

# Технология программирования

Курс лекций для групп 3057, 4057

## Лекция №3

Лектор: проф. И.Штурц

2010 г.

# Содержание

1. Введение
2. Модели жизненного цикла ПП
3. Модели команды разработчиков
4. Управление проектами
5. Коммуникация и документирование
6. Структуризация ПП: модульное и компонентное программирование
7. Языки и методы проектирования
8. Тестирование и верификация ПП
9. CASE-системы
10. Надежность ПП, ее оценка и меры по ее повышению
11. Стандарты качества технологии программирования

# Команда разработчиков – Project team

- Как распределить обязанности (роли) исполнителей?
- Как организовать их взаимодействие?
- Как распределить полномочия и ответственность за результаты?

Модель команды – абстрактное описание ролевых функций и связей между участниками проекта

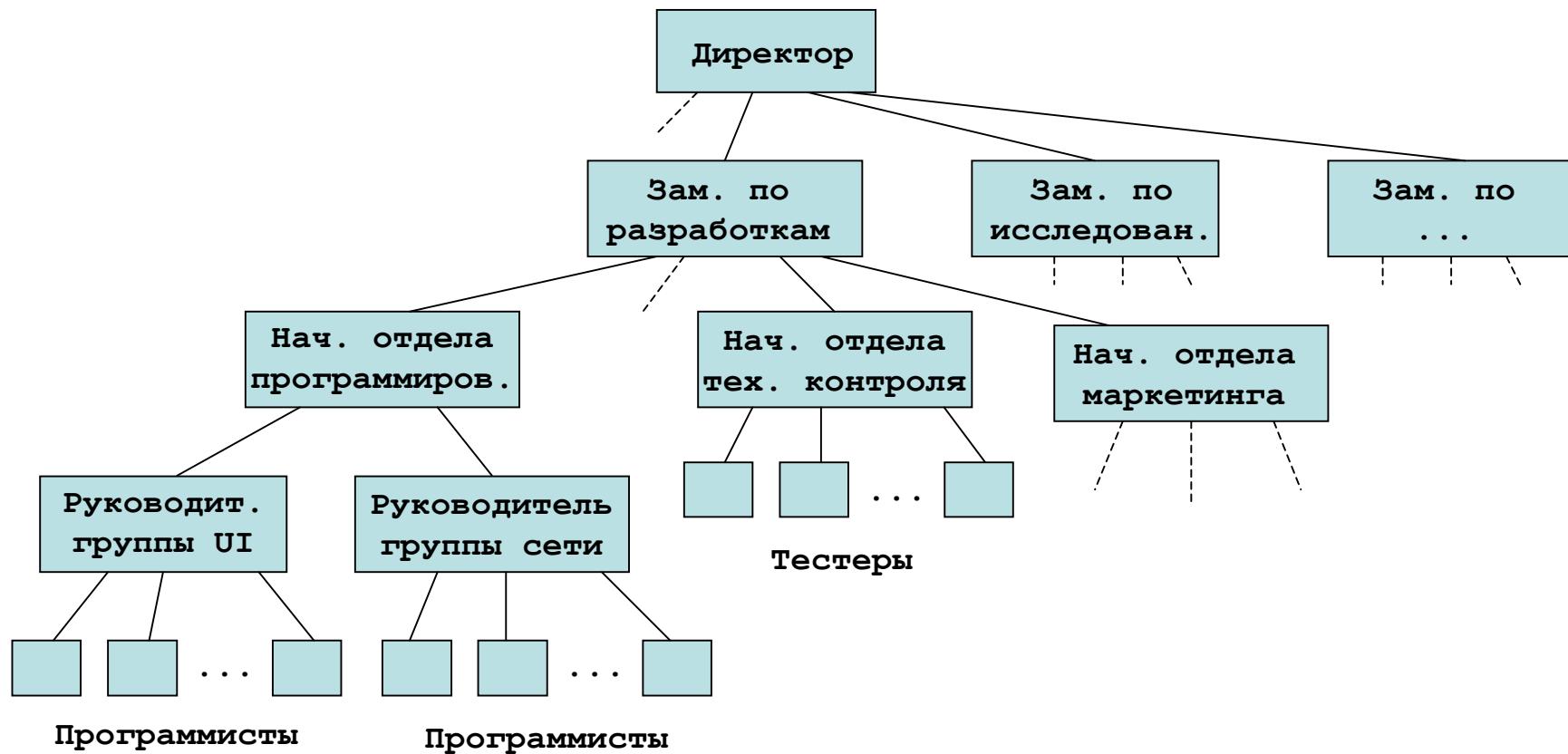
Модель команды - методическая основа для разработки должностных инструкций и формализованных процедур, связанных с подбором и расстановкой кадров, вовлеченных в процесс разработки

Основные типы моделей команды:

- ❖ Иерархическая
- ❖ Бригада главного программиста
- ❖ Модель равных (MSF, Agile)

# Должностная структура организации

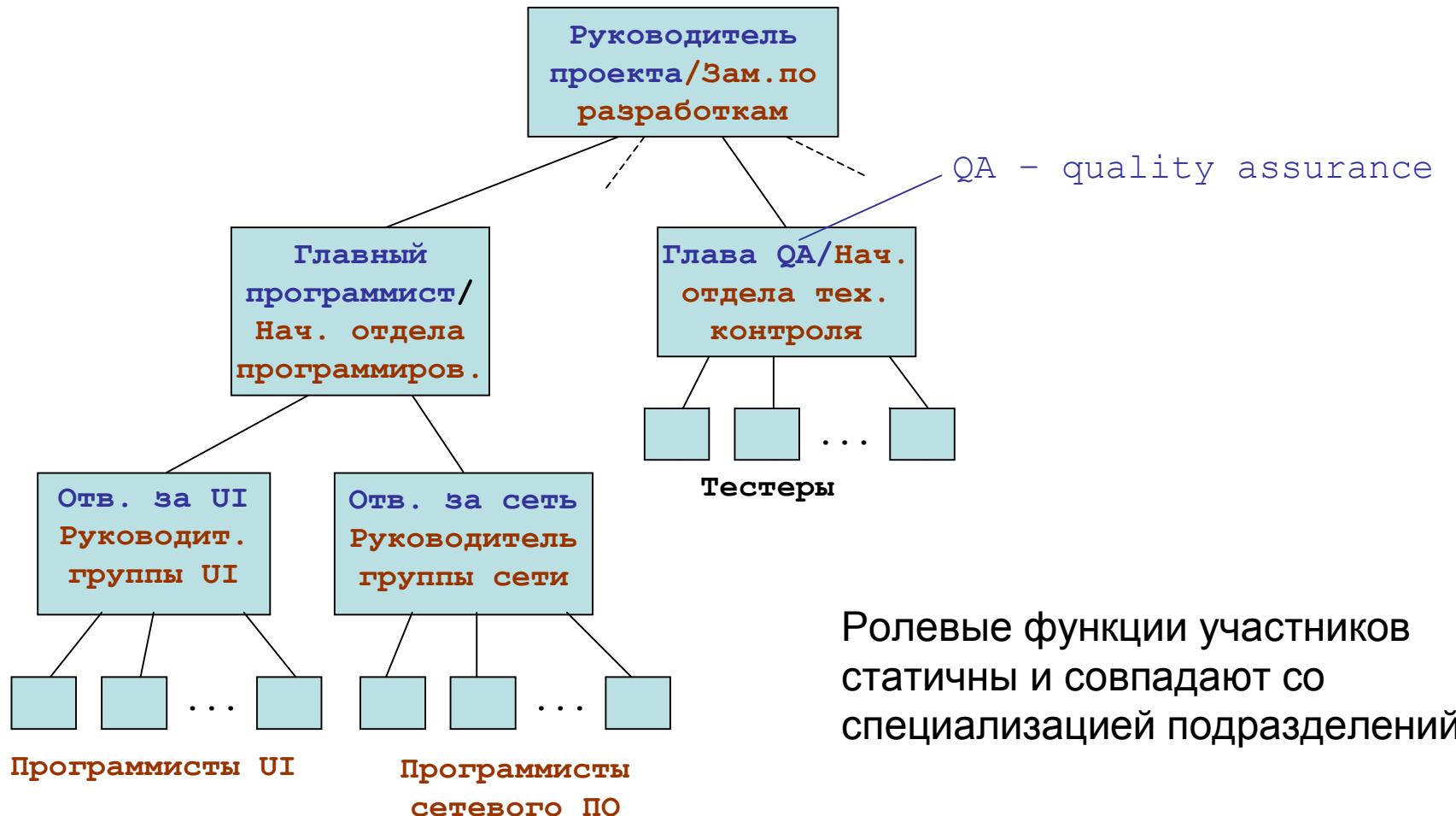
Определяет иерархическое отношение «руководитель-подчиненный»



