулировать бизнес-цели и помогают в определении требований на этапах технико-экономического обоснования, иногда даже до того, как будет собрана вся команда разработчиков.

Бизнес-аналитики больше всего нужны, когда у владельца продукта недостаточно времени для написания требований. Они создают документацию, сравнивая ее с существующими политиками и процедурами / протоколами для соответствия требованиям качества. Они также собирают команду разработчиков программного обеспечения, чтобы выработать лучшую стратегию, основанную на потребностях заинтересованных сторон.

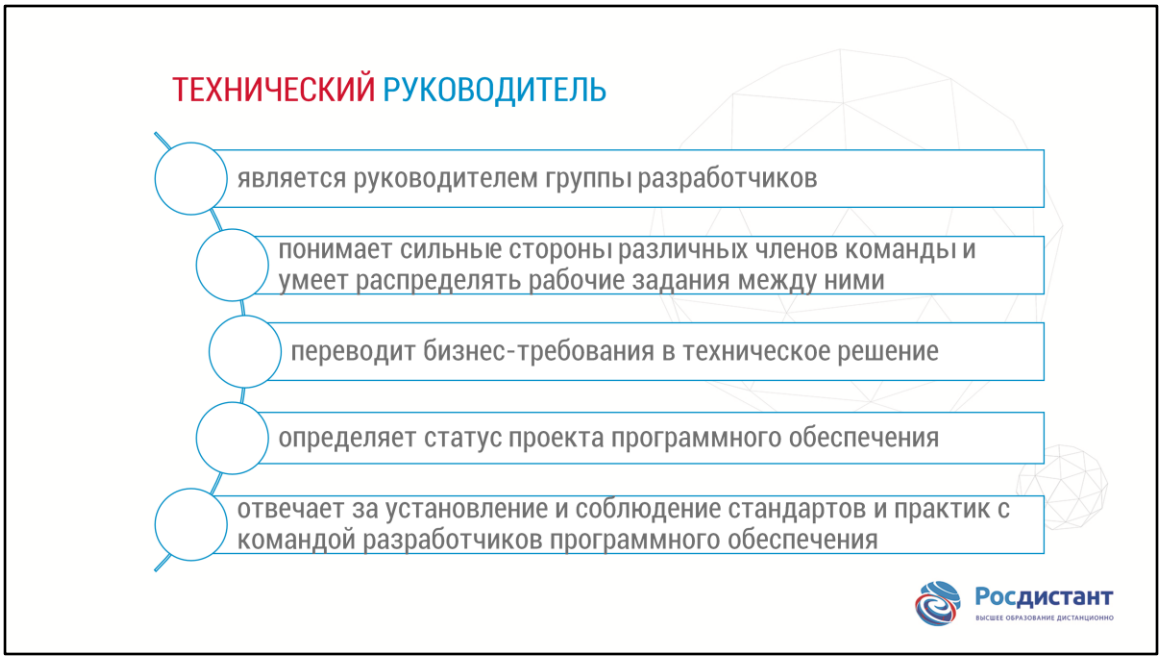
Среди ролей в группе гибкой разработки бизнес-аналитики являются одними из самых важных. Бизнес-аналитики подключаются к процессу разработки проекта на самом первом этапе, сразу после продажи, а иногда и раньше. Основная обязанность бизнес-аналитика заключается в общении как с клиентом, так и с

выделенной командой разработчиков.
Клиенты обычно думают о своих целях. Программисты думают в терминах функций и строк кода, сосредотачиваясь на том, как делать то, что хочет клиент. Существует разрыв между фактическими функциями и элементами, которые необходимо реализовать, и целью клиента.
Бизнес-аналитик должен точно определить, что хочет и в чем нуждается заказчик. BA собирает требования клиента к желаемому продукту, анализирует, как продукт должен выглядеть, какие функции должны быть реализованы, а также что для этого нужно сделать.
Он проводит исследование конкурентов вместе со специалистами по маркетингу, чтобы убедиться, что проект клиента будет жизнеспособным и конкурентоспособным по сравнению с уже существующими продуктами такого типа.



Какие качества нужны хорошему бизнес-аналитику?
Профессиональный бизнес-аналитик должен иметь сильно развитое воображение, чтобы находить путь в лабиринте различных путей, чтобы добраться до места назначения, несмотря ни на что.
Изменения, различные идеи и творческий подход – эти качества отличают бизнес-аналитиков от других ролей. Предположения всегда должны подкрепляться анализом, поэтому бизнес-аналитики должны ориентироваться на данные.
Бизнес-аналитик должен полностью разбираться в сложных бизнес-процессах с разных точек зрения – как финансовых, так и экономических. Он может корректировать ИТ-проекты для создания реальной ценности для бизнеса. Он обладает навыками для поддержки и постоянного улучшения отношений между пользователем и ИТ-командой, чтобы общение и понимание оставались неизменными.
BA также определяет целевую аудиторию, создает образы пользователей, каркасы и составляет спецификацию требований к программному обеспечению, если все это не было полностью предоставлено клиентом.
Таким образом, роль BA заключается в построении оптимального процесса разработки, интерпретации требований клиента, подготовке документации и

обеспечении эффективного взаимодействия между клиентом и командой.



