Лекция 3. Управление проектами. Определения и концепции.

# Управление разработкой программного обеспечения

Материал из Википедии — свободной энциклопедии

**Управле́ние разрабо́ткой програ́ммного обеспе́чения** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Software project management*) — особый вид [управления проектами](https://ru.wikipedia.org/wiki/%D0%A3%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B0%D0%BC%D0%B8), в рамках которого происходит планирование, отслеживание и контроль за проектами по разработке [программного обеспечения](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5). Ключевым моментом в управлении проектом по разработке программного обеспечения является правильный выбор метода разработки.

## История

### Причины возникновения

Основная статья: [**Кризис программного обеспечения**](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%B7%D0%B8%D1%81_%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)

В связи с быстрым увеличением мощностей [компьютеров](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80) в 60-е и 70-е годы [XX века](https://ru.wikipedia.org/wiki/XX_%D0%B2%D0%B5%D0%BA) проблемы, которые могли быть решены с их помощью, становились сложнее. Поэтому требовались более масштабные [проекты](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B5%D0%BA%D1%82), включавшие в себя координацию труда большего числа людей и написание гораздо большего объёма [кода](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4). Однако методы, применявшиеся к управлению такими проектами, были рассчитаны на решение задач в рамках намного меньших проектов. Отсутствие необходимой методологии привело к огромному числу провальных проектов. Попытки изменить положение к лучшему привели к созданию [новой модели процесса разработки](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%81%D0%BA%D0%B0%D0%B4%D0%BD%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C), концентрировавшей больше внимания на соответствие конечного программного продукта изначальным требованиям [заказчика](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%BA%D0%B0%D0%B7%D1%87%D0%B8%D0%BA).

### Дальнейшее развитие

Исследования проектов, окончившихся неудачей, показали, что самыми распространёнными причинами провалов были:[[1]](https://ru.wikipedia.org/wiki/%D0%A3%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%BE%D0%B9_%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#cite_note-1)

1. Невыполнимые или неясно сформулированные цели проекта
2. Ошибочный подсчет необходимых [ресурсов](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%81%D1%83%D1%80%D1%81)
3. Некорректно определённые [системные требования](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%BD%D1%8B%D0%B5_%D1%82%D1%80%D0%B5%D0%B1%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)
4. Неосведомлённость [управляющего проектом](https://ru.wikipedia.org/w/index.php?title=%D0%9C%D0%B5%D0%BD%D0%B5%D0%B4%D0%B6%D0%B5%D1%80_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%BE%D0%B2&action=edit&redlink=1) о точном состоянии проекта
5. Неуправляемые риски
6. Слабое взаимодействие между заказчиком, [разработчиком](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D1%87%D0%B8%D0%BA_%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) и [пользователем](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C)
7. Использование слишком новых, нестабильных технологий
8. Неспособность справиться со сложностями проекта
9. Слабое управление проектом
10. Финансовые ограничения

С тех пор было представлено несколько усовершенствований уже существующих ([итеративный подход](https://ru.wikipedia.org/wiki/%D0%98%D1%82%D0%B5%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%B0%D1%8F_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0)) и совершенно новых ([разработка через тестирование](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D1%87%D0%B5%D1%80%D0%B5%D0%B7_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) методов управления разработкой программного обеспечения. Тем не менее, сегодня проглядывается тенденция к переходу от каскадной модели к [циклической](https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%80%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C), имитирующей [стадии разработки программного обеспечения](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D0%B4%D0%B8%D0%B8_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8_%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).

## Основные методы разработки программного обеспечения

### [ГОСТы](https://ru.wikipedia.org/wiki/%D0%93%D0%BE%D1%81%D1%83%D0%B4%D0%B0%D1%80%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9_%D1%81%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82)

ГОСТ 19 «Единая система программной документации»[[2]](https://ru.wikipedia.org/wiki/%D0%A3%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%BE%D0%B9_%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#cite_note-2) и ГОСТ 34 «Стандарты на разработку автоматизированных систем»[[3]](https://ru.wikipedia.org/wiki/%D0%A3%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%BE%D0%B9_%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#cite_note-3) ориентированы на последовательный подход в разработке программного обеспечения. Разработка в соответствии с этими стандартами проводится по этапам, каждый из которых предполагает выполнение строго определенных работ. Строгое следование этим ГОСТам приводит к каскадной модели. На основе этих стандартов разрабатываются программные системы по [госзаказам](https://ru.wikipedia.org/wiki/%D0%93%D0%BE%D1%81%D1%83%D0%B4%D0%B0%D1%80%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9_%D0%B7%D0%B0%D0%BA%D0%B0%D0%B7) в России.

