

---

# Управление IT проектами

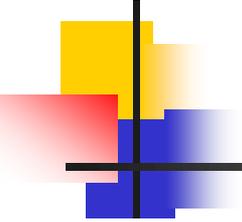
## Часть 3

Финансовое обоснование проекта.

Контракты.

Управление конфигурацией.

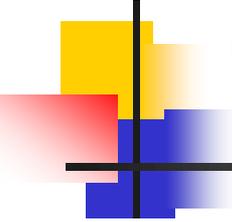
Модели организаций.



---

# Обоснование проекта

Окупаемость,  
Приведенная стоимость / Present value,  
Поток средств / Cash flow,  
Возврат инвестиций / ROI, Discounted ROI

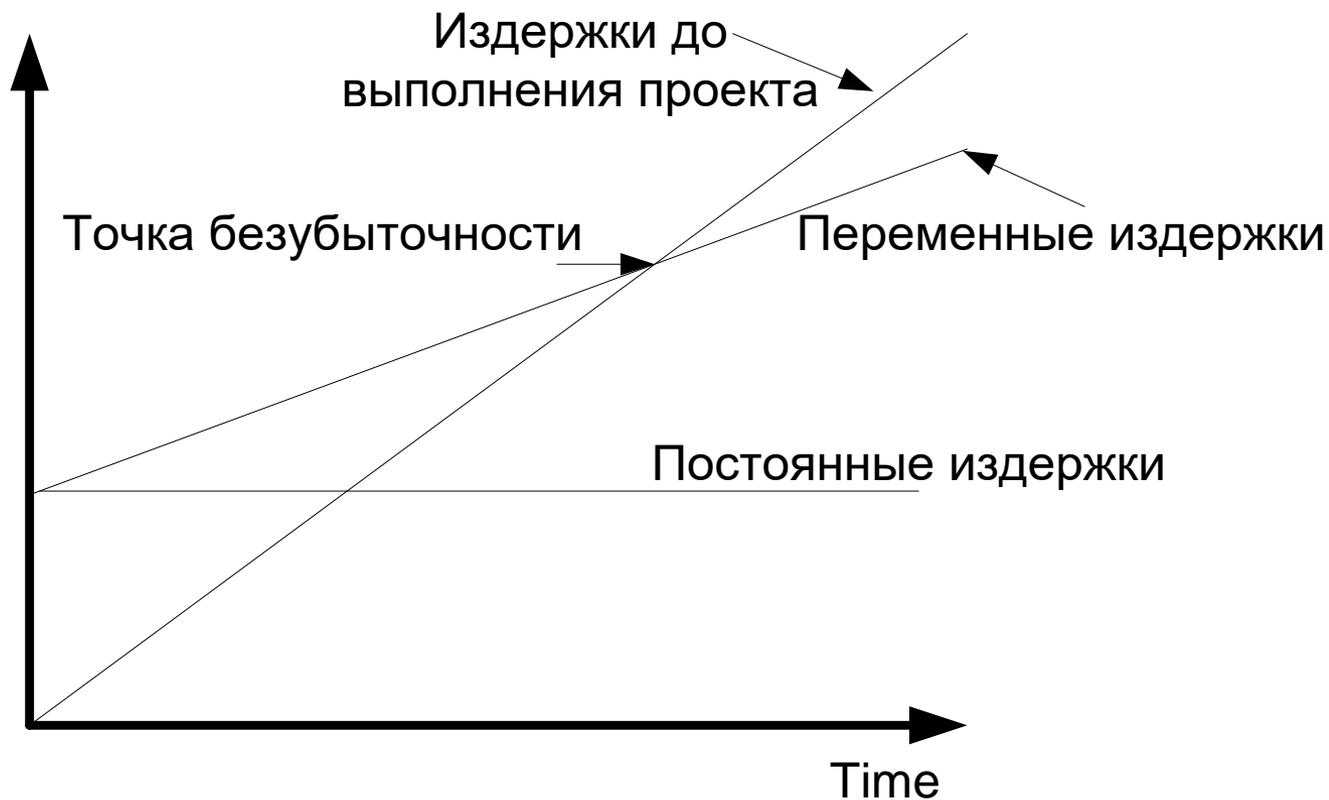


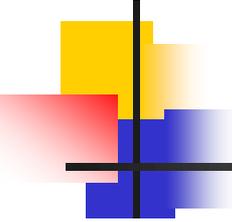
# Обоснование проекта

---

- Всякий ли проект действительно нужен? Выгоден?
- Как посчитать, выгодно ли проведение проекта?
- Как сравнить 2 альтернативных проекта?