Технический руководитель – это человек, который переводит бизнес-требования в техническое решение. Из-за этой ответственности полезно привлечь технического руководителя на этапе планирования, чтобы выслушать бизнес-требования с точки зрения клиента и задать вопросы.

Технический руководитель является руководителем группы разработчиков и работает с ними, чтобы предоставить технические детали и оценки предлагаемого решения. Эта информация используется менеджером проекта для создания документов технического задания и иерархической структуры работ для программного проекта.

Технические руководители, как правило, понимают сильные стороны различных членов команды и хорошо умеют распределять рабочие задания между соответствующими ответственными сторонами.

Очень важно, чтобы технический руководитель мог эффективно сообщать статус проекта программного обеспечения менеджеру проекта, чтобы проблемы или отклонения могли быть эффективно устранены как можно скорее.

Технический руководитель также отвечает за установление и соблюдение стандартов и практик с командой разработчиков программного обеспечения.

Разработчики Front-end/Back-end – это люди, которые и занимаются кодированием. Разработчики внешнего интерфейса работают над элементами продукта, ориентированными на клиента. Разработчики внутреннего интерфейса заботятся о его функциональности.

Эта роль незаменима. Для ИТ-проекта всегда нужен специалист по программированию, который пишет код и создает продукты. Он должен писать надежное программное обеспечение.

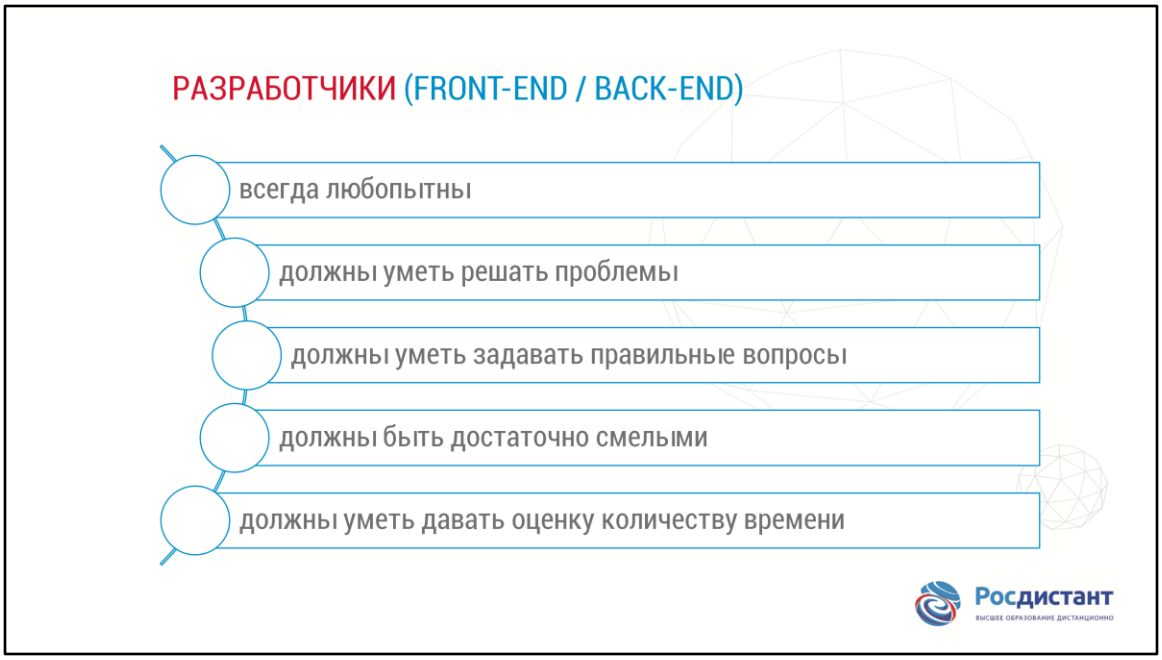
Разработчики программного обеспечения, интерфейсные и внутренние, несут ответственность за использование технических требований технического руководителя для создания оценок затрат и сроков.

Разработчики программного обеспечения несут ответственность за создание результатов и сообщение статуса проекта программного обеспечения техническому руководителю или менеджеру проекта.

Чрезвычайно важно, чтобы другие члены команды эффективно сообщали технические требования разработчикам программного обеспечения, чтобы снизить риски проекта и предоставить проекту программного обеспечения наибольшие шансы на успех.