Подразделения имеют определенную специализацию

# Иерархическая модель команды

- изоморфна фрагменту должностной структуры:



Ролевые функции участников  
статичны и совпадают со  
специализацией подразделений

# Достоинства иерархической модели

- Высокая степень надежности, устойчивости и управляемости команды благодаря принципу единоначалия
- Хорошая масштабируемость команды: от десятков до тысяч исполнителей
- Иерархическая модель привычна и известна абсолютно всем

## Проблемы

- Узкий специалист занят на конкретном проекте не 100% времени
- Потребность в исполнителях разных ролей переменна в ходе проекта
- Несколько проектов выполняются параллельно

## Решения

- Исполнитель – широкий специалист - совмещает несколько ролей или переключается между ними
  - нарушаются принцип единоначалия
- Исполнитель – узкий специалист - работает в нескольких командах проекта параллельно

Оба решения требуют повышенной квалификации исполнителя

# Недостатки иерархической модели

- Экстенсивность: наращивание функциональности – только увеличением штатов
- Недостаточная гибкость в условиях изменений требований
- ! Руководители подразделений вынуждены выполнять два различных вида обязанностей:
  - административное управление персоналом
  - принятие технических решений по проекту

Эти два вида деятельности требуют разных способностей, которые редко совмещаются в одной голове

Начальниками часто становятся талантливые программисты, производительность и качество работы которых на порядок выше среднего,

**-различие в 10:1 (вплоть до 25:1) характерно для профессии программиста, в отличие от землекопа**

и начинают заниматься не своей работой (частный случай закона Паркинсона ☺)

**Ф.Брукс: "Сожгите 275 трупов, если у вас 25 талантливых начальников руководят 300 программистами"**

# Модель главного программиста (1970)

Главный программист выполняет весь проект сам, а прочие члены команды ассициируют ему в рамках своих ролевых функций

**Аналогии: хирургическая бригада или экипаж самолета**



**Главный программист** - проектирует и пишет весь код  
**Второй пилот** – дублер Главного программиста: участвует в проектировании и кодировании, но не несет ответственности за код  
**Администратор** – управление проектом и связь с внешним миром  
**Секретарь** – помощник администратора и хирурга: внешняя переписка  
**Редактор** – документирование  
**Контролер** – тестер / верификатор  
**Инструментальщик** – технологическая поддержка  
**Архивариус** – ведение версий

# Модификации модели

1. Изменение количества и ролей ассистентов
  - Неизменны три роли: Главный программист, Второй пилот и Администратор, остальные зависят от технологии и могут совмещаться одним участником
2. На роль Главного программиста назначается не кодировщик, а системный аналитик или менеджер продукта
  - Кодирование ведет Второй пилот, но все принципиальные решения принимает Главный программист
3. «Сдвоенный центр» - позволяет немного масштабировать бригаду
  - Имеются два Главных программиста, которые делят проект пополам, все решения принимают консенсусом и являются вторыми пилотами друг для друга

## Достоинства модели

- ✓ Высокая производительность и качество работы, т.к. Главный программист освобожден от административных и вспомогательных функций, но принцип единоличия сохранен
- ✓ Хорошая адаптивность к изменениям
- ✓ Небольшая численность участников → устная коммуникация

# Недостатки модели главного программиста

- Ограниченнaя масштабируемость
  - Модель отлично работает на небольших проектах:
    - с объемом не более  $10^5$  строк исх. кода, или
    - с трудоемкостью 6-8 человек  $\times$  1-2 года
  - Если проект требует более коротких сроков или существенно больших объемов, то его приходится разбивать на части для нескольких таких команд,
    - но поскольку части не могут быть полностью автономны, проблемой (социально-психологической) является координация работы лидеров
- Не распараллеливаемая структура - действует по принципу: один проект - одна команда
- Имеется уязвимое центральное звено
  - велик риск мгновенной аннигиляции команды, если что-то случается с хирургом, хотя Второй пилот и подстраховывает ситуацию

# Модель команды равных на примере MSF

Нет управлеченческой иерархии, руководство – коллективное  
Все роли равноправны и связаны друг с другом



# Роли в команде MSF

Роль в команде	Целевая установка; ответственность
Управление продуктом	Удовлетворение требований заказчика, коммуникация между командой и заказчиком
Управление программой	Управление проектом: планирование сверху-вниз" и контроль сроков выполнения
Разработка	Проектирование и кодирование, планирование "снизу-вверх"
Тестирование	Выявление ошибок; сборка продукта и тестирование
Удовлетворение потребителей	Обучение и улучшение продуктивности пользователей; посредничество между командой и пользователями
Управление выпуском	Внедрение продукта; посредничество между командой и службами продаж, рекламы, поддержки и т.д.

# Внешние коммуникации команды MSF



# Масштабирование вниз (для маленьких проектов)

- совмещение нескольких ролей одним сотрудником, с ограничениями:

	Управление продуктом	Управление программой	Разработка	Тестирование	Удовлетворение потребителя	Управление выпуском
Управление продуктом		-	-	+	+	±
Управление программой	-		-	±	±	+
Разработка	-	-		-	-	-
Тестирование	+	±	-		+	+
Удовлетворение потребителя	+	±	-	+		±
Управление выпуском	±	+	-	+	±	

⊕ Допустимо

± Нежелательно

- Нельзя

Например, управление продуктом и управление программой в одном лице несовместимы, а разработка несовместима ни с одной из других ролей

# Масштабирование вверх (для больших проектов)

Несколько способов:

1. Каждый элемент в схеме команды – это *ролевой кластер*: несколько человек, тоже колесико из нескольких элементов
2. Команда разбивается на подкоманды двух типов:
  - команды свойств (feature teams)
    - ответственны за определенные наборы функциональных возможностей продукта
  - функциональные команды (function teams)
    - существуют в пределах ролей (например, подкоманды разработки)Эти подкоманды общаются через своих выделенных представителей
3. Вводятся элементы иерархии: группы свойств с усеченным составом ролей подчиняются руководящей группе  
(см. следующий слайд)

