# Newbies course content

## Обзор методологий и подходов к разработке ПО

## Детализация agile методологий

## Agile (описание, принципы, техники)

## Scrum (процессы, события, артефакты)

## Вводный модуль. Описание курса, зачем и кому он нужен.

* описание курса
* назначение курса
* краткое содержание/оглавление курса
* ответы на вопросы

## Модуль №1. Общие советы начинающему профессионалу

### Совет №1. Освой сферу бизнеса (предметную область), в которой работаешь. Разберись, как работает бизнес в целом.

Следует разрушить границу между бизнесом и информационными технологиями, смотреть на технологии глазами бизнеса компании. Изучи основные принципы бизнеса, с которым связана твоя деятельность.

Разберись в основах финансирования бизнеса (откуда в твоем бизнесе берутся деньги?), как ты участвуешь в бизнесе своей компании, в бизнесе Заказчика. Ты должен понимать, как твоя работа помогает зарабатывать или экономить деньги, как Заказчику, так и твоей компании. Ты не сможешь помогать бизнесу творчески, если не знаешь, как он устроен.

### Совет №2. Будь специалистом.

Выбери свою специализацию и планомерно углубляй свои познания/навыки в ней. Невозможно быть "всеведущим", поэтому выбери одну технологию/язык разработки и сконцентрируй свое внимание и усилия на нем.

### Совет №3. Будь универсалом.

Универсалы встречаются редко, поэтому ценятся высоко. Ваши навыки не должны ограничиваться одной технологической платформой. Необходимо регулярно выделять время на изучение чего-то нового.

Казалось бы, что это противоречит предыдущему пункту, но нет - речь идет о том, что необходимо иеть широкий кругозор в технологиях и знать инструментарий - чем о бОльшем числе инструментов ты знаешь, тем лучше ты будешь понимать какой из них более применим в конкретной ситуации, какой следует изучить глубже, чем знаешь сейчас. Как и в предыдущем пункте - постоянно практикуйся/инвестируй. Также, аналогия со спортом.

### Совет №3. Постоянно практикуйся, инвестируй время в свое ремесло.

Тренируйся на пределе своих возможностей. Аналогия со спортом. Code Kata упражнения.

### Совет №3. Будь "самым худшим специалистом" в комнате. Найди наставника.

Всегда ищи у кого и чему ты можешь научиться. В зависимости от другого человека нет ничего страшного. Главное - убедиться, что это правильный человек. Без примера для подражания сложно найти стимул к совершенствованию. Наставник поможет выбрать верные направления для обучения, из огромного множества вариантов. Еще наставник служит доверенным лицом, которое наблюдает и оценивает принимаемые тобой решения и твои успехи.

В идеале было бы здорово найти человека, который будет активно тебя направлять, но это далеко не всегда реализуемо на практике. Ты можешь стать наставником самому себе :)

### Совет №3. Помогай учиться коллегам. Стань наставником.

Всегда делись полученными знаниями, помни - обучение один из самых лучших способов самому лучше разобраться в предмете. Чтобы понять, на самом ли деле ты хорошо разбираешься в теме, попробуй объяснить эту тему кому-то еще.

Не следует также недооценивать тот факт, что помогать людям *приятно*. Если мысленно вы можете завладеть вниманием публики, имеет смысл воспользоваться этой способностью в альтруистических целях. В неопределенности современной экономической ситуации реальная *помощь* другим — это работа, с которой тебя не могут уво­лить. И платят за нее валютой, которая не обесценивается с инфляцией. Как правило - наставники не попадают под сокращение :)

### Полюби или уходи.

Если ты хочешь достичь высот в своей работе, ты должен быть влюблен в нее. Если же тебе все равно, это непременно станет заметно. Работай потому, что не можешь не работать. Найди работу, которая тебе по-настоящему интересна.