# Анализ безубыточности (Break Even chart)



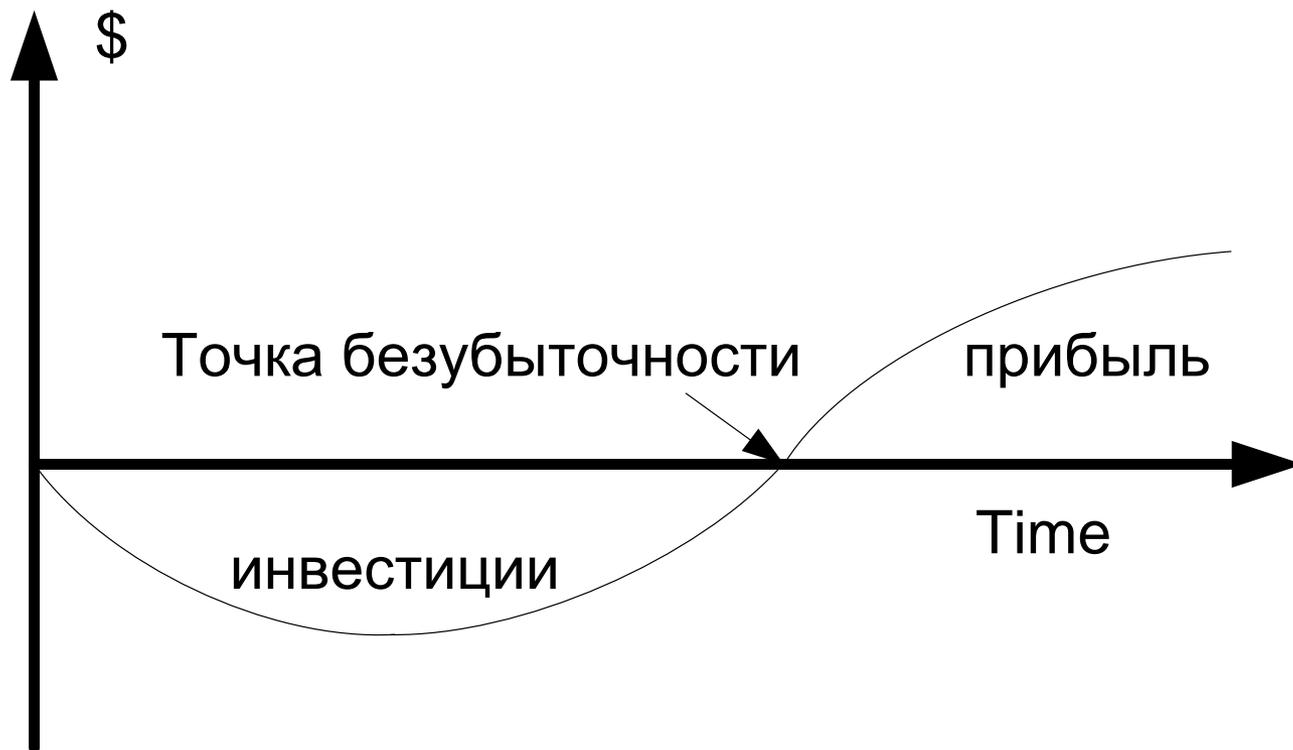


# Период окупаемости

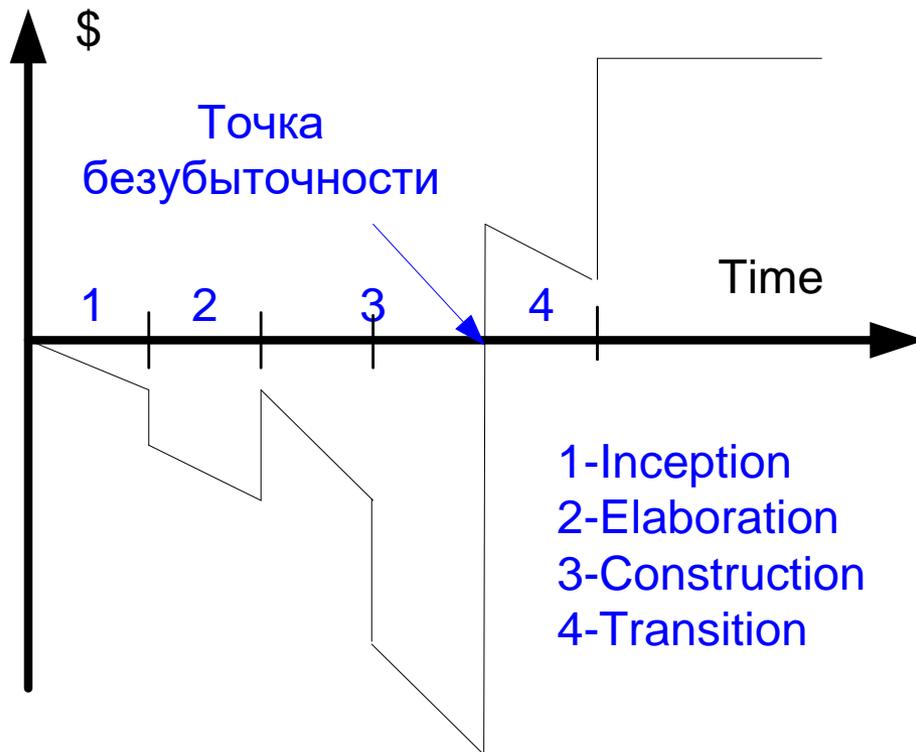
---

- Payback period – период времени до момента, когда чистый поток денежных средств станет положительным
- Обычно используется как грубый способ обоснования и ранжирования проектов
- Обычно учитывает только основные затраты

# Поток денежных средств



# На самом деле ...



**1 деление = 1 месяц**

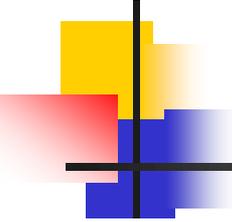
**1** – постоянные расходы, разовые выплаты

**2** – постоянные расходы, разовые выплаты, получение оплаты

**3** – постоянные расходы, разовые выплаты, получение оплаты

**4** - постоянные расходы, разовые выплаты, получение оплаты

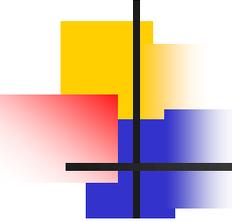
Инвестиции могут быть весьма значительными



# Методы исправления ситуации

---

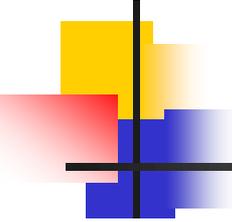
- Кредиты
- Оптимизация расходов
- Предоплата
- Дробление проекта на фазы с проведением приемки-передачи (Acceptance) и полной оплатой фаз



# Пример

---

- Вкладываем в проект по \$1000 в течение 3 лет
- Получаем оплату по \$700 в год, плюс
  - За второй год \$100
  - За третий \$900
- Определите прибыль и точку безубыточности
- Прибыль =  $700 \cdot 3 + 100 + 900 - 3000 = 100$   
(но этот ответ не совсем верен)



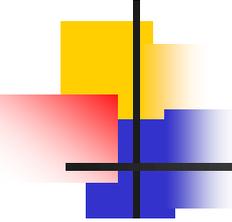
# Почему ответ неверен?

---

- Во что мне обойдутся сегодня \$100, которые я отдам через 2 года?
- Или... Сколько надо иметь сегодня, чтобы через 2 года иметь \$100 ?
- Или... Что я мог бы получить за те \$1000, которые вложил в первый год проекта из нашего примера? Только ли \$100 через 2 года?