### [SW-CMM](https://ru.wikipedia.org/wiki/Capability_Maturity_Model)

Данная модель была разработана в середине 80-х годов XX века Институтом программной инженерии, входящим в состав [Университета Карнеги-Мелона](https://ru.wikipedia.org/wiki/%D0%A3%D0%BD%D0%B8%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%82%D0%B5%D1%82_%D0%9A%D0%B0%D1%80%D0%BD%D0%B5%D0%B3%D0%B8_%E2%80%94_%D0%9C%D0%B5%D0%BB%D0%BB%D0%BE%D0%BD) с целью создать эталонную модель организации разработки программного обеспечения. Основана на проверке соответствия организации определённым требованиям и определении уровня зрелости процесса разработки программного обеспечения.

### [RUP](https://ru.wikipedia.org/wiki/Rational_Unified_Process)

Унифицированный процесс был разработан компанией [Rational Software](https://ru.wikipedia.org/wiki/Rational_Software) в качестве дополнения к языку [UML](https://ru.wikipedia.org/wiki/UML). Модель RUP описывает абстрактный общий процесс, на основе которого организация или проектная команда должна создать конкретный специализированный процесс, ориентированный на её потребности.

### [MSF](https://ru.wikipedia.org/wiki/Microsoft_Solutions_Framework)

Microsoft Solutions Framework построена на основе итеративной разработки. Особенностью MSF является большое внимание к созданию эффективной и небюрократизированной команды.

### [PSP](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%B4%D0%B8%D0%B2%D0%B8%D0%B4%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8)/TSP

Personal Software Process определяет требования к компетенциям разработчика для того, чтобы они смогли получить необходимые навыки для Team Software Process. Team Software Process в комбинации с Personal Software Process делает ставку на самоуправляемые команды численностью 3-20 человек. Команды должны:

* Установить собственные цели
* Составить свой процесс и планы
* Отслеживать работу
* Поддерживать мотивацию и максимальную производительность

### [Agile](https://ru.wikipedia.org/wiki/Agile)

Основная идея всех гибких моделей заключается в том, что применяемый в разработке программного обеспечения процесс должен быть адаптивным. Они ставят своей целью ориентированность на людей и их взаимодействие, а не на процессы и средства. Все гибкие модели основываются на итеративности, инкрементальности, самоуправляемости команды и адаптивности процесса.

## Сопутствующие процессы при управлении проектом

Процесс управления проектом по разработке программного обеспечения включает в себя другие, более специфицированные процессы, направленные на принятие тех или иных бизнес-решений. Многие из них могут применяться к другим видам проектов. Например:

* [Управление рисками](https://ru.wikipedia.org/wiki/%D0%A3%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D1%80%D0%B8%D1%81%D0%BA%D0%B0%D0%BC%D0%B8) начинается с составления [технико-экономического обоснования](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%85%D0%BD%D0%B8%D0%BA%D0%BE-%D1%8D%D0%BA%D0%BE%D0%BD%D0%BE%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D0%BE%D0%B1%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), включающего в себя расчет возможных доходов и расходов проекта и список возможных неуправляемых рисков, а также план действий в случае их наступления. Важным моментом в управлении рисками проекта по разработке программного обеспечения является постоянный мониторинг текущих рисков на протяжении всего проекта.
* [Управление требованиями](https://ru.wikipedia.org/wiki/%D0%A3%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D1%82%D1%80%D0%B5%D0%B1%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F%D0%BC%D0%B8), включающее в себя [анализ требований](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7_%D1%82%D1%80%D0%B5%D0%B1%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B9), является важной частью процесса управлением разработкой программного обеспечения. Посредством анализа требований [бизнес-аналитики](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B7%D0%BD%D0%B5%D1%81-%D0%B0%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7) и разработчики программного обеспечения выявляют потребности и требования заказчика, предъявляемые к конечному программному обеспечению.
* [Конфигурационное управление](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%84%D0%B8%D0%B3%D1%83%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%BE%D0%B5_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5) в рамках управления программным проектом в общем заключается в [управлении версиями](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%8F%D0%BC%D0%B8), определении правил именования переменных, функций, классов и т. д. в исходном коде и документации, а также определении соглашений об архивировании программного обеспечения.
* [Управление изменениями](https://ru.wikipedia.org/wiki/%D0%A3%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%B8%D0%B7%D0%BC%D0%B5%D0%BD%D0%B5%D0%BD%D0%B8%D1%8F%D0%BC%D0%B8) проекта по разработке программного обеспечения ориентировано на анализ влияния изменений свойств и функций конечного программного обеспечения в процессе реализации проекта. Управление изменениями тесно связано с управлением требованиями, так как бизнес-аналитики и разработчики программного обеспечения, выявив изменения в потребностях и требованиях заказчика, способны перестроить или улучшить дальнейшую реализацию проекта. Однако каждое изменение или нововведение способно так или иначе повлиять на сроки проекта или его [бюджет](https://ru.wikipedia.org/wiki/%D0%91%D1%8E%D0%B4%D0%B6%D0%B5%D1%82_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B0), поэтому очень важно провести предварительную оценку рисков.