# Команда MSF с элементами иерархии



# Пример структуры команды для большого проекта

# Формирование команды проекта

Три возможных типа исполнителей:

- **Лидер.** Человек, который стремится управлять другими людьми, проектами, для которого нестерпимо быть просто «винтиком» в сложном механизме
- **Технарь.** Человек, который получает настоящее удовольствие от самого процесса поиска решения задачи
- **Общительный.** Эти люди разносят информацию, популяризуют результаты других людей

Хороший коллектив должен представлять собой удачное сочетание исполнителей всех трех типов

Специальная наука: *групповая динамика* – отрасль социальной психологии, изучающая совокупность психологических процессов в малых группах

# Обсуждение модели команды равных

## Преимущества модели

- Высокая производительность, т.к. непроизводительные трудозатраты на поддержание формальных связей сведены к минимуму
- Сравнительно легкая масштабируемость
- Сильная положительная мотивация труда и высокая заинтересованность всех участников в конечном успехе

## Недостатки модели

- Для формирования команды нужны равные (равно квалифицированные и равно заинтересованные) участники
- Критическое значение имеет коммуникабельность (большая часть коммуникаций неформальны), умение и готовность работать в коллективе («артельный дух»)

Поэтому такая модель не подходит рядовым фирмам с программистами средней квалификации, каковых (фирм) большинство

Им приходится более жестко регламентировать структуру команды, приближая ее к иерархической

# Резюме по моделям команды

- Иерархическая модель используется для выполнения больших рутинных проектов слабо мотивированными исполнителями средней квалификации
- Модель Главного программиста успешно применяется для небольших, но сложных проектов при условии, что команду возглавляет талантливый «харизматический» лидер. Модель идеально подходит для инновационных старт-апов
- Большинство организаций стремятся в той или иной степени приближать команды проекта к команде равных, как наиболее продуктивной модели

# Задачи управления проектом ПП

*Проект (project)* – это ограниченная временными рамками деятельность, цель которой состоит в создании уникального продукта или услуги

*Управление проектами (project management)* – это область знаний, навыков, инструментария и приемов, используемых для достижения целей проектов в рамках согласованных параметров объема, качества, бюджета, сроков и прочих ограничений

## Задачи / функции:

- ❖ Управление верхнего уровня: принятие общих решений о технологии, запуске/остановке проекта, принципиальных изменениях в нем
- ❖ Планирование и распределение ресурсов
  - Материальных: аппаратура, софт, инфраструктура
  - Человеческих: специалисты - участники разработки
  - Временных: календарное планирование – задачи и сроки их выполнения
- ❖ Контроль выполнения календарного плана и качества проекта

# Треугольник проекта

Метафора компромисса между тремя факторами: «много-быстро-дешево»



**Деньги** – бюджет проекта

**Время** – продолжительность проекта

**Область охвата или действия (scope)** – диапазон функциональных возможностей ПП

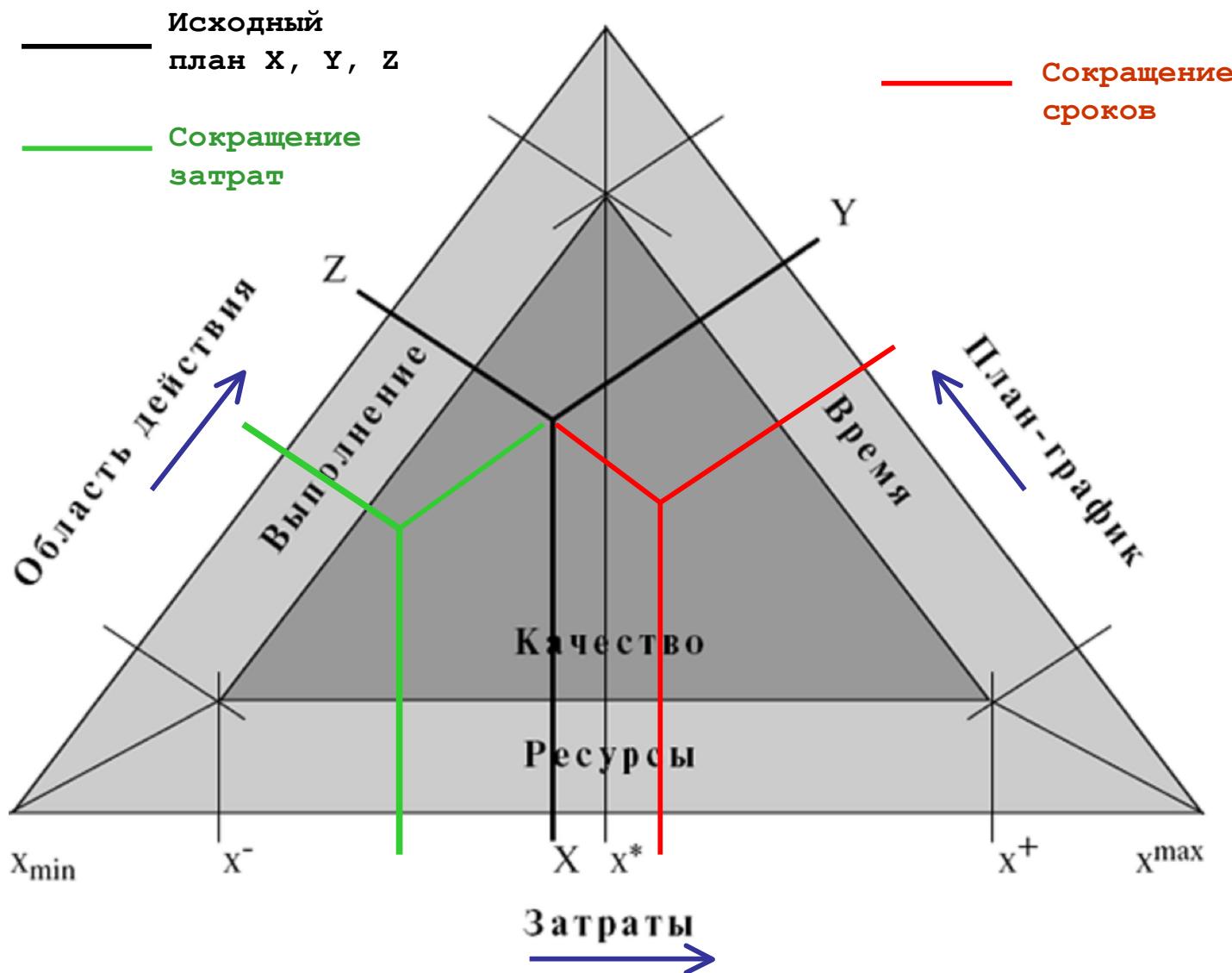
**Качество ПП** – производный фактор

- Одна или две стороны проекта обычно фиксированы
- Треугольник символизирует зависимость между сторонами
  - Напр., при фиксированном бюджете сокращение времени потребует сужения области охвата и, соответственно, ухудшения качества ПП

Wideman R. M.

Project management Body of Knowledge (PMBOK),  
Project Management Institute, PA, 1987.