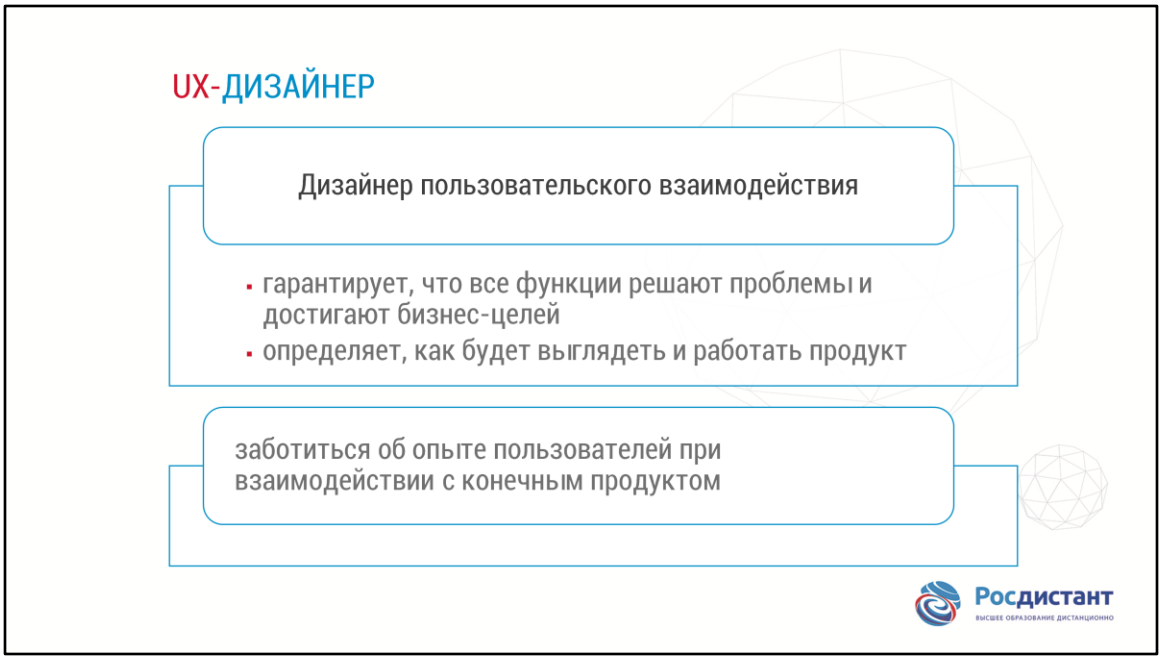
Из всех ролей и обязанностей команды разработчиков программного обеспечения роль разработчика можно назвать ядром команды. Программисты

– это своего рода интерпретаторы, которые переводят человеческий язык и язык графики на язык цифровых технологий, понятный компьютерам и техническим устройствам, и таким образом выполняют то, что хочет клиент.



Какие еще черты характера присущи разработчикам?
Хорошие разработчики всегда любопытны – следят за тенденциями и узнают новое, овладевая навыком, который обычно отличает разработчиков. Они обладают умением учиться. Разработчик должен уметь решать проблемы, четко понимать концепции и препятствия, находить решения и реализовывать их. Разработчики также должны уметь задавать правильные вопросы, которые ускорят процесс кодирования, а иногда и быть достаточно смелыми, чтобы сказать «нет» или предложить лучшие варианты, когда это возможно. Кроме того, важным навыком помимо написания связного кода является оценка количества времени, необходимого для выполнения данной задачи.
С одной стороны, когда дело доходит до опыта, это не самый важный фактор при выборе разработчиков. Иногда свежий творческий подход может быть ценным для проекта, где вам нужно много решений. С другой стороны, опытный старший разработчик чувствует себя уверенно в разных проектах, требующих более широкого подхода. Он комфортно себя чувствует в таких практиках, как гибкая разработка. Он знает различное программное обеспечение для управления задачами, лучше оценивает ситуацию и не боится работать в различных средах.
Можно сказать, что лучший вариант – нанять как опытных, так и молодых

