

# Оценка трудоемкости и сроков разработки ПО

С. Архипенков

# Об авторе

- Сергей Архипенков, PMP PMI.
- Стаж в разработке ПО более 30 лет.
- Автор книг, статей, учебных курсов.
- [www.arkhipenkov.ru](http://www.arkhipenkov.ru)
- [sergey@arkhipenkov.ru](mailto:sergey@arkhipenkov.ru)

# Концепция

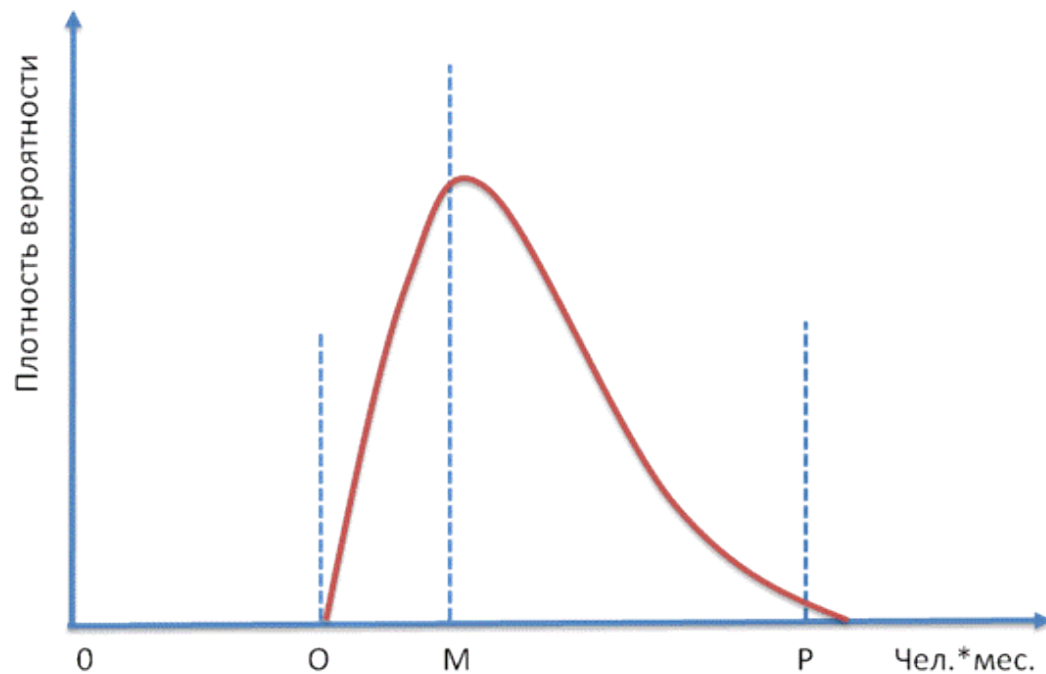
- Название проекта
- Цели проекта
- Результаты проекта
- Допущения и ограничения
- Ключевые участники и заинтересованные стороны
- Ресурсы проекта
- Сроки
- Риски
- Критерии приемки
- Обоснование полезности проекта

Подтверждение и согласование единого понимания целей, задач и результатов проекта.

# Содержание и состав работ

Иерархическая структура работ - *ориентированная на результат* иерархическая декомпозиция работ, выполняемых командой проекта для достижения целей проекта и необходимых результатов. С ее помощью структурируется и определяется все содержание проекта. Каждый следующий уровень иерархии отражает более детальное определение элементов проекта.