# Приведенная стоимость / Present Value

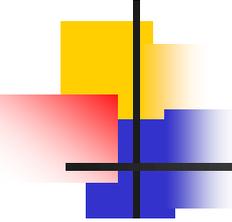
- Время = тоже деньги
- Present Value – стоимость *сегодня* денег, получаемых или выплачиваемых *завтра*
- Допустим, ставка рефинансирования ЦБ РФ по \$ = 10% годовых
- $\$100/1.1/1.1 = \$82.64$
- $\$82.64 = PV \$100$  через 2 года



# Поток денежных средств

---

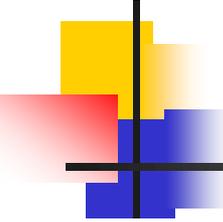
- Cash Flow - показывает приток и отток наличности во времени
- Представлен в виде *приведенной стоимости* денег (дисконтирование)
- Приведенная стоимость (t) = стоимость \*  
учетная ставка (t)
- Идея: Сравнивать сравнимые величины, то есть стоимость денег, приведенную к одному моменту времени



# ROI – возврат инвестиций

---

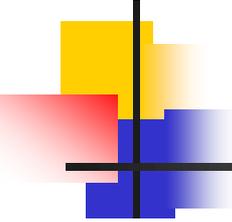
- ROI – Return On Investment
- $ROI = (\text{приток-отток}) / \text{отток денежных средств}$
- Discounted ROI – ROI с учетом дисконтирования



# IRR - Internal Rate of Return

---

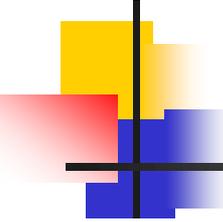
- Внутренний уровень возврата инвестиций
- Отвечает на вопрос: какой уровень дисконтирования даст нулевой чистый поток денежных средств через период (напр. 10 лет)
- Используется для сравнения инвестиционной привлекательности проектов



# Пример

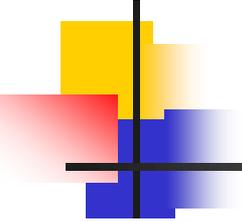
---

- Вкладываем в проект по \$1000 в течение 3 лет
- Получаем оплату по \$700 в год, плюс
  - За второй год \$100
  - За третий \$900
- Определите прибыль, точку безубыточности, ROI, IRR при ставке 13%
- Какую надо получить предоплату, чтобы получить нулевую прибыль?



# Пример

Flow	Discount rate	Present Value
-\$ 300	1.00	-\$ 300
-\$ 200	0.85	-\$ 169.49
\$ 600	0.72	\$ 430.91
\$ 100		-\$ 38.58



---

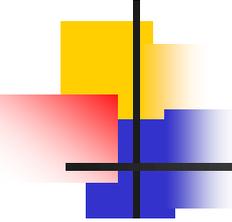
# Контракты и Поставки

Контракт, специфика контракта в ИТ, жизненный цикл контракта, авторские права

Типы контрактов; риски покупателя и продавца

Поставки; общие, форвардные и разобщенные

Делать или купить – критерии выбора

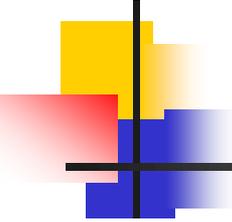


# Контракт (договор) - это

---

- Соглашение между компетентными сторонами
- Для достижения законных целей
- С четкими условиями
- С употреблением действительных обменных единиц

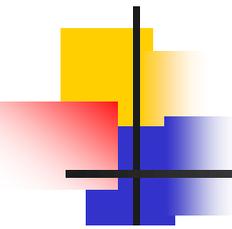
Источник: РМВОК



# Примеры контрактов в ИТ

---

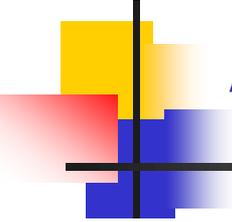
- Контракт на разработку ПО
- Передача части проекта с субподряд
- Аренда помещений
- Аренда техники
- Подключение к Интернет
- Внешний хостинг
- .....



# Специфика контрактов на разработку ПО

---

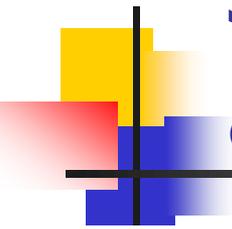
- Трудность определения цены
  - Уникальность услуги
  - Отсутствие «рыночной цены товара»
- Трудность фиксации соответствия объекта поставки спецификациям
  - Частично разрешается SRS, Test Plan, Acceptance Plan
- Лицензии на компоненты
- Авторское право
  - Требуется согласования процедуры передачи прав



# Авторские права

---

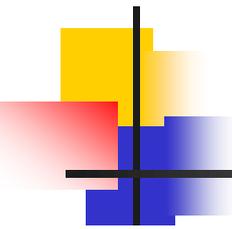
- Неимущественные права
  - Право на имя (неотторжимо)
- Имущественные права (права на воспроизведение, использование, передачу, модификацию)
  - Исключительные
  - Неисключительные



# Законодательство РФ об авторском праве

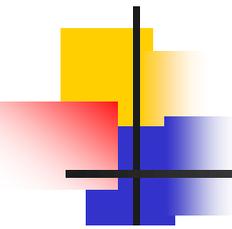
---

- Гражданский кодекс РФ, часть 4я
- Охрана прав разработчика и заказчика ПО
- ПО охраняется как литературное произведение



# Типичное содержание контракта на разработку ПО

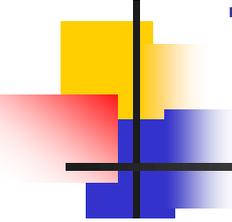
- Предмет договора
  - Обычно ссылается на техническое предложение
- Цена и порядок расчетов
  - Фиксирует цену и моменты оплаты с привязкой к поставкам
- Права и обязанности сторон
  - Отчетность, привлечение 3-х лиц, ...
- Ответственность сторон
  - Штрафы, пени ...
- Авторские правоотношения
  - Порядок передачи имущественных прав
- Приемка-передача ПО
  - Порядок и условия подписания акта приемки-передачи (Acceptance procedure)
- Действие договора
  - Порядок вступления в силу и порядок расторжения



# Жизненный цикл контракта на разработку ПО

---

- Техническое предложение
- Согласование и подписание контракта
- Выполнение задач проектного плана
- Передача объектов поставки
- Приемочное тестирование
- Подписание акта приемки передачи
- Передача исходных текстов
- Выставление счета (invoice)
- Получение оплаты



# Типы контрактов

---

- Fixed Price (FP)
  - Firm Fixed Price (FFP)
  - FP + Economic Adjustment (FP+EA)
  - FP + Incentive Fee (FP+IF)
- Time & Material (T&M)
  - Cost + Fixed Fee (TM + FF)
  - Cost + Incentive Fee (TM + IF)

# Риски заказчика и исполнителя

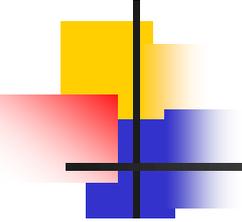
Заказчик



FFP	FP+IF; FP+EA	TM+EA	TM+IF	TM+FF
-----	--------------	-------	-------	-------



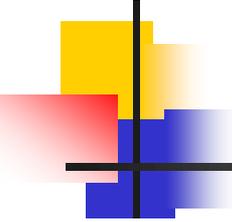
Исполнитель



# Закупки

---

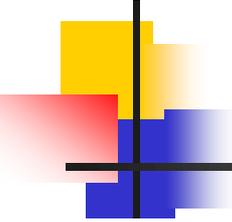
- Сделать или купить? Критерии выбора:
  - Наличие свободных ресурсов
  - Наличие/отсутствие навыков
  - Необходимость в спец. оборудовании
  - Желание контролировать процесс
  - Секретность проекта