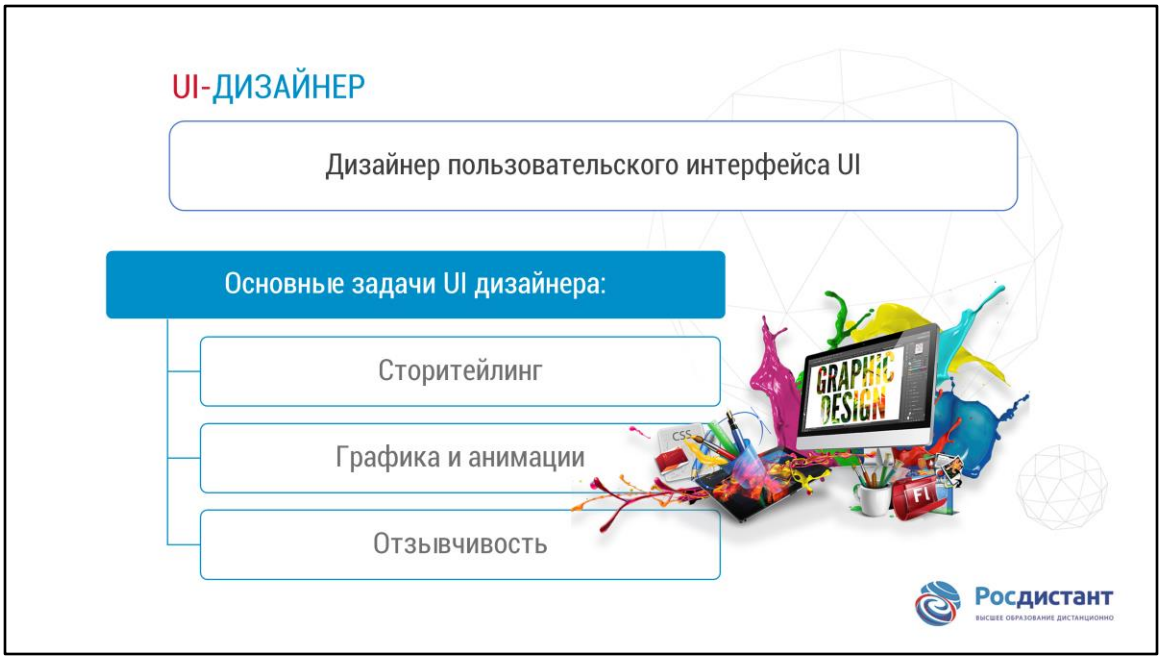
разработчиков. Они могут идеально дополнять навыки друг друга и заполнять пробелы, всё время влияя друг на друга.
Существуют разные уровни программистов, а именно младшие, средние и старшие разработчики, в зависимости от их опыта работы и уровня знаний. Все они также обладают разными областями знаний, разными языками, на которых они кодируют, и платформами, с которыми они работают. Поэтому даже в рамках одного проекта существует большое разнообразие разработчиков программного обеспечения. Например, для обычного проекта разработки мобильного программного обеспечения требуется как минимум разработчик Android , разработчик iOS , а также бэкэнд-разработчик.



Дизайнер пользовательского взаимодействия – UX-дизайнер. Это тот, кто разрабатывает способ взаимодействия пользователей с продуктом. Именно он определяет, как будет выглядеть продукт и как он будет работать. Основное внимание в UX-дизайне уделяется функциональности и удобству использования. Основная задача дизайнера пользовательского взаимодействия – учитывать опыт пользователей при взаимодействии с конечным продуктом. Чтобы достичь удовлетворения, UX-дизайнер должен проявлять чуткость, осознавать потребности пользователей и знать различные методы, позволяющие проверять, работает ли дизайн для целевых пользователей. Хороший дизайнер UX должен быть внимателен к деталям. Его творческий потенциал помогает решать различные проблемы, понимать бизнес-цели и смотреть на них с разных точек зрения, чтобы найти лучшее решение.

С технической стороны UX-дизайнер должен знать некоторые стандарты и процедуры, а также использовать множество различных инструментов для тестирования и решения пользовательских проблем. UX собирают и анализируют информацию о пользователях и основных конкурентах, определяют цели и пожелания клиентов, а также анализируют, как данные задачи уже решаются на рынке. На основе этого анализа находят наиболее удачный и гибкий способ решения для конкретного продукта. Вырабатывают

структуру и стратегию работы приложения.
Некоторые говорят, что разработчики также могут проектировать системы, поэтому нет необходимости использовать UX . Но хотя один разработчик может одновременно выполнять несколько ролей и успешно создавать отличный продукт, наличие дополнительных, более специализированных ролей обеспечивает лучшую оптимизацию навыков. UX и разработчик работают вместе, чтобы оказать положительное влияние на дизайн и разработку продукта. Фактически должны быть обе роли в команде, поскольку они значительно дополняют друг друга



Дизайнер пользовательского интерфейса UI отвечает за внешнее оформление продукта и интерактивность. Он гарантирует, что элементы продукта легко доступны, понятны и ясны благодаря графическому и брендовому дизайну. Как правило, пользовательский интерфейс объединяет взаимодействие с пользователем, визуальный дизайн и информационную архитектуру.
Дизайнер пользовательского интерфейса:

- умеет рисовать;
- умеет превращать идеи в макеты и прототипы.