# Оценка – вероятностное утверждение



*Хорошо?*

$$PM = A \times SIZE^E \times \prod_{i=1}^n EM_i$$

$$A = 2,94$$

$$E = B + 0,01 \times \sum_{j=1}^5 SF_j$$

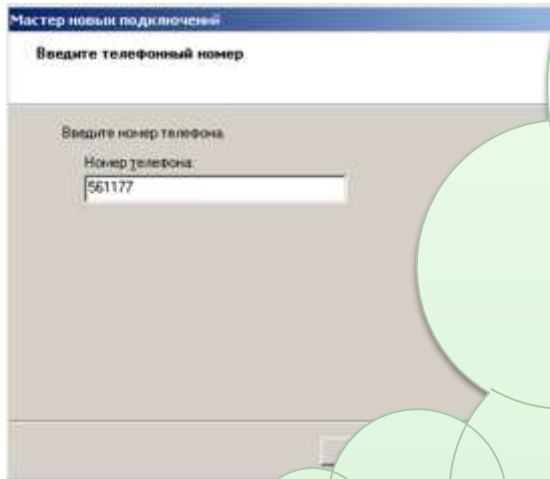
$$B = 0,91$$

*Плохо?*

*Трудоемкость = КоличествоФакторов x СредниеЗатратыНаФактор*

# Откуда берется неопределенность?

Пример. Форма ввода телефонного номера



Может ли вводиться несколько номеров?

Должна ли быть проверка номеров на действительность?

Простая или сложная проверка?

Если реализуем простую проверку, то не захочет ли клиент заменить ее на более сложную?

Должна ли проверка работать для иностранных номеров?

Можно ли воспользоваться готовым решением?

Каково должно быть качество реализации?

Вероятность ошибки после поставки?

Сколько времени потребуется на реализацию и отладку? (зависит от конкретного исполнителя).

# Способы оценки

- Использование собственного опыта аналогичных проектов.
- Экспертная оценка.
- Использование методик на основе отраслевого опыта.
  - FPA IFPUG - метод функциональных точек (14 параметров 150 стр.),
  - метод COCOMO II, Constructive Cost Model (21 параметр, описание модели – 90 стр.).

## Важно

- Оценки должны согласовываться.
- Пример из IBM.
- Если с не удастся достичь соглашения по оценкам, то в проекте должен быть выделен начальный этап.

# **ПРАГМАТИЧНЫЙ ПОДХОД. МЕТОД PERT**



# История

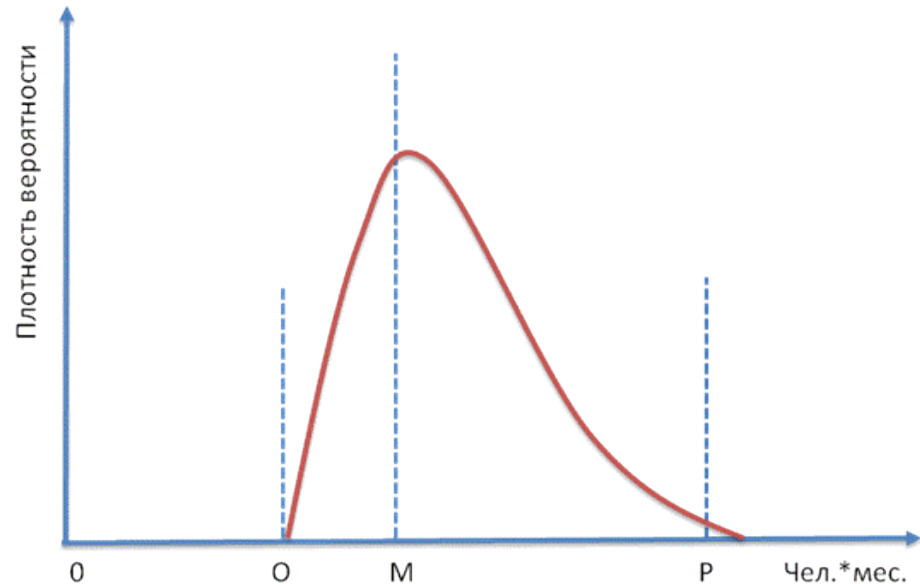
Инженерный метод оценки трудоемкости проекта PERT (Program / Project Evaluation and Review Technique) был разработан в 1958 году в ходе проекта по созданию баллистических ракет морского базирования «Поларис». Входом для данного метода оценки служит список элементарных пакетов работ.