# Треугольник компромиссов проекта (по РМВОК)

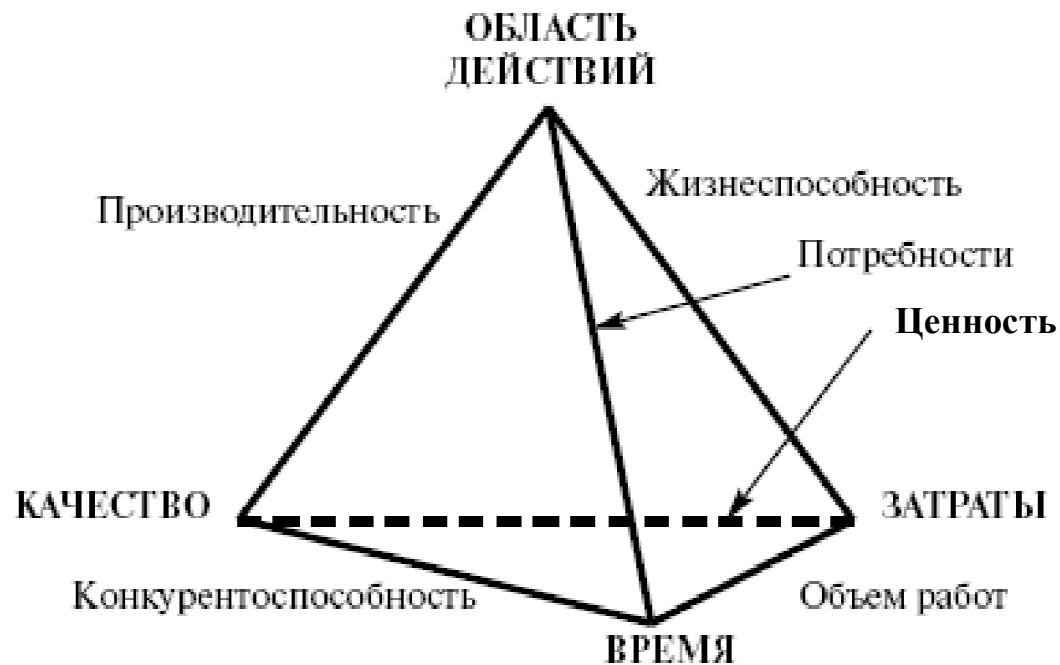


# Звезда компромиссов проекта (по РМВОК, 1987)



Качество стало четвертым независимым параметром, но интерпретация  
(т.е., выбор вариантов) перестала быть наглядной

# Пирамида проекта



Объем пирамиды – инвариант

Интерпретация компромиссов: перпендикуляры из выбранной  
точки внутри пирамиды к ее ребрам

# 5-параметрическая пирамида проекта

(Д.Мараско. IT проекты: фронтовые очерки. 2008)



**Охват = функциональные возможности ПП**

**Качество = уровень показателей качества ПП**

**Скорость = работа в единицу времени**

**Экономичность:** тем больше тратится ресурсов, тем короче эта сторона

С увеличением площади основания растет сложность, рискованность и прибыльность проекта

**Высота пирамиды – абстрактное представление вероятности успеха проекта – величина, обратно пропорциональная риску**

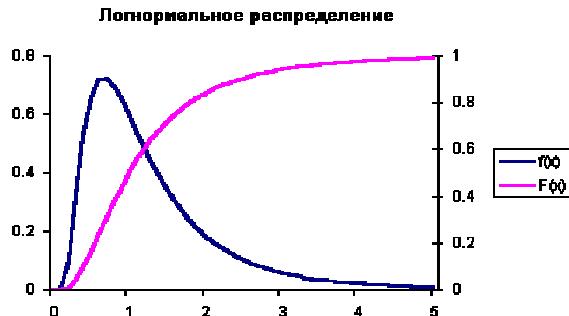
Объем пирамиды пропорционален произведению:  
сложность \* вероятность успеха

**Гипотеза:**

Объем пирамиды постоянен для данной команды и данного проекта

# Детали модели Мараско

- Наиболее продвинутая в математическом отношении модель компромиссов
- Для вероятности успеха принята **логнормальная функция распределения**
- Удобна для коррекции планов распределения ресурсов на каждом витке итеративного ЖЦ проекта, когда варьируются *относительные* изменения факторов
- Модель и методика включена в РМВОК в 2004



## Уязвимые места модели:

- Параметры в основании пирамиды:
  - не полностью независимы
  - не равнозначны: время (обратное скорости) наиболее важно
  - их количественное выражение - *метрики* - не formalизованы
- Допущение о логнормальном распределении вероятностей успеха не обосновано экспериментально
- Гипотеза об инвариантности объема пирамиды не подтверждена эмпирически

# Оценка трудоемкости проекта

- Необходима для планирования всех ресурсов до начала проекта
- Постоянно корректируется в ходе проекта - для отдельных задач
- Обычная мера трудоемкости: человеко-месяц (неделя, день, час)

## Проблемы:

- ❖ Большая неопределенность прогноза
  - ❑ Программисты не умеют точно оценивать требуемое им время
    - чаще занижают оценку (до 2 раз и больше)
    - реже чрезмерно завышают (перестраховываются)
- ❖ Человеко-месяц – неподходящая мера трудоемкости (однако другой нет!):



Производительность:  $O(N)$

Затраты на коммуникацию:  $O(N^2)$

Разделяемая (shared) =  
совместно выполняемая

# Время как самый критический ресурс

- Программные проекты чаще проваливаются от нехватки календарного времени, чем по всем другим причинам
- «Закон Брукса»: Если проект не укладывается в сроки, то добавление рабочей силы задержит его еще больше
  - уменьшение охвата (сокращение функциональности) в этом случае – гораздо лучший выход
- Одно из эмпирических наблюдений Марраско:

Задержка проекта в неделях может составить корень квадратный из числа оставшихся по плану недель

Пример: по плану осталось 16 недель, на самом деле потребуется от 16 до 20 недель

Как повысить точность оценок и снизить риски нарушения сроков?

- Вести подробный учет оценок и реально затраченного времени на задачи выполненных проектов для прогнозирования трудоемкости будущих проектов
- Делать внутренний календарный график (рабочий план) более жестким, чем внешний – в контракте с заказчиком
- Предусматривать тем больший буфер (запас времени), чем сложнее задача

# Планирование загрузки исполнителей и оценка общей трудоемкости

Занятость программистов на проекте АБВ

	bvs	paul	igork	nata	anton	vlad	vladik	mitja	matthew
november	0,5	0,5							
december	1	0,5	1	1	0,5	0,5	0,25		
january		1	1	1	1	1	0,25		
february		1	1	1	1	1		0,25	0,125
march		1	1	1	1	1		0,5	0,5
april		0,75	0,75	0,75	0,75	0,25		0,25	0,125