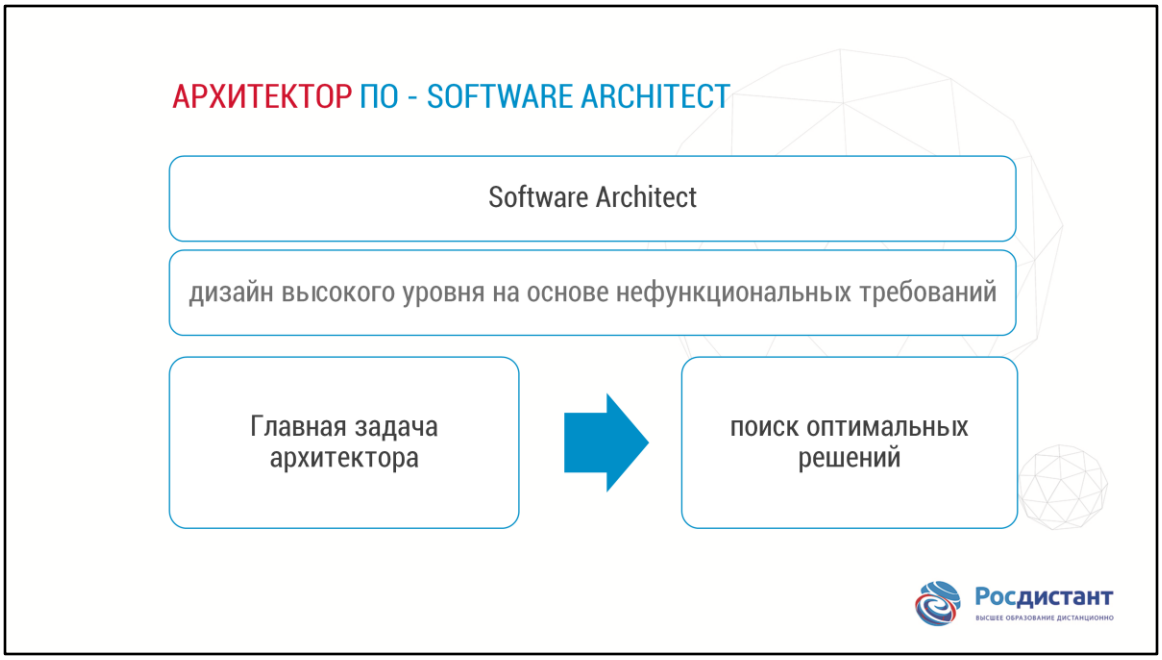
Способность передавать идеи другим разработчикам также имеет решающее значение для этой роли, поскольку UI должны показывать и объяснять, а не просто рассказывать, например, с помощью руководств пользователя. Отличный пользовательский интерфейс понимает ключевые основы UI -дизайна и может применить их к графическому дизайну продукта.
Основные задачи UI-дизайнера включают следующее.

- Сторителлинг. Для UI очень важно понимать, как человеческий мозг реагирует на определенные визуальные сигналы. Задача UI – помочь пользователю правильно применять приложение, опираясь на его визуальные элементы, и при этом используя как можно меньше слов.
- Графика и анимации. Иллюстрации, различные графические элементы и

плавные переходы создают ощущение особенности и уникальности.

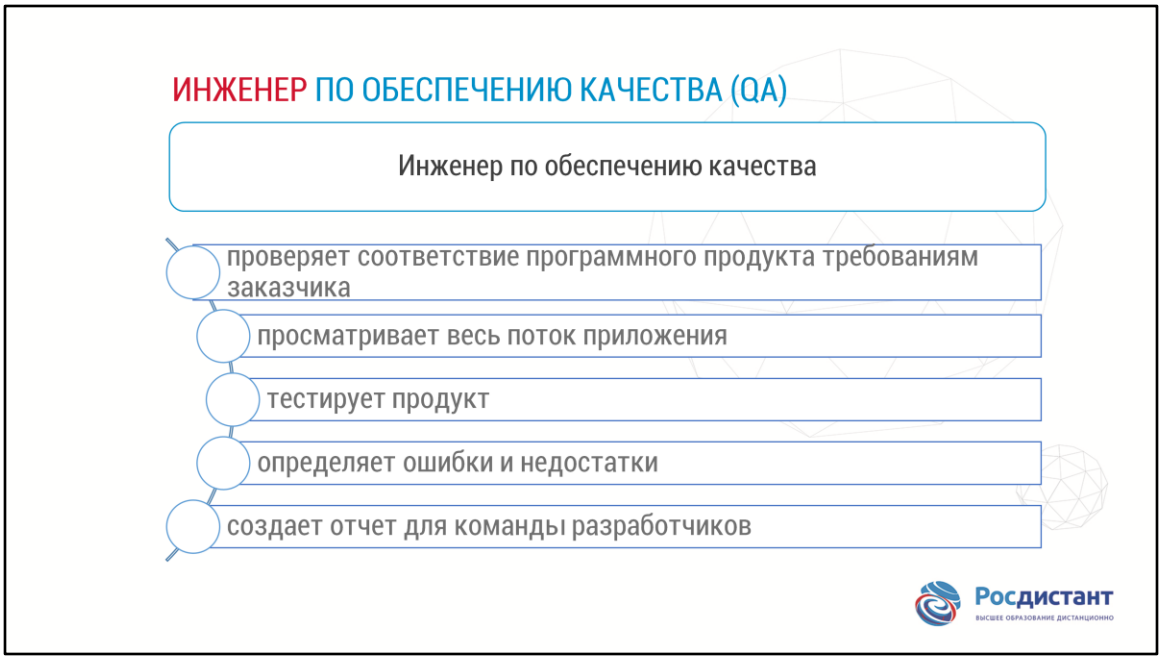
- Отзывчивость. UI-прототипирование, анимации и адаптивность – это те аспекты, которые обеспечат максимальный комфорт использования продукта на любых девайсах.

Чем интереснее и приятнее дизайн продукта, тем лучше пользовательский опыт. Когда пользователи могут легко ориентироваться в продукте и делать с ним то, что хотят, их удовлетворение от использования продукта растет.