# 3 стратегии закупок

---

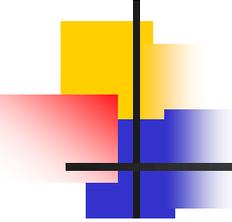
- Форвардные (Forward Buying)
- Общие (Blanket Orders)
- Разобщенные (Splitting Orders)



# Forward Buying

---

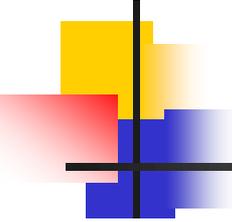
- Стратегия «закупать в избытке по отношению к требуемому минимуму»
- Предохраняет от недостачи
- Скидки на количество
- Защита от роста цен
- «-» замораживание средств в пассивах
- Пример в IT: лицензии



# Blanket Orders

---

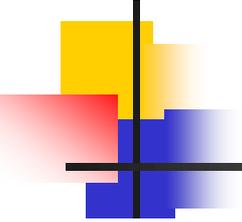
- Долгосрочный заказ вперед
- «+» Скидка на количество
- «-» Заказанный ресурс может не потребоваться в полном объеме
- Пример в IT: «аренда» команды разработчиков (outsourced IT department)



# Splitting Orders

---

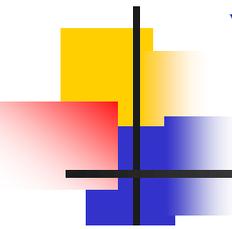
- Проекту требуются некие критические компоненты
- Известно, что поставщик имеет 90% вероятность поставить компонент вовремя
- Что даст нам выбор еще одного (второго) поставщика, который также имеет 90% вероятность своевременной поставки?
- «+» 99% вероятность получить компонент в срок
- «-» более высокая цена



---

# Управление проектной конфигурацией

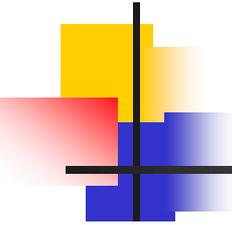
СММІ об управленіи конфигурацией  
Управление требованиями: Источники  
неприятностей (обзоры Standish group), US AirForce



# Управление конфигурацией

---

- Что такое конфигурация? Почти все результаты работ (work products)
- Требования
- Архитектурные решения
- Исходные тексты, используемые библиотеки, их версии
- Средства разработки и их настройка
- Объекты поставок, документация



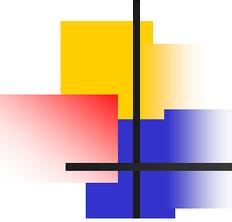
# Конфигурация ИТ проекта

---

- Конфигурация - это:
  - исходные тексты, документы, модели
  - требования и их изменения
  - артефакты проектной деятельности
  - используемые библиотеки
  - конфигурация средств разработки
  - правила кодирования и документирования кода
- Конфигурация подлежит управлению:
  - управлению версиями и изменениями, в т.ч версиями самой конфигурации
  - контролю за соблюдением правил

Ниже приведены основные постулаты конфигурационного управления по СММ (дословный перевод требований):

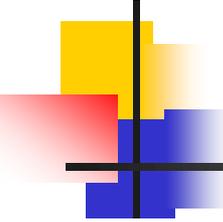
- любые действия по направлению конфигурационного управления заранее запланированы;
- любые программные работы идентифицированы, управляются и являются общедоступными;
- любые изменения в продукте являются управляемыми; заинтересованные группы и индивидуумы постоянно информируются о состоянии развития проекта.



# Технические средства

---

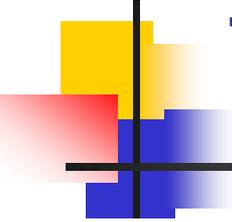
- Системы контроля версий
  - RCS (1982г), CVS (1986г) – устарели
  - SVN (2004г) – вышел из моды, но еще используется
  - Git (GitHub, GitLab) – наиболее распространен
  - Коммерческие: Team Foundation Server (Microsoft)
- Системы контроля требований и инцидентов (ITS – Issue Tracking System)
  - Rational ClearQuest (IBM) – позиционируется как CRM – Customer Relationship Management
  - JIRA – недорогая коммерческая ITS
  - Redmine - свободная ITS
- Облачные платформы
  - Напр., Microsoft Azure содержит все в одной платформе: контроль версий, ITS, CI, развертывание (deployment), средства анализа логов, health check и т.д



# Терминология: контроль версий

---

- Репозиторий – место хранения артефактов проекта в системе управления версиями
- Релиз / release – любая версия ПО, вышедшая за пределы проектной команды; подлежит обязательному присвоению номера версии и тега
- Ветка / branch – версия ПО, начавшая изменяться параллельно с текущей версией
- Тег / tag – моментальный снимок состояния репозитория