# Достаточно три оценки

- $M_i$  – наиболее вероятная оценка трудозатрат.
- $O_i$  – минимально возможные трудозатраты на реализацию пакета работ. Ни один риск не реализовался. Быстрее точно не сделаем. Вероятность такого, что мы уложимся в эти затраты, равна 0.
- $P_i$  – пессимистическая оценка трудозатрат. Все риски реализовались.

*Важно. Оценки должны быть независимыми.*



# Оценка суммарной трудоемкости

- Средняя трудоемкость по каждому пакету

$$Ei = (Pi + 4Mi + Oi)/6$$

- Среднеквадратичная ошибка оценки каждого пакета

$$CKOi = (Pi - Oi)/6$$

- Оценка средней суммарной трудоемкости


$$E = \sum Ei$$

- Среднеквадратичная ошибка суммарной оценки


$$CKO = \sqrt{\sum CKO_i^2}$$

- Гарантированная оценка с вероятностью 95%

$$E_{95\%} = E + 2 * CKO$$



Ошибка  
300%



Ошибка  
30%  
100 работ

# Качество оценки

Приведите верхнюю и нижнюю границу, которые обеспечивают 90% вероятность покрытия истинного значения.

- Сколько городов в СНГ имеют метро?
- Температура плавления этилового спирта (°C)?
- Длина реки Волга (км)?
- Максимальная продолжительность жизни Галапагосской черепахи?
- Сборы к/ф «Ирония судьбы-2» за 2008 год (\$)?

Хорший ли вы оценщик? Качество оценки:

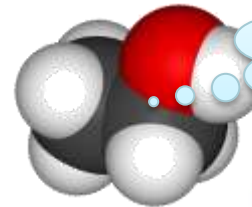
$$D = 100 * 2 * (O_v - O_n) / (O_v + O_n)$$