Архитектор программного обеспечения – Software Architect делает выбор дизайна высокого уровня на основе нефункциональных требований и диктует стандарты кодирования вместе с инструментами и платформами. Этот человек также отвечает за проверку кода, обеспечение качества дизайна, избежание излишней сложности и сосредоточение внимания на ясности.
Software Architect – это IT-специалист, принимающий решения относительно внутреннего устройства и внешних интерфейсов программного комплекса с учётом проектных требований и имеющихся ресурсов.
Главная задача архитектора – поиск оптимальных, простых, удобных, дешевых решений, которые будут максимально соответствовать потребностям заказчика и возможностям команды. На основании бизнес-требований этот специалист создает функциональную и техническую спецификацию системы, планирует и проектирует способы технической реализации, выбирает технологии.
Архитектор обязан иметь целостное видение всей системы и грамотно определять, как система будет разбита на модули. Он должен понимать, как эти модули будут взаимодействовать между собой. Только после принятия этих решений команда разработчиков сможет приступить к работе над отдельными модулями.
Хороший архитектор программного обеспечения знает программирование,

управление людьми, интересуется психологией, ясно передает идеи и следит за финансами. Но главное качество – это ответственность. Этот человек должен оказывать техническую поддержку и знать требования с самого начала проекта, через выпуск, разработку и завершение улучшений.
Потребность в архитекторе программного обеспечения возникает, когда необходимо решить сложные нефункциональные требования и внести изменения. Вот почему хороший архитектор программного обеспечения имеет решающее значение, когда речь идет о высококачественных проектах. Как правило, архитектура программного обеспечения определяет модель и функции программного обеспечения.



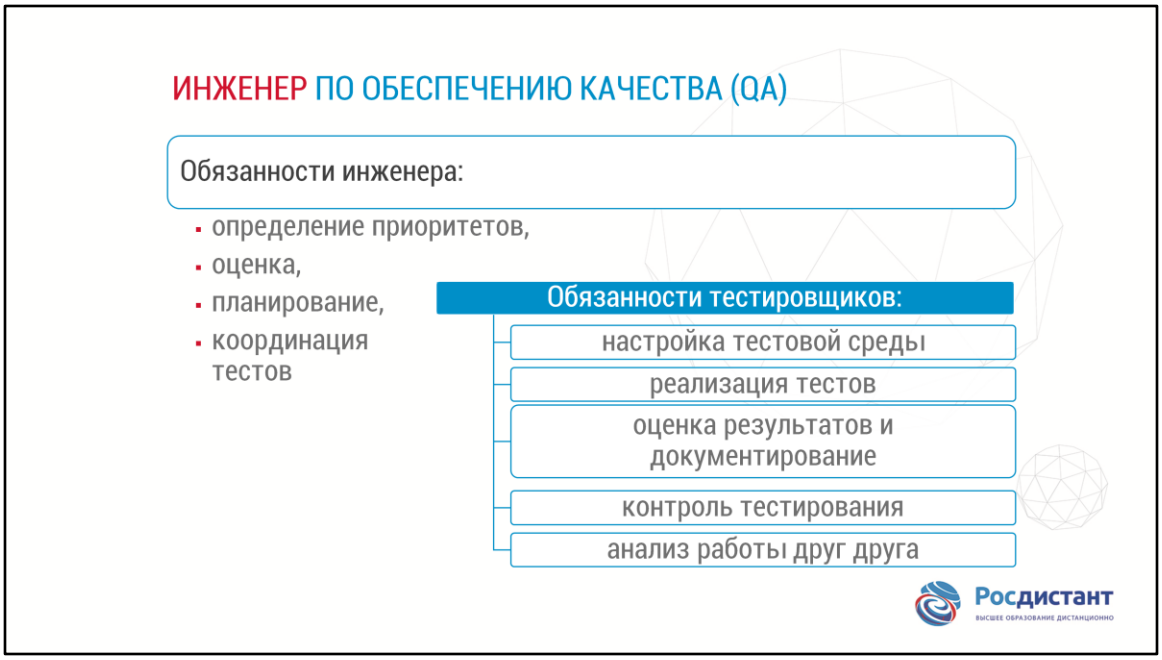
Инженер по обеспечению качества QA – это тот, кто тестирует продукт, чтобы убедиться, что он хорошо работает, соответствует стандартам качества и требованиям клиентов. QA уделяет пристальное внимание мельчайшим деталям. Он обнаруживает ошибки на раннем этапе, чтобы команда могла исправить их до того, как они попадут к пользователям.

Специалисты по обеспечению качества имеют решающее значение для каждого процесса разработки программного обеспечения. Они несут ответственность за безопасность и высокое качество продукта, тестируют продукт, просматривают весь поток приложения, выявляют ошибки и недостатки, составляют отчет для команды разработчиков, которая должна будет провести исправление ошибок. Важно выявить ошибки до завершения работы над продуктом и показать его реальным пользователям, так как это сохранит внешний вид продукта и гарантирует положительный пользовательский опыт.

В другие обязанности QA-специалистов входит не только общее тестирование продукта, но и проверка его соответствия требованиям заказчика.