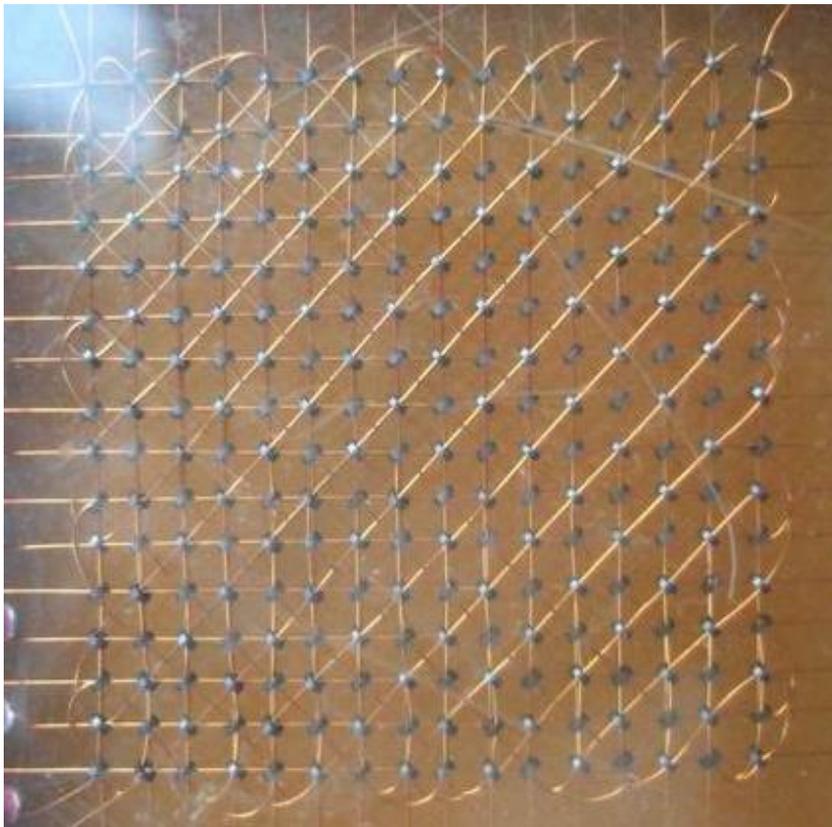


# Терминология

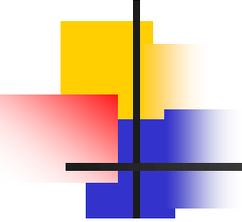
---

- Требование / requirement – любое функциональное или нефункциональное требование
- Изменение / change – изменение исходного требования
- Уточнение / adjustment – уточнение требования, не влияющее на оценку проекта
- Ошибка / bug – подтвержденная ошибка
- Тикет / Ticket, Issue – проблема, нерешенный вопрос; может стать ошибкой, изменением, уточнением или новым требованием

# Термин Bug (жук)



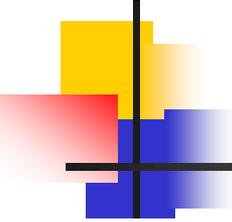
- Есть мнение, что термин возник из-за настоящих жуков, иногда попадавших на проволочки памяти на ферритовых кольцах, и мешавших считывать информацию



---

# Управление требованиями

Управление требованиями  
Источники неприятностей (обзор Standish group), US  
AirForce project

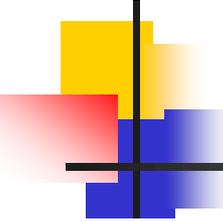


# 1995 – CHAOS report

---

- Тип 1й: 16.2% проектов завершаются вовремя и в рамках бюджета (в крупных компаниях – только 9%)
- Тип 2й: 31.1% проектов остановлены до планового завершения
- Тип 3й: 52.7% завершены с неполной функциональностью, и стоили в сумме 189% от своих начальных оценок

Источник: Standish Group CHAOS report, 1995, (обследовано 8380 проектов)

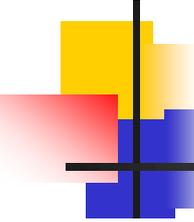


# Причины остановки (тип2)

---

- Недостаточное взаимодействие с пользователями - 12,8%
- Неполнота требований и спецификаций - 12,3%
- Изменение требований - 11.8 %
- Нереалистичность ожиданий - 5.9%
- Новизна технологии (для сравнения) - 3.7%

Источник: Standish Group CHAOS report, 1995, (обследовано 8380 проектов)

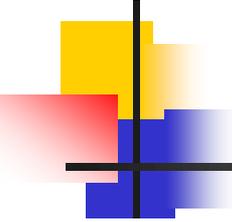


# Причины перерасхода (тип3)

---

- Недостаточное взаимодействие с пользователями - 12.4%
- Неполнота требований и спецификаций - 13.1%
- Нереалистичность ожиданий - 9.9%
- Изменение требований - 8.7 %
- Новизна технологии (для сравнения) - 3.7%

Источник: Standish Group CHAOS report, 1995, (обследовано 8380 проектов)



# US Air Force project

---

## Источники проблем

- **Требования** - **41%**
- Архитектура - 28%
- Ошибки в данных - 6%
- UI - 6%
- Системное окружение - 5%
- Ошибки людей - 5%
- Документация - 2%
- Другие - 7%

- Sheldon F., “Reliability Measurement from Theory to Practice”, 1992

# Стоимость исправления ошибок в требованиях по фазам

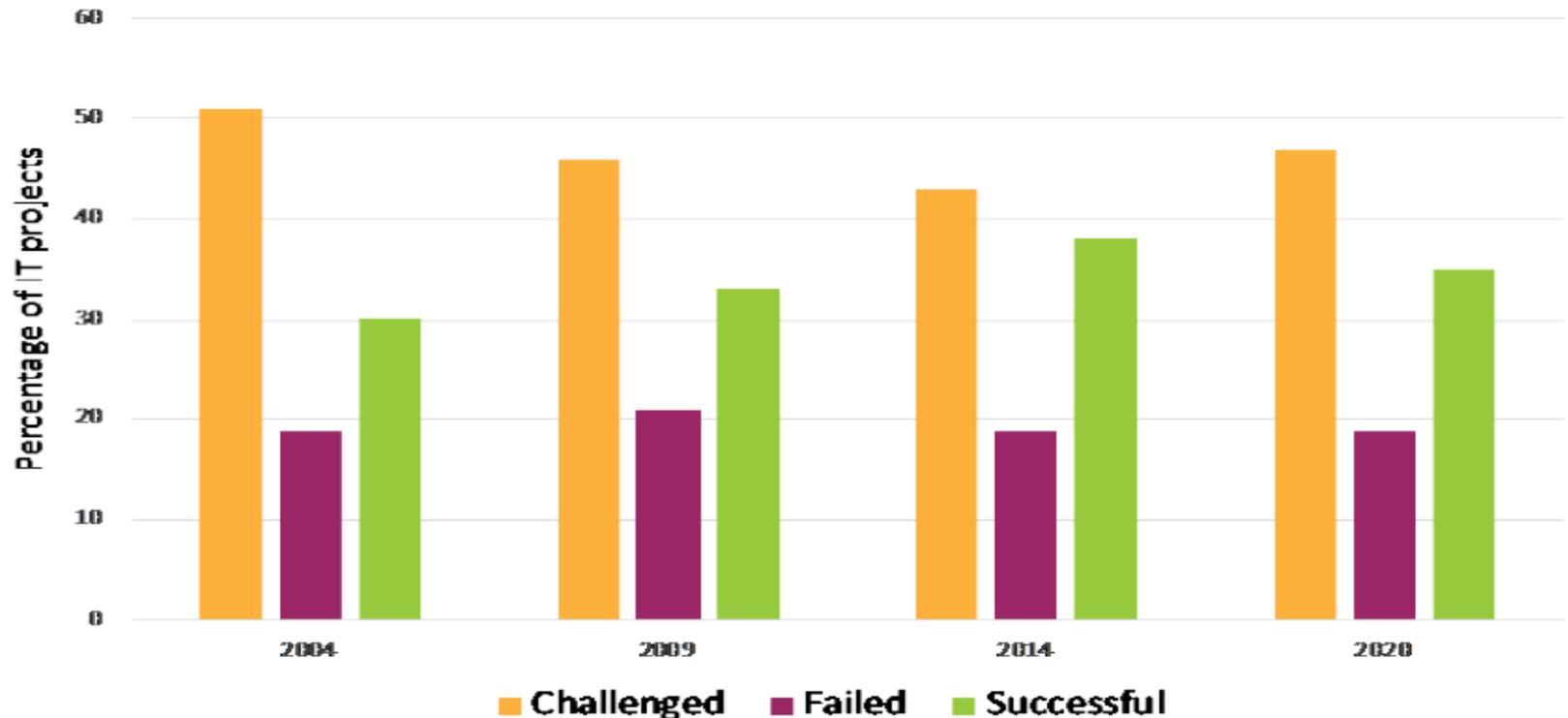
- Начало проекта - 0,1 – 0,2
- Проектирование - 0,5
- Кодирование - 1
- Юнит-тестирование - 2
- Приемка-передача - 5
- Поддержка - 20