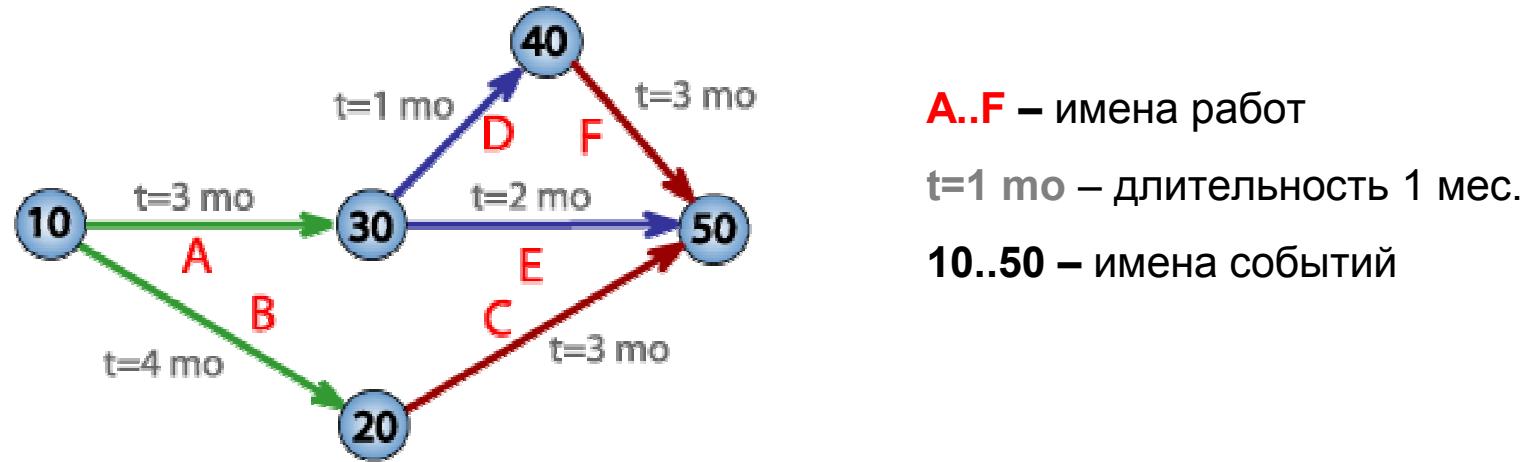
total	26
-------	----

man-months ← Опять чедовеко-месяцы!

Более детальное планирование – по дням и часам

# Временное планирование

Сетевой график, или PERT-диаграмма (1958) – традиционный способ планирования проектов:



- Дуги – работы; нагружены их длительностью
- Вершины – события начала и завершения работ
- Путь от начала к концу проекта с наибольшей длиной (= длительности всего проекта) наз. *критическим*
  - в каждый момент времени контролировать нужно состояние именно тех работ, которые лежат на нем.

# Календарное планирование

Таблица «календарное время – задачи» - другая форма сетевого графика:

Задача	Июнь				Июль				
	1..7	8..14	15..21	22..28	29..5	6..12	13..19	20..26	27..2
Задача № 1									
Задача № 2									
Задача № 3									
Задача № 4									

Таблица «время – исполнители»:

Исполнитель	Июнь				Июль				
	1..7	8..14	15..21	22..28	29..5	6..12	13..19	20..26	27..2
Иванов	Задача № 1					Задача № 3			
Соколов			Задача № 2						
Жилин	Задача № 4								
Костылин									



- Отношение следования задач

# Инструменты планирования

## Электронные таблицы (MS Excel или OOo Calc)

Двухходовые таблицы:

- время – ресурсы (напр., исполнители)
  - время – задачи (работы)
  - задачи – ресурсы
- } На предыдущих слайдах

Excel – инструмент планирования, удобный только для *небольших* проектов:

- перепланирование – трудоемкая ручная процедура (синхронный сдвиг многих сроков)

Программы управления проектами: MS Project, OpenProj, Jira, ...

- Лучшая наглядность
- Разнообразные диаграммы взаимосвязей задач, ресурсов, времени и затрат
- Автоматизация рутинных операций
- Возможен учет сотен задач и исполнителей

# Основные понятия MS Project

**Задача, или работа (Task)** – наименьшая единица работы ( $\leq$  несколько дней)

- например: Кодирование программы инсталляции приложения

**Фаза, или составная задача/работа ((Summary task)** – совокупность задач, завершающихся определенным результатом

- Завершающая задача фазы (возможно, пустая) называется вехой (Milestone)

**Ресурсы (Resources)**

- Люди (роли или имена)
- Оборудование

**Затраты (Cost)**

- Повременная ставка (Rate) – для людей, напр., 100 руб/час

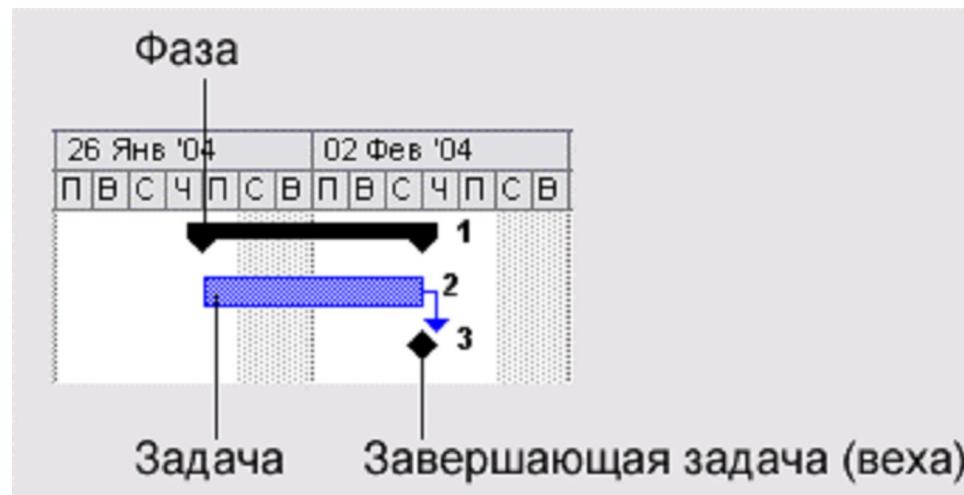
**Назначения (Assignments):** связь задач и ресурсов для их выполнения

**Задачи планирования:**

- составление списка задач и фаз с временными оценками
- назначение ресурсов, в т.ч. исполнителей и ответственных за исполнение задач
- расчет времени и стоимости – по исполнителям и суммарного
- регулярное перепланирование (изменение плана) из-за изменения требований, уточнения временных оценок, нарушения сроков выполнения и т.п.

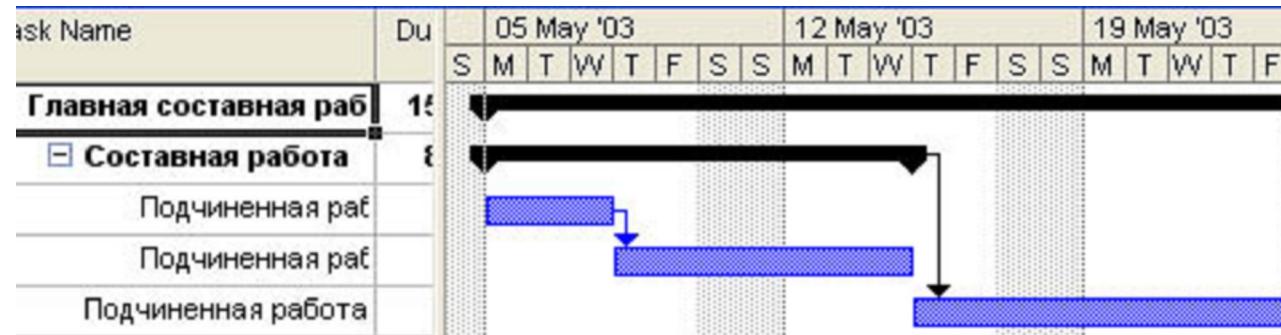
# Элементы диаграммы Ганта

В MS Project календарный план – это диаграмма Ганта (Gantt Chart) - столбчатая диаграмма «время – задачи» или временной график, расписание