Инженер по обеспечению качества проверяет, соответствует ли разработанное решение требуемой спецификации, уделяя особое внимание качеству и разрабатывая документы для своевременной и полезной обратной связи. QA в команде разработчиков должен быть перфекционистом с упором на

планирование тестов и тестовых примеров, которые должны быть подробными, структурированными и хорошо спланированными.



Основные действия инженера по обеспечению качества можно описать следующим образом: определение приоритетов, оценка, планирование и координация тестов. Задача хорошего QA заключается в определении, согласовании и посредничестве в процессах, процедурах, спецификациях и стандартах качества. QA также оценивает требования клиентов и проверяет их выполнение.

Тестировщики программного обеспечения гарантируют, что программное решение соответствует бизнес-требованиям и не содержит ошибок и дефектов. На этапах планирования и подготовки тестирования программного обеспечения QA-инженеры должны анализировать и вносить свой вклад в планы тестирования. Они должны уметь анализировать и оценивать технические требования и спецификации проекта.

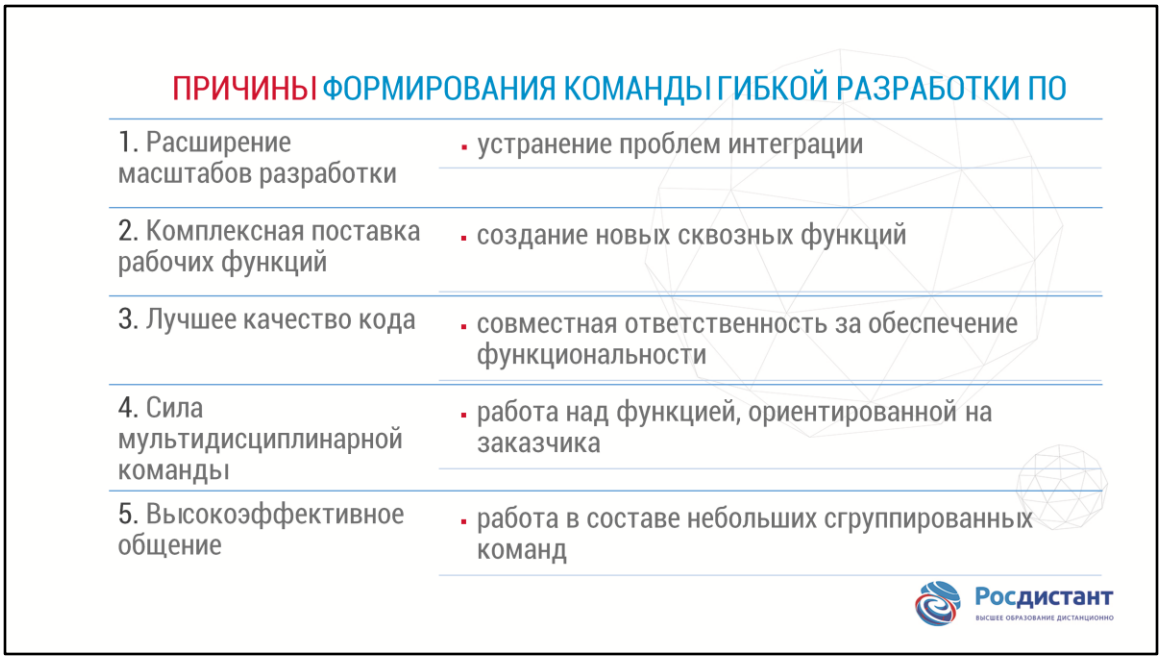
Тестировщики программного обеспечения участвуют в определении условий тестирования и создании тестовых проектов, тестовых примеров, спецификаций тестовых процедур и тестовых данных. Они также могут автоматизировать или помогать автоматизировать тесты.

Некоторые обязанности тестировщиков программного обеспечения могут включать:

- настройку тестовой среды;

- реализацию тестов в тестовой среде;
- оценку результатов и документирование обнаруженных проблем.

А также контролирование тестирования и тестовой среды;
анализ работы друг друга, включая спецификации тестов, отчеты о дефектах и результаты тестов на протяжении всего жизненного цикла тестирования программного обеспечения



И завершим эту тему, выделив наиболее важные причины, по которым мы формируем команду по разработке программного обеспечения, основываясь на гибкой методологии AGILE .