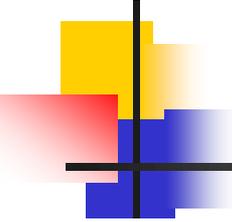
- Davis A., “Software Requirements – Objects, Functions and States”, 1993

# CHAOS-2020 – лучше не стало

IT Project Outcomes  
Based on CHAOS 2020: Beyond Infinity Report



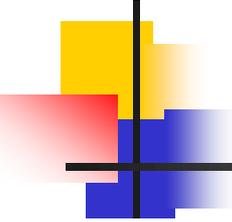
Herb Krasner, Consortium for Information & Software Quality (CISQ)  
"The Cost of Poor Software Quality in the US: A 2020 Report", p.14



# В чем проблема?

---

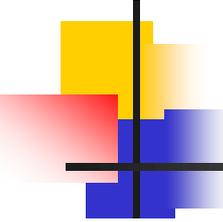
- Требования обсуждаются *вербально*
- Документируются на *естественном языке*
- Сложно разделить «изменение» и «уточнение» требований
- Неоднозначности трактуются в пользу клиента



# Как решать?

---

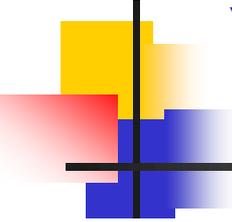
- Управление требованиями
  - Систематический подход к выявлению, структуризации и документированию требований
  - Четкий процесс изменения требований к программному решению



# Управление требованиями

---

- Документирование требований
  - SRS – главный документ; описание и приоритеты требований
  - функциональные, нефункциональные требования и требования к процедуре приемки
  - прототипы, их утверждение заказчиком
- Контроль требований и их изменений
  - статус по требованиям ( (НЕ)реализовано /изменено)
  - база данных изменений
  - стоимость каждого изменения определяется и утверждается ДО того, как изменение принято к исполнению
- Метризация процесса управления требованиями
  - общее количество изменений
  - количество изменений, принятых/Непринятых к исполнению
  - стоимость изменений



# Управление изменениями

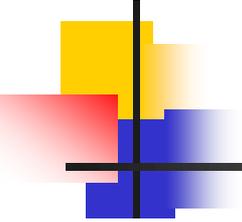
---

Хороший Менеджер может выиграть тендер за

**\$ 0,00**

и выполнить проект в точном соответствии с контрактом.

(шутка с долей шутки)



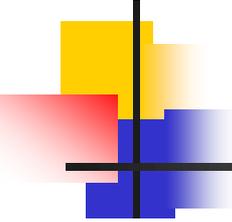
# Управление IT проектами

---

## Управление ресурсами. Модели организаций

Проектная, Функциональная, Матричная  
модели.