## Планирование, отслеживание и контроль за проектом

* Целью составления плана проекта является определение объёма и содержания работ, необходимых для успешного осуществления проекта, [оценка затрат](https://ru.wikipedia.org/wiki/%D0%9E%D1%86%D0%B5%D0%BD%D0%BA%D0%B0_%D0%B7%D0%B0%D1%82%D1%80%D0%B0%D1%82_%D0%BD%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D1%83_%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) и составление [графика работ](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0_%D0%93%D0%B0%D0%BD%D1%82%D0%B0). Планирование прежде всего начинается с анализа требований, определяющих свойства и функции создаваемого программного обеспечения. Затем определяются задачи, выполнение которых приведет к успешному завершению проекта.
* Цель отслеживания и контроля за проектом заключается в поддержании соответствия действий команды текущему состоянию проекта. В случае отклонения проекта от плана управляющий проектом может оперативно исправлять выявленные ошибки. Отслеживание состояния проекта включает в себя регулярные встречи с командой для обсуждения текущего состояния проекта.

## Философия

В целом к управлению разработкой программного обеспечения, имеющим много заимствований из управления проектами, можно применять методики из [традиционного управления](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D0%BD%D0%B5%D0%B4%D0%B6%D0%BC%D0%B5%D0%BD%D1%82). Однако в силу уникальности отрасли опыт профессионалов, накопленный в материальном производстве и изложенный например в стандарте [PMI PMBOK](https://ru.wikipedia.org/wiki/Project_Management_Body_of_Knowledge), мало способствует успеху в управлении проектом по созданию программного обеспечения. По поводу того, какими знаниями и навыками должен обладать управляющий проектом по разработке программного обеспечения, существует много мнений. Например, известный американский ученый в области [компьютерных наук](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D1%8B%D0%B5_%D0%BD%D0%B0%D1%83%D0%BA%D0%B8) Джон Рейнольдс писал:

Некоторые утверждают, что можно управлять созданием программного обеспечения, не имея никаких навыков в [программировании](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5). Такая уверенность, кажется, возникает в результате ошибочного мнения о том, что создание программного обеспечения является одной из форм производства. Но производство является созданием повторяющихся идентичных объектов, в то время как производство программного обеспечения является созданием уникальных объектов, то есть, это одна из форм [творчества](https://ru.wikipedia.org/wiki/%D0%A2%D0%B2%D0%BE%D1%80%D1%87%D0%B5%D1%81%D1%82%D0%B2%D0%BE). Таким образом, производство программного обеспечения сродни [издательскому делу](https://ru.wikipedia.org/wiki/%D0%98%D0%B7%D0%B4%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D0%BA%D0%BE%D0%B5_%D0%B4%D0%B5%D0%BB%D0%BE) — управляющий разработкой программного обеспечения, не умеющий программировать, подобен редактору газеты, который не умеет писать.

## См. также

* [Инженерия программного обеспечения](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%B6%D0%B5%D0%BD%D0%B5%D1%80%D0%B8%D1%8F_%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)
* [Управление проектами](https://ru.wikipedia.org/wiki/%D0%A3%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B0%D0%BC%D0%B8)
* [Разработка программного обеспечения](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%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)

Рассмотрим понятие жизненного цикла программы, данное в [1].

Согласно стандарту жизненный цикл программы, программной системы,

программного продукта включает в себя разработку, развертывание, поддержку

и сопровождение. Если программный продукт не коробочный, а достаточно

сложный, то его развертывание у клиентов, как правило, реализуется

отдельными самостоятельными проектами внедрения. Сопровождение

включает в себя устранение критических неисправностей в системе и

реализуется часто не как проект а, как процессная деятельность. Поддержка

заключается в разработке новой функциональности, переработке уже

существующей функциональности, в связи с изменением требований, и

улучшением продукта, а также устранение некритических замечаний к ПО,

выявленных при его эксплуатации (Рисунок 2.1). Жизненный цикл программного

продукта завершается выводом продукта из эксплуатации и снятием его с

поддержки и сопровождения.



Рисунок 2.1 Жизненный цикл программного продукта

**Что надо делать для успеха программного проекта**

Стив Макконнелл в своей книге [17] приводит тест программного проекта на

выживание. Этот чек-лист из 33-х пунктов, который я считаю необходимым

процитировать с небольшими корректировками. Руководитель программного

проекта должен его периодически использовать для внутреннего аудита своих

процессов.