Если, конечно накрыли истинное значение

15



-114,5 °C



3690 км



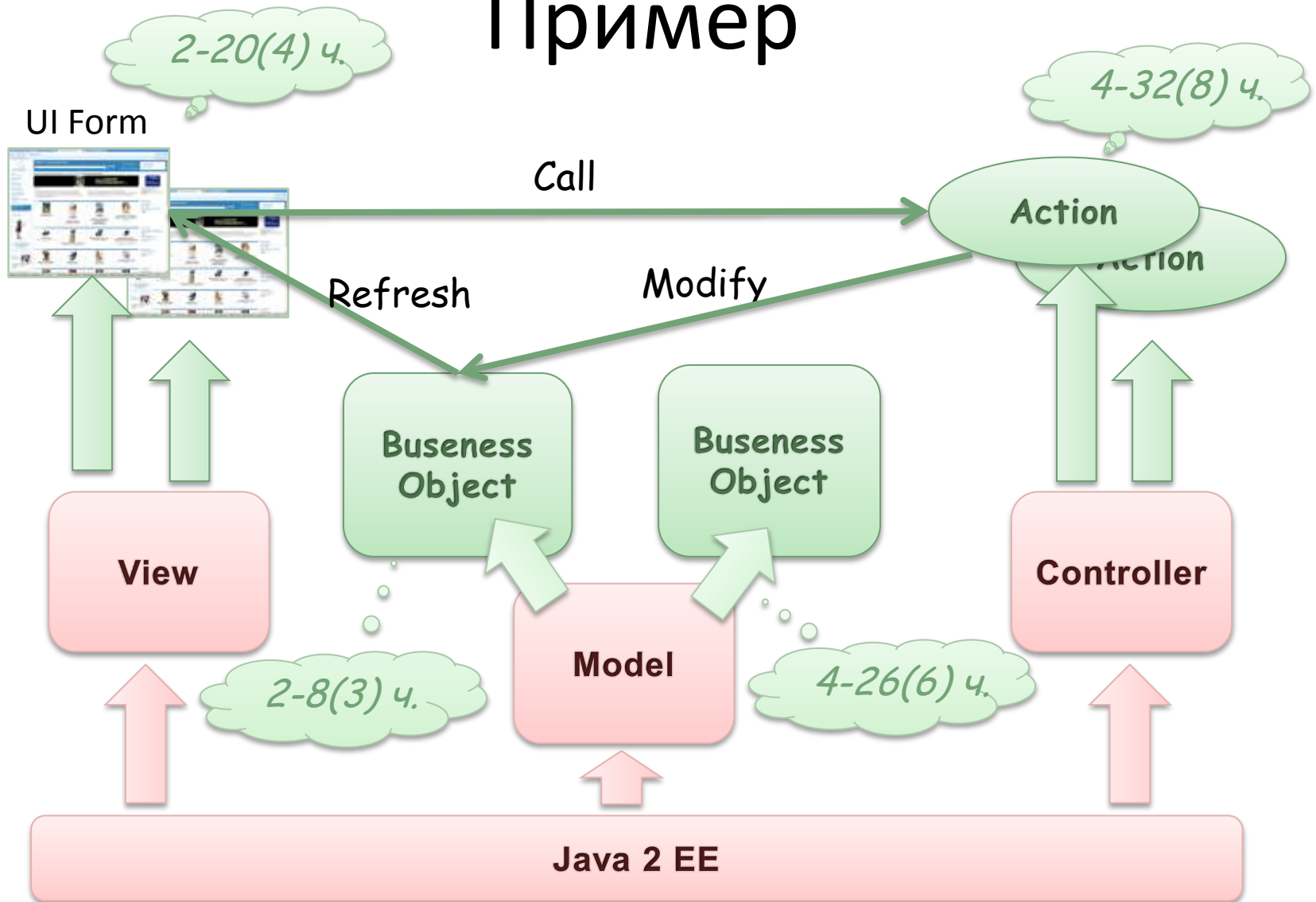
177 лет



\$49,92  
млн.




# Пример



# Оценки

•Экранов	20
•Обработчиков	60
•Новых объектов	16
•Новых методов	40



*Не забыть  
умножить  
на 4!*

•  $E_{95\%} \approx 1300 \text{ чел.} \cdot \text{час.}$   
(относительная  
ошибка 4%)

• Управление	– 10%
• Конфигурирование	– 10%
• Управление требованиями	– 10%
• Проектирование	– 15%
• <b>Разработка</b>	<b>– 25%</b>
• Тестирование	– 25%
• Документирование	– 5%
<i>Итого:</i>	<i>100%</i>

# Увеличение трудозатрат в зависимости от масштаба проекта

•10	KSLOC	1,0
•100	KSLOC	1,3 - 1,7
•1000	KSLOC	1,6 - 3,0



# Зависимость успеха проекта от масштаба

Размер, SLOC	Досрочно, %	В срок, %	Опоздание, %	Провал, %
1 000	11	81	6	2
10 000	6	75	12	7
100 000	1	61	18	20
1 000 000	< 1	28	24	48
10 000 000	0	14	21	65

Стоит хорошо  
подумать  
перед стартом!



# Влияние сложности продукта

Производительность в  
SLOC/мес на проекте в 100  
KSLOC

- 300-7000 (800) интранет.
- 200-7000 (600) бизнес.
- 100-2000 (300) Интернет.
- 50-600 (100) системное ПО, телеком.
- 20-300 (50) системы реального времени.



# **ОБЗОР МЕТОДА ФУНКЦИОНАЛЬНЫХ ТОЧЕК**

# Назначение

Оценка трудоемкости на основе логической модели объема программного продукта, измеренного количеством функционала, востребованного заказчиком и поставляемого разработчиком.