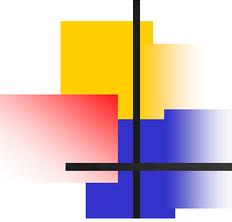
Преимущества и недостатки



# Модели организаций

---

- Проектная
- Функциональная
- Матричная

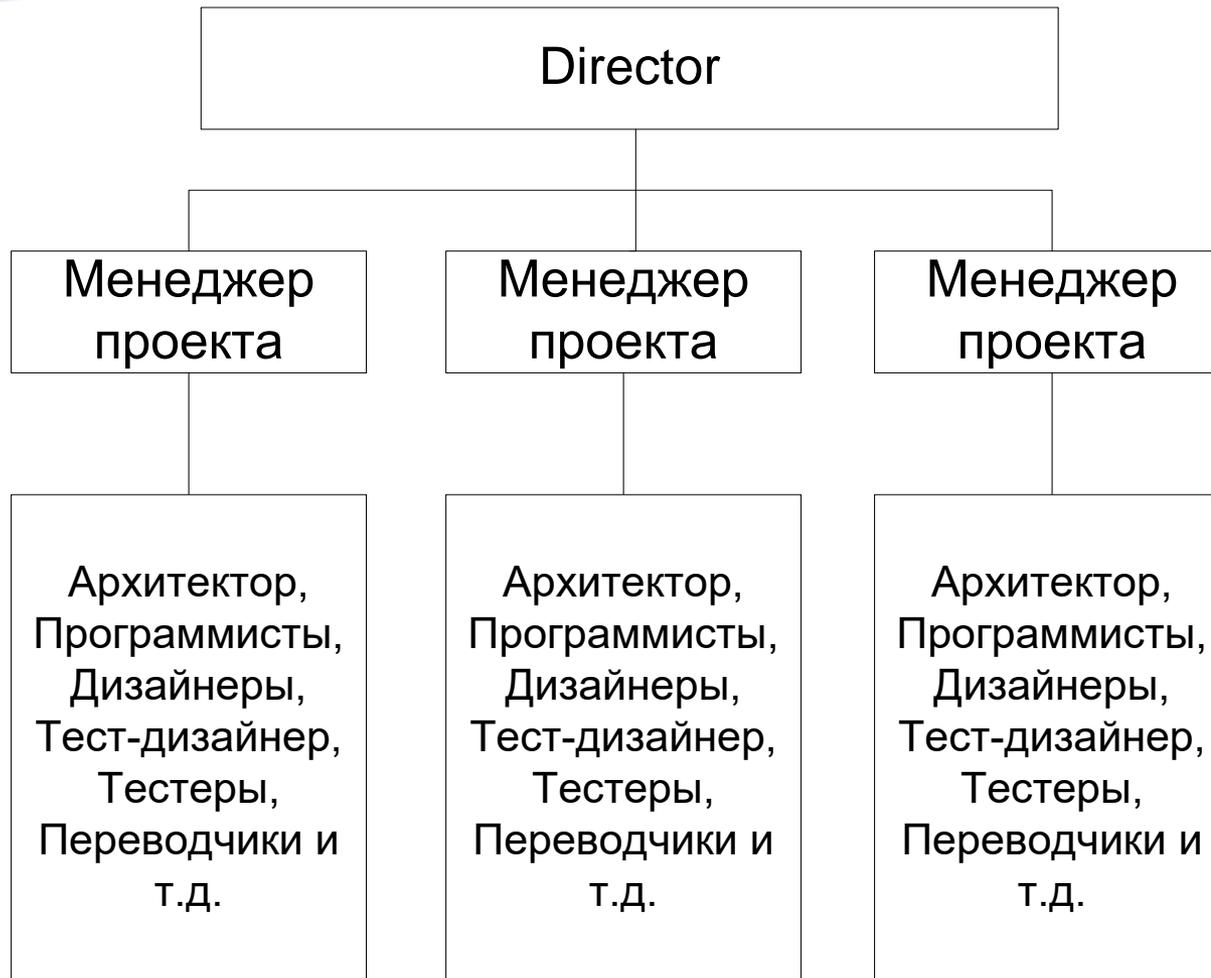


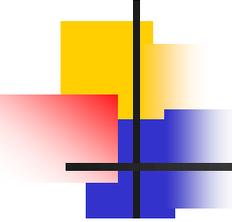
# Проектная организация

---

- Полная власть проектного менеджера
- Ресурсы распределены по проектам
- На период проекта – высокая мотивация
- Что будет с людьми по окончании проекта?
- Обычно невелика по размеру
- Сложно обеспечивать общие процессы, такие как контроль качества, аттестацию и развитие персонала

# Пример структуры проектной организации



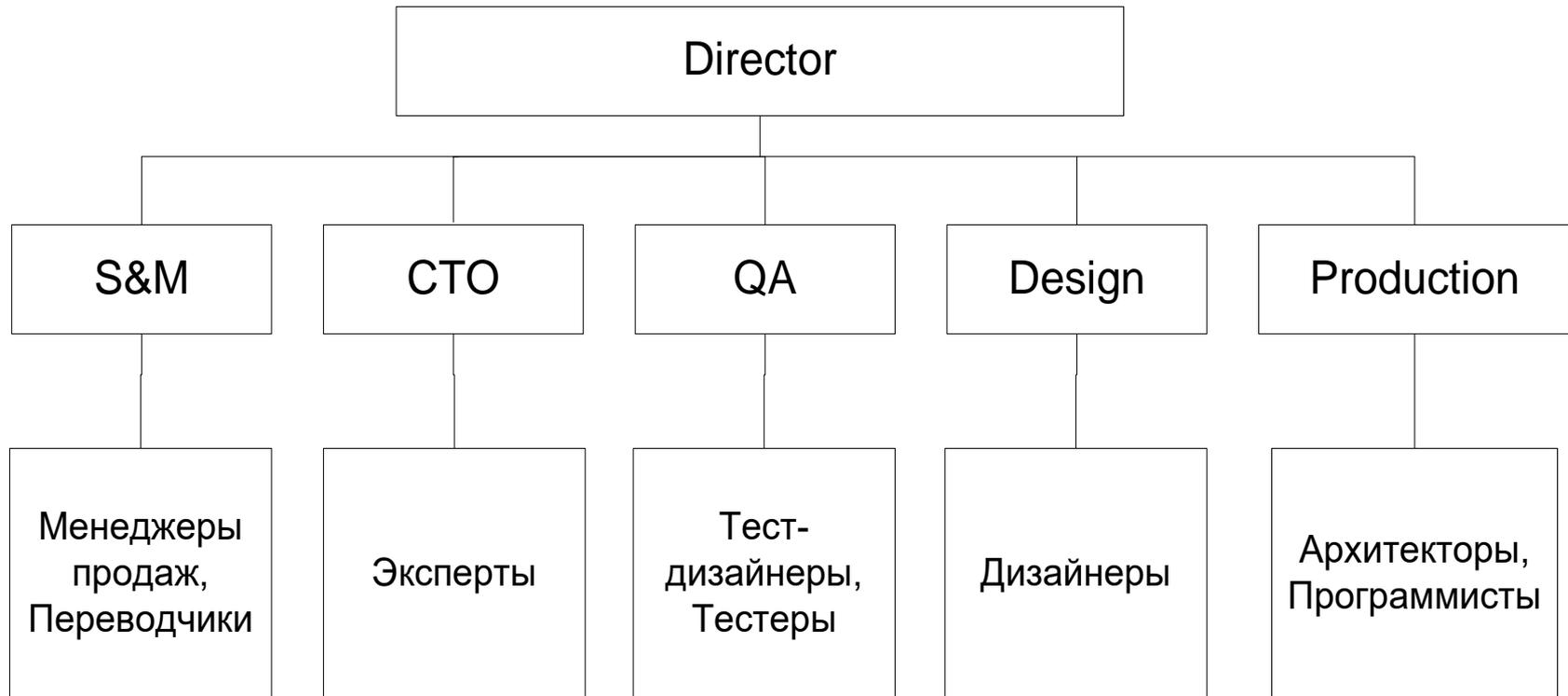


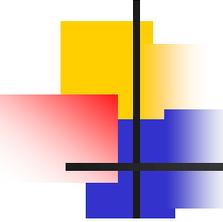
# Функциональная организация

---

- Фокусировка на навыках (обыкновенно - по отделам)
- Функциональный менеджер хорошо знает людей
- Очень стабильная организация
- Один большой босс, много функциональных менеджеров
- Недостаточно внимания на конкретные проекты
- Затруднен переход из проекта в проект, ресурсы распределены скорее по функциональным отделам, чем по проектам
- Свойственна большим организациям