Mood monitoring. Проведи следующий эксперимент: начиная со следующего понедельника на протяжении двух недель веди журнал. Каждый будний день, проснувшись, отмечай уровень энтузиазма по десятибалльной шкале: 1 означает, что ты предпочел бы заболеть, лишь бы не идти на работу, а 10 — что высокая температура не может удержать тебя в постели, так тебя захватила идея по поводу решения очередной задачи. Через две недели внимательно посмотри на результаты. Есть ли на графике пики? Какие тенденции он показывает? Он прижат вниз или стремится ввысь? Какую бы ты поставил себе оценку, если бы это был школьный тест? На протяжении следующих двух недель каждое утро думай над тем, каким образом ты собираешься сегодня работать на 10. Планируй сегодняшнее рабочее расписание так, чтобы на следующий день ты не мог дождаться момента, когда приступишь к работе. Каждый день фиксируй вчерашний уровень энтузиазма. Если и после двух недель ничего коренным образом не поменяется, имеет смысл задуматься о смене работы.

### Учись ловить рыбу. Будь проактивным.

Не жди, пока тебе расскажут, спрашивай сам! Анализируй чужой код из открытых источников, код приложений и библиотек, которые используешь в своей работе. Чужой код не только позволяет найти ответы на конкретные вопросы, но и служит своего рода увеличительным стеклом для рассмотрения собственного стиля и способностей. Доступное количество таких программ столь велико, что вряд ли их все можно прочитать. Разумеется, попадаются среди этого изобилия и плохие проекты, тем не менее нам доступно и довольно много *великолепных* примеров. Существует открытый код, реализующий практически любую решаемую программно задачу и почти на всех доступных языках программирования. Критически анализируя этот код, ты постепенно начнешь вырабатывать собственный вкус, как это бывает в музыке, живописи и литературе. Различные стили и приемы тебя позабавят, удивят, рассердят и (надеюсь) *стимулируют* к работе. Ищущий найдет тут всё — от трюков, повышающих продуктивность работы, до парадигм проектирования, полностью меняющих подход к целому классу проблем. Как и в искусстве, изучая особенности чужих работ и учась на них, ты выработаешь свой ни на кого не похожий стиль разработки ПО. Изучай паттерны и анти-паттерны разработки ПО.

### Подход к работе.

Разработка ПО - это не некая вещь, а процесс создания некой вещи. При написании кода важно сосредоточиться не только на разрабатываемом продукте, но и на самом процессе разработки. Отвлекаясь от процесса, мы рискуем опоздать со сроками выполнения задания, получить некачественный продукт или не получить вообще ничего. Ни один из этих вариантов заказчика не обрадует. К счастью, процесс создания хорошего программного обеспечения (и в целом продукции) давно как следует обдуман. И изрядная часть этой информации легла в основу целой группы методик.

**ссылка на раздел с обзором методик**

Как программист, тестировщик или дизайнер ПО ты можешь считать, что сам по себе процесс разработки не входит в сферу твоей ответственности. Возможно, это правильно, так как ты всего лишь наемный работник. Но, к сожалению, обычно эта ответственность повисает в воздухе. И даже если ее на кого-то возлагают, она передается «группе организации процесса» или другому отдельному подразделению. Но дело в том, что для успешного внедрения метода разработки программного обеспечения его должны принять те, кто будет им пользоваться, — такие, как ты.

Найти человека, который сможет писать программы, куда проще, чем человека, умеющего организовать процесс написания программ. Поэтому имеющиеся в твоем арсенале сведения о принципах организации процесса разработки ПО будут тебе только в плюс.

Автоматизируй свою работу и повседневные задачи.

Выбери повторяющуюся задачу, с которой тебе часто приходится сталки- ваться, и напиши для нее генератор кода Начни с простого О возможности многократного использования пока можно не беспокоиться Просто сделай так, чтобы генератор экономил твое время Подумай, как поднять уровень абстракции генерируемого кода

**ссылка на CI/CD модуль**

## Методы оценки задач

## Решение вопросов внутри команды (рабочий конфликт)

## Обзор CI/CD практик

## Do's and don'ts в корпоративной среде

## Коммуникация с заказчиком

## Вопросы/ответы

## Использованная/рекомендуемая литература

1. Чед Фаулер. Программист-фанатик.
2. аа
3. ааурп