Здесь фаза состоит из единственной задачи 2;  
задача 3 – пустая (веха) следует после задачи 2

# Примеры: фрагменты диаграмм Ганта

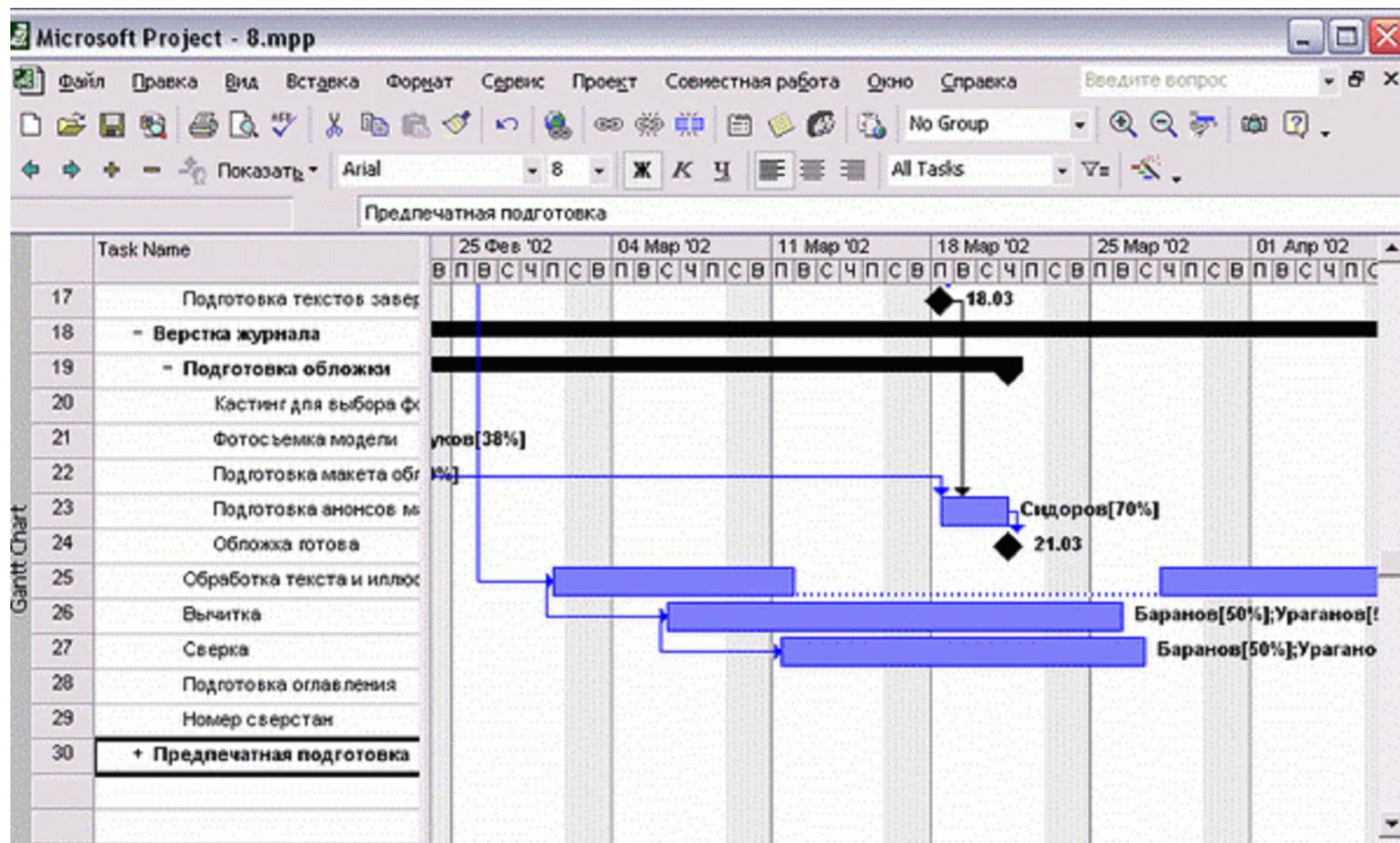


Одна составная задача (фаза) вложена в другую (главную)



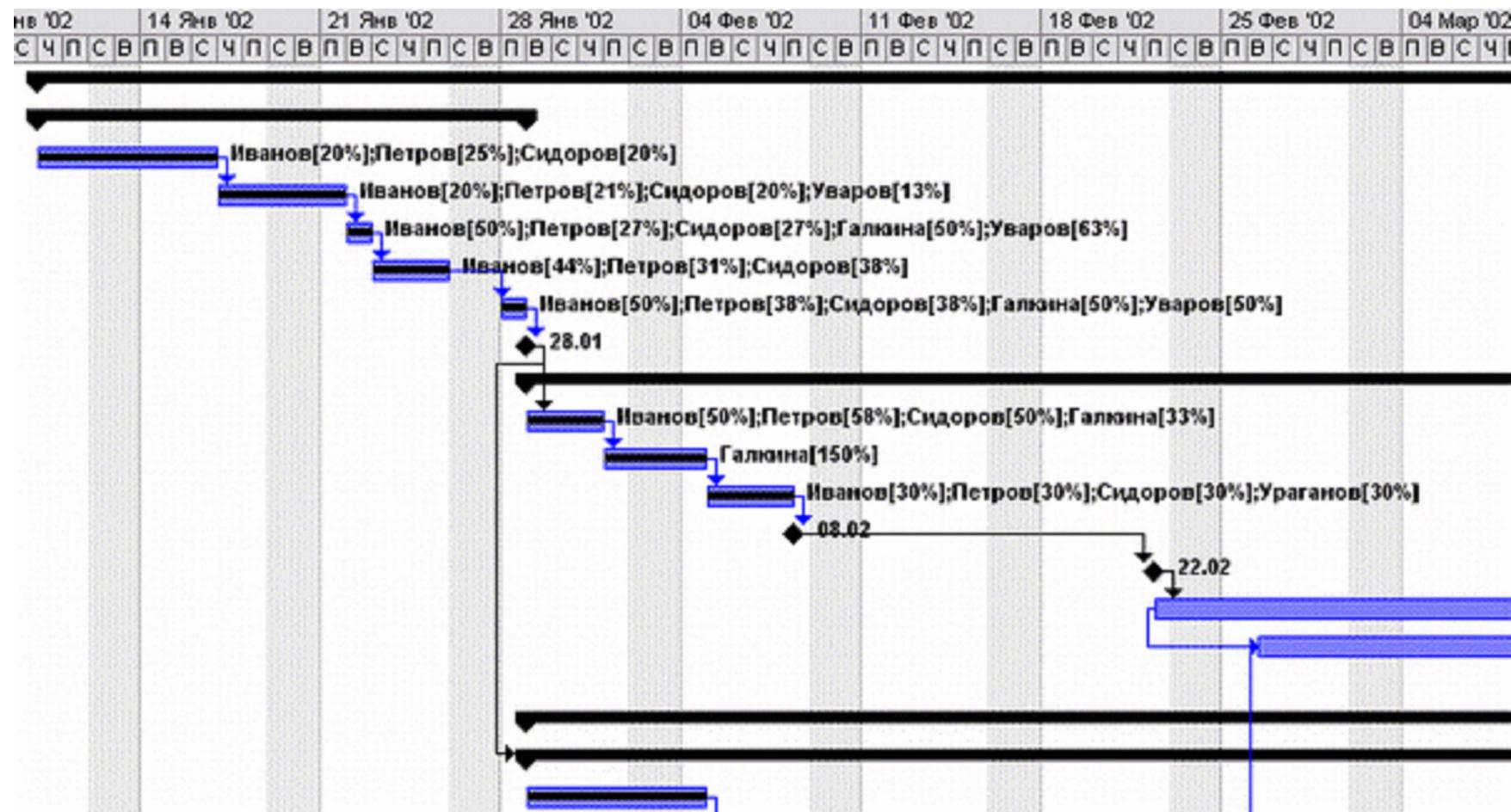
Две взаимосвязанных составных задачи

## Примеры ... (2)



Задачи нагружаются атрибутами – ресурсами (здесь - исполнителями)  
и комментариями