# Пример структуры функциональной организации



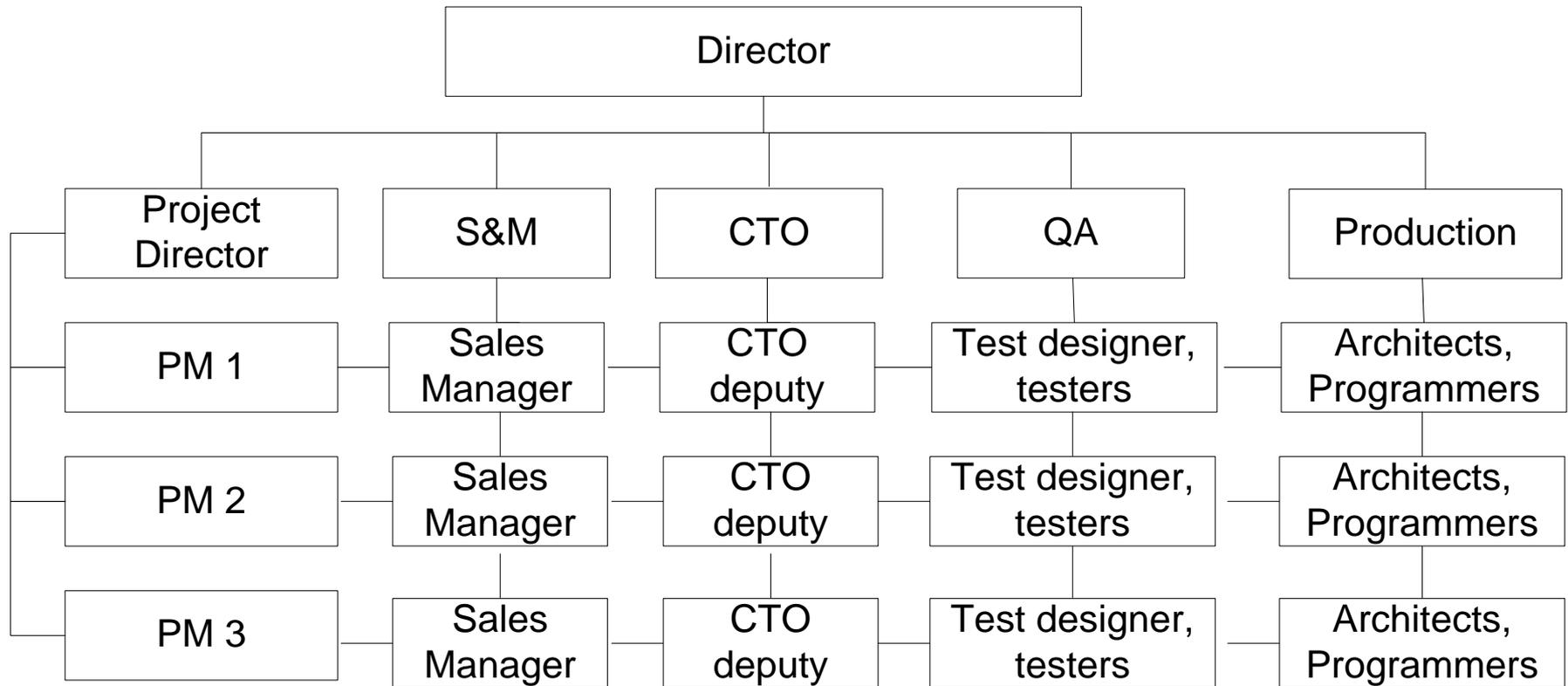


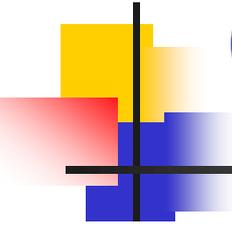
# Матричная организация

---

- Характеристики проектной и функциональной одновременно
- Общие ресурсы, люди входят и выходят из проектов легко и по необходимости
- Много менеджеров проектов
- Сложные коммуникации
- Проектно-гибкая структура, эффективно поддерживает много проектов
- Необходимо иметь больше менеджерских кадров
- Высокая проектная мотивация сотрудников

# Пример матричной структуры

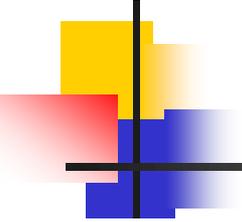




# Орг. структура и проект

<b>Тип организации Характеристика</b>	<b>Функциональная</b>	<b>Матричная</b>	<b>Проектная</b>
<b>Ответственность менеджера</b>	Мала	Средняя- Высокая	Полная
<b>Занятость РМ в проекте</b>	Частичная	Полная	Полная
<b>Процент* персонала проекта, занятого в нем на 100%</b>	Менее 50%	50-95%	85-100%

\* Не стоит принимать эти цифры как абсолютные, они дают лишь качественную картину

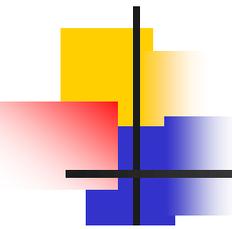


---

некоторые  
«управленческие»  
шаблоны\*

Как лучше **НЕ ДЕЛАТЬ**

\* Выполнено «профессионалами», не пытайтесь повторить

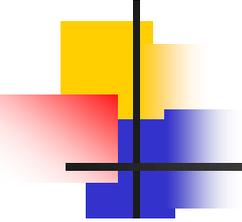


# Манипуляция инстинктами

---

**Если к среде это не заработает – ВСЕ уволены!!!**

- Попытка переложить ответственность за собственные ошибки (планирование, управление рисками) на команду
- Иногда срабатывает, мобилизуя остатки деморализованной команды, при условии, что в ней есть неформальный лидер, способный взять все на себя (по сути, выполнить работу менеджера)
- Непременно имеет отдачу в виде падения авторитета и снижения мотивации

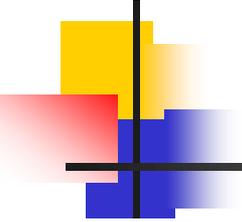


# Манипуляция инстинктами

---

**Вариант: Делайте к сроку или останетесь без премии!**

**ВОПРОС: а как стоило бы подать эту мысль?**



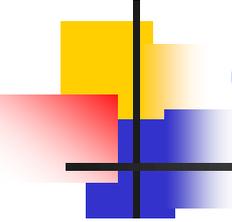
# Road to hell

---

**Делайте ТАК, как Я сказал!**

**Варианты: КТО тут главный?**

- Менеджера «можно понять», у него дедлайн и нервы
- Иногда *любое* решение лучше, чем *никакое*
- Но:
  - Если люди возражают или сомневаются – возможно, это реальная проблема (риск)
  - Уход от обсуждения проблему не решает, а риски - однозначно увеличивает



# Сам начальник – сам дурак

---

**Я – менеджер проекта, я не пишу код, это ваша задача – сделать правильный код!**

- Менеджер проекта изначально и по определению отвечает **за все**
- Менеджер проекта
  - **Делегирует** часть своих полномочий членам команды
  - Организует своевременный текущий **контроль** за исполнением делегированных задач
- Делегирование не снимает ответственности с менеджера