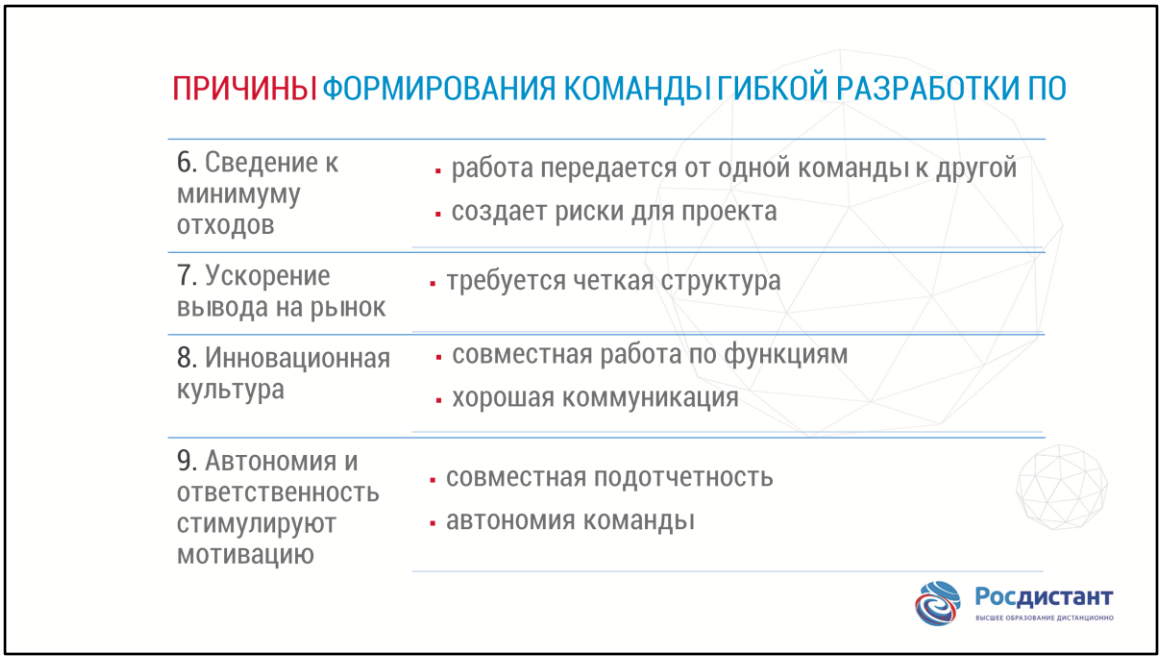
Первая причина. Расширение масштабов гибкой разработки. Команды, ориентированные на технологические уровни, сталкиваются с проблемами интеграции, невмешательства и неправильных представлений о коммуникации. Разработка на основе функциональных групп – естественное следствие нашего подхода к гибкой разработке.

Вторая причина. Комплексная поставка рабочих функций. В модели функциональной группы каждый спринт завершается созданием новых сквозных функций. Команда должна пройти все уровни технологического стека, поэтому нет места для спринтов без явного увеличения продукта.

Третья причина. Лучшее качество кода. Совместная ответственность за обеспечение функциональности продукта заставляет команду поддерживать чистоту кода и повышает конкуренцию между командами, работающими над одним проектом.

Четвертая причина. Сила сбалансированной мультидисциплинарной команды. Функциональные группы работают над полной ориентированной на клиента функцией, охватывающей все компоненты, уровни архитектуры и дисциплины.

Команду составляют люди разного профиля. Это означает, что они могут переключаться между различными функциями, и их роли не ограничиваются только одной компетенцией в области разработки программного обеспечения.
Пятая причина. Высокоэффективное общение. Функциональная командная работа основана на небольших, сгруппированных командах, которые могут легко общаться ежедневно, обеспечивая четкий процесс разработки и открытое общение. Такая рабочая среда способствует командной работе и укрепляет командные отношения.
Еще одним аспектом совместного размещения является улучшение процессов обучения на индивидуальном и групповом уровнях за счет сотрудничества с экспертами в различных областях.



Продолжим выделять причины, по которым мы формируем команду по разработке программного обеспечения, основываясь на гибкой методологии AGILE .

Шестая причина. Сведение к минимуму отходов. В традиционной модели разработки компонентного программного обеспечения работа передается от одной команды к другой, что отнимает много времени и создает риски для проекта. Всегда есть опасность, что разработанная функциональность окажется недостаточной для работы следующей команды.

И наоборот, структурированная разработка по функциям значительно сокращает потери при передаче. Это означает меньше ожидания и более быстрые результаты.

Седьмая причина. Ускорение вывода на рынок. Для управления разработчиками программного обеспечения требуется четкая структура, что на практике отражается в разделении модулей приложения и распределении их по разным группам специалистов. Параллельная работа множества команд над разными модулями приложений ускоряет весь процесс разработки программного обеспечения.

Восьмая причина. Инновационная культура. Совместная работа по функциям стимулирует инновации. Члены команды чувствуют себя в безопасности и

уверены друг в друге, потому что они долгое время работают вместе и имеют хороший коммуникационный поток. Люди не боятся предлагать и тестировать новые решения. Кроме того, решения принимаются локально, поэтому в процессе создания нет никаких препятствий. Эта рабочая среда формирует культуру инноваций.

Девятая причина. Автономия и ответственность стимулируют мотивацию. Члены специальной группы извлекают огромные выгоды из совместной подотчетности и автономии команды, в том числе на индивидуальном уровне.

Они чувствуют более высокую мотивацию и удовлетворение от работы, потому что они могут работать над сквозной функцией и, как следствие, полностью контролировать конечные результаты своей работы.

Таким образом, мы посмотрели основные роли в командной разработке и перейдем к рассмотрению ИТ-специальностей, которые сегодня востребованы.