## Фрагмент диаграммы для цепочки задач



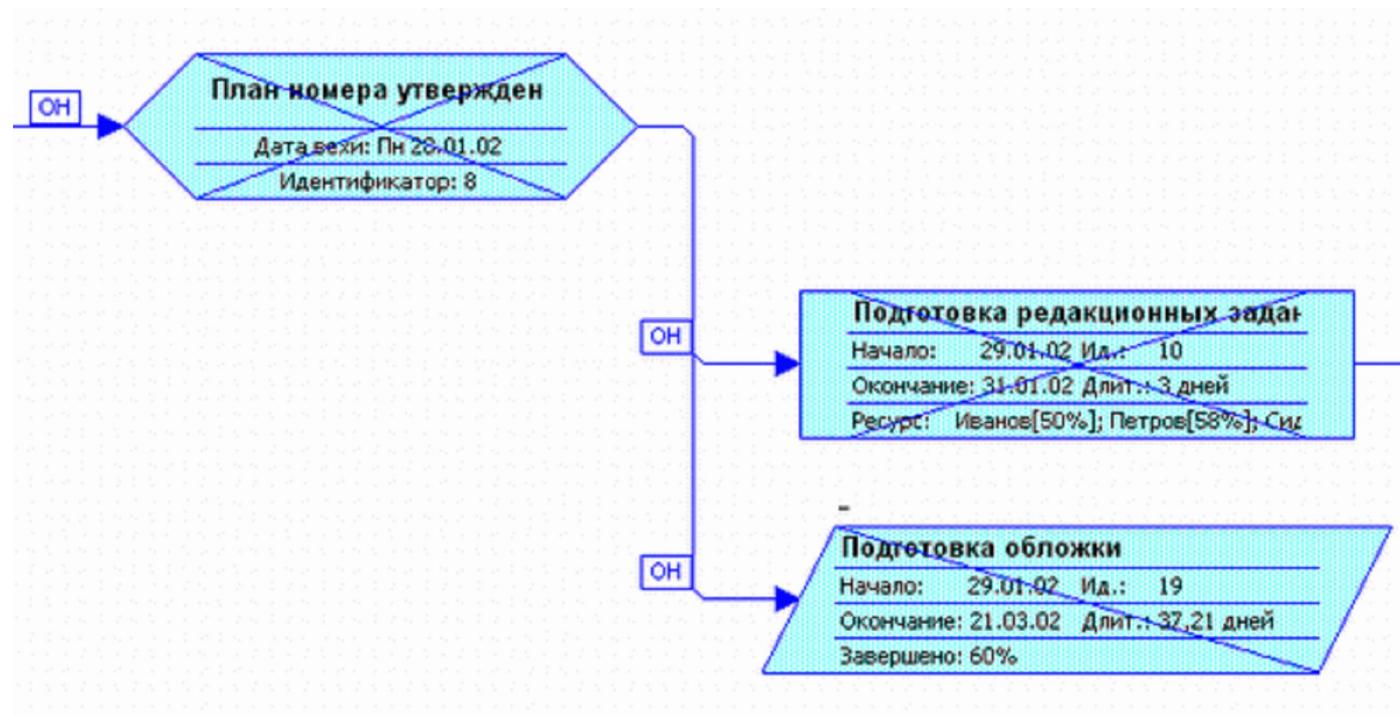
**Перепланирование:** изменение длительности любой задачи или добавление новой задачи влечет автоматический сдвиг всех более поздних и зависимых задач

# Сетевой график

- вспомогательная диаграмма № 1

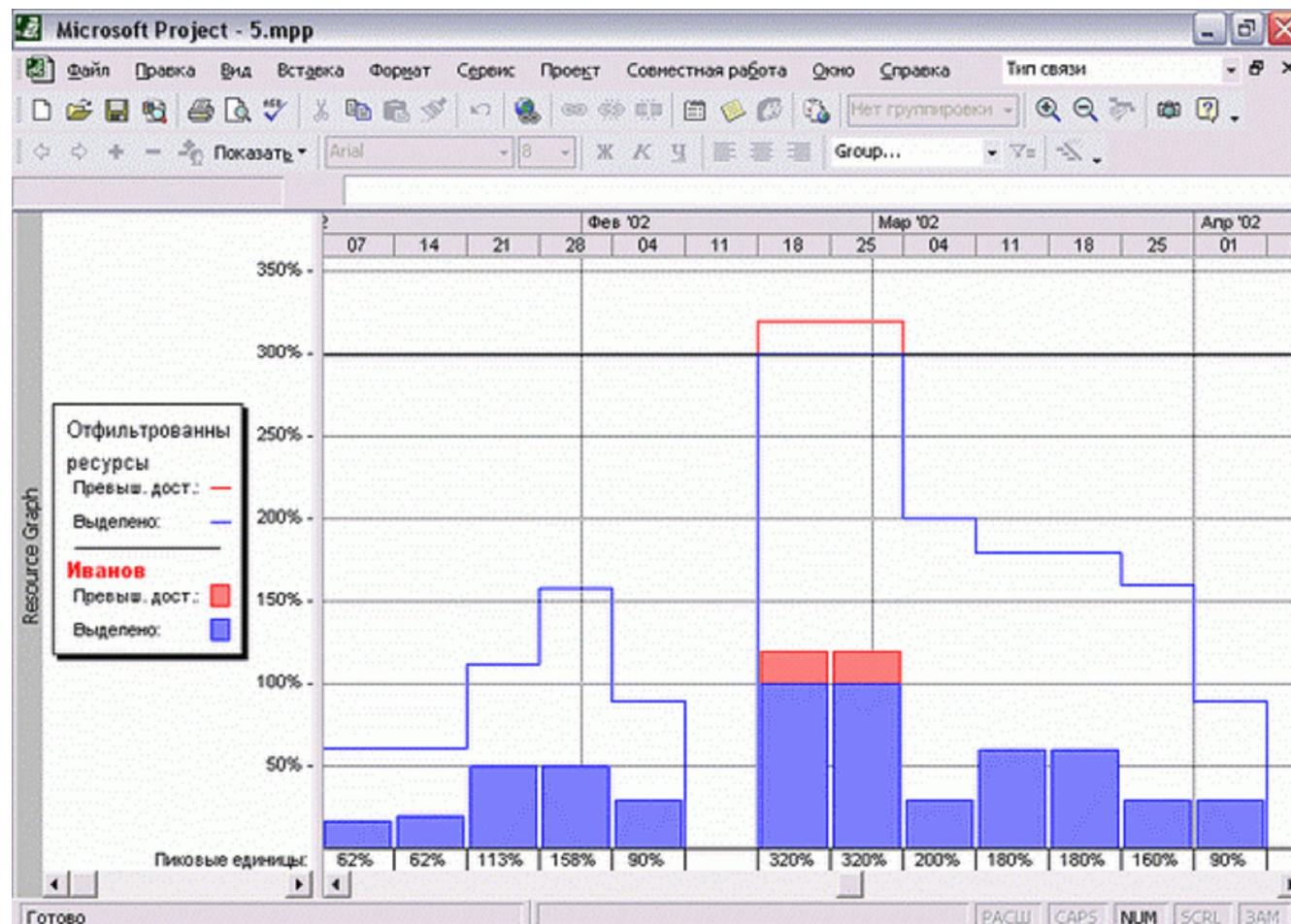
Задачи представлены в виде блоков, соединенных стрелками в блок-схему в соответствии с взаимосвязями задач в плане проекта