Чтобы программный проект стал успешным, необходимо:

1. Четко ставить цели.

2. Определять способ достижения целей.

3. Контролировать и управлять реализацией.

4. Анализировать угрозы и противодействовать им.

5. Создавать команду.

1. Ставим цели

1.1. Концепция определяет ясные недвусмысленные цели.

1.2. Все члены команды считают концепцию реалистичной.

1.3. У проекта имеется обоснование экономической эффективности.

1.4. Разработан прототип пользовательского интерфейса.

1.5. Разработана спецификация целевых функций программного продукта.

1.6. С конечными пользователями продукта налажена двухсторонняя связь

2. Определяем способ достижения целей

2.1. Имеется детальный письменный план разработки продукта.

2.2. В список задач проекта включены «второстепенные» задачи

(управление конфигурациями, конвертация данных, интеграция с

другими системами).

2.3. После каждой фазы проекта обновляется расписание и бюджет.

2.4. Архитектура и проектные решения документированы.

2.5. Имеется план обеспечения качества, определяющий тестирование и

рецензирование.

2.6. Определен план многоэтапной поставки продукта.

2.7. В плане учтены обучение, выходные, отпуска, больничные.

2.8. План проекта и расписание одобрен всеми участниками команды.

3. Контролируем и управляем реализацией

3.1. У проекта есть куратор. Это такой топ-менеджер исполняющей

компании, который лично заинтересован в успехе данного проекта.

3.2. У проекта есть менеджер, причем только один!

3.3. В плане проекта определены «бинарные» контрольные точки.

3.4. Все заинтересованные стороны могут получить необходимую

информацию о ходе проекта.

3.5. Между руководством и разработчиками установлены доверительные

отношения.

3.6. Установлена процедура управления изменениями в проекте.

3.7. Определены лица, ответственные за решение о принятии изменений в

проекте.

3.8. План, расписание и статусная информация по проекту доступна

каждому участнику.

3.9. Код системы проходит автоматическое рецензирование.

3.10. Применяется система управления дефектами.

4. Анализируем угрозы

4.1. Имеется список рисков проекта. Осуществляется его регулярный

анализ и обновление.

4.2. Руководитель проекта отслеживает возникновение новых рисков.

4.3. Для каждого подрядчика определено лицо, ответственное за работу с

ним.

5. Работаем над созданием команды

5.1. Опыт команды достаточен для выполнения проекта.

5.2. У команды достаточная компетенция в прикладной области.

5.3. В проекте имеется технический лидер.

5.4. Численность персонала достаточна.

5.5. У команды имеется достаточная сплоченность.

5.6. Все участники привержены проекту.

Оценка и интерпретация теста

Оценка: сумма баллов, каждый пункт оценивается от 0 до 3:

 0 – даже не слышали об этом;

 1 – слышали, но пока не применяем;

 2 – применяется частично;

 3 – применяется в полной мере.

Поправочные коэффициенты:

 для малых проектов (до 5 человек) - 1.5;

 для средних (от 5 до 20 человек) – 1.25.

Результат:

 <40 – завершение проекта сомнительно.

 40-59 – средний результат. В ходе проекта следует ожидать

серьезные проблемы.

 60-79 – хороший результат. Проект, скорее всего, будет успешным.

 80-89 – отличный результат. Вероятность успеха высока.

 >90 – великолепный результат. 100% шансов на успех.

Этот чек-лист перечисляет, что надо делать для успеха программного проекта,

но не дает ответ на вопрос как это следует делать. Именно об этом пойдет речь

в остальных лекциях.

Выводы

То, что производят программисты нематериально – это коллективные мысли и

идеи, выраженные на языке программирования. В силу уникальности отрасли

опыт, накопленный в отраслях материального производства, мало способствует

успеху в управлении программным проектом. Прямые аналогии с этими

отраслями не работают. Управлять разработкой ПО надо иначе.

Не существует единственного правильного процесса разработки ПО.

Эффективный производственный процесс должен основываться на

итеративности, инкрементальности, самоуправляемости команды и

адаптивности. Главный принцип: не люди должны строиться под выбранную

модель процесса, а модель процесса должна подстраиваться под конкретную

команду, чтобы обеспечить ее наивысшую производительность.

Чтобы программный проект стал успешным, необходимо:

1. Четко ставить цели.

2. Определять способ достижения целей.

3. Контролировать и управлять реализацией.

4. Анализировать угрозы и противодействовать им.

5. Создавать команду.

Задание:

1. Продолжить анализ материалов Tadviser.ru по разработке программного обеспечения.

2. Начать анализ лекции 2 [1] и подготовится к обсуждению.

Литература:

* 1. С.Архипенков. Лекции по управлению программными проектами. Москва, 2009, 127с.