Так удобно планировать проекты с большим числом связей между задачами



# Диаграмма ресурсов

- вспомогательная диаграмма №2



# Достоинства MS Project

по сравнению с электронными таблицами

- Диаграмма Ганта – традиционная форма планирования
- Диалоговый ввод и редактирование диаграмм взаимосвязей задач, ресурсов, времени и затрат
- Многоаспектные, а не только двухходовые диаграммы
  - ✓ благодаря возможности нагрузки дуг комментариями
- Переменный масштаб времени в диаграммах
  - ✓ месяцы, недели, дни, часы
- Помощь в отслеживании реальных сроков выполнения задач: сигнализация о вехах/дедлайнах
- Отслеживание критического пути
- Автоматическое перепланирование всего календарного плана при изменении любого срока
- Вычисление различных сводных характеристик проекта
  - ✓ напр., суммарная загрузка сотрудника за определенный период

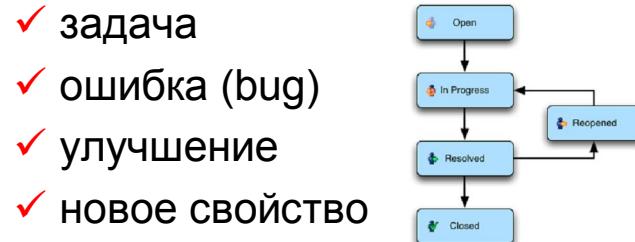
# Развитие MS Project

- переход от индивидуального инструмента менеджера  
к коллективному инструменту команды

- MS Project Professional
  - ✓ корпоративная, клиент-серверная версия
  - ✓ несколько проектов с общими ресурсами
  - ✓ совместное управление несколькими менеджерами
- MS Enterprise Project Management Solution
  - ✓ поддержка ЖЦ MSF для разработки ERP-систем
- MS Visual Studio Team System (VSTS)
  - ✓ команда исполнителей – тоже клиенты
  - ✓ интеграция с Visual Studio, Source Safe и SQL Access

# JIRA: пример современного инструмента планирования

- Веб-интерфейс
- Учет планируемого и реального времени на работы
- Agile-стиль: карточки работ →
- Уведомление о событиях по эл. почте
- Отслеживание состояния работ 4-х видов:



- Наглядное отображение состояния и сводных характеристик проекта →
  - ✓ «Приборный пульт»: Visual Analytics
- ✓ Универсальность
  - ✓ не только программные проекты



# Управление рисками

**Риск проекта** – всякое событие или условие, которое может оказать негативное влияние на ход выполнения проекта

Управление рисками - это процесс их предсказания, анализа и эффективной превентивной работы над ними, под лозунгами:

- Предугадывайте проблемы вместо того, чтобы реагировать на них после их возникновения
- Работайте над первопричиной, а не над проявляющимися симптомами
- Готовьте планы решения проблем заранее, до того как проблемы возникнут
- Используйте понятный, структурированный и воспроизводимый процесс разрешения проблем

**Risk management; to manage** – **справляться, обходиться, а не только управлять**

# MSF: 6 этапов процесса управления рисками

- *Выявление рисков (risk identification)* – члены команды выносят на обсуждение всей команды факты наличия рисков
- *Анализ (risk analysis) и ранжирование рисков (risk prioritization)*. Даются оценки вероятности риска, его *степени ущерба*
- *Планирование рисков (risk planning)* - выработка стратегий, планов и конкретных шагов по уменьшению отрицательного влияния рисков (*risk mitigation* - смягчение)
- *Отслеживание рисков (risk tracking)* - наблюдение за конкретными рисками и прогрессом в осуществлении составленных планов.  
*Отчетность о рисках (risk reporting)* - информирование о состоянии рисков проекта и планов по управлению ими
- *Корректировка ситуации (risk control)* - процесс исполнения принятых в отношении рисков планов и контроля за ходом их исполнения, включая инициирование изменений всего проекта
- *Извлечение уроков (risk learning)* - формализованный процесс усвоения накопленного за время работы над проектом опыта в форме, доступной для использования как внутри проектной группы, так и на уровне всего предприятия

# Меры по управлению рисками

## Анализ: классификация и примеры рисков

1. Технологические
  - Запланированное инструментальное средство не работает
2. Связанные с персоналом
  - Ведущий разработчик заболел в самое неподходящее время
3. Организационные
  - Финансовые затруднения в организации привели к уменьшению бюджета проекта
4. Связанные с требованиями заказчика
  - Изменения требований приводят к значительным повторным работам по проектированию
5. Риски оценивания трудоемкости и характеристик проекта
  - Не удается достичь требуемой производительности программы
  - Трудоемкость задачи оказалась заниженной в 5 раз

## Возможные стратегии планирования для них

2. Реорганизовать команду проекта так, чтобы обязанности ее членов перекрывались
- 3.а Подготовить краткий документ для руководства, показывающий важность данного проекта для достижения финансовых целей организации
- 3.б Урезать функциональность ПП (см. треугольник проекта)
4. Определить требования, наиболее вероятно подверженные изменениям

# Заключение

- Модель проектной команды – составная часть технологии программирования, определяющая расстановку кадров, их обязанности и ответственность
- Управление проектом – комплексная деятельность, требующая знаний в технической, экономической и психологической областях
- MS Project – де-факто стандарт программы автоматизации управления проектами менеджером
- Тенденции автоматизации управления проектами:
  - коллективное планирование и учет в локальной/ глобальной сети
  - средства визуального анализа данных (Visual Analytics)
- Анализ рисков и подготовка мер противодействия им – обязательная часть планирования проекта

# Дополнительные вопросы

- В модели команды главного программиста (слайд 8):
  - предложите другие возможные роли
  - в чем проявляется недостаточная гибкость модели?
  - почему большой проект трудно разрезать пополам и запустить две бригады главного программиста параллельно ?
- В модели команды MSF:
  - в чем причины несовместимости некоторых ролей в таблице на слайде 13
  - каково минимальное число исполнителей в команде, удовлетворяющее условиям совместимости ролей ?
- Почему треугольник компромиссов проекта и другие подобные модели называются метафорами?
- Почему в 5-параметрической пирамиде проекта Марраско для вероятности успеха принята логнормальная, а не нормальная функция распределения?
- Что следует предпринимать при срыве или угрозе срыва срока сдачи этапа проекта?
- Приведите другие примеры рисков пяти типов (слайд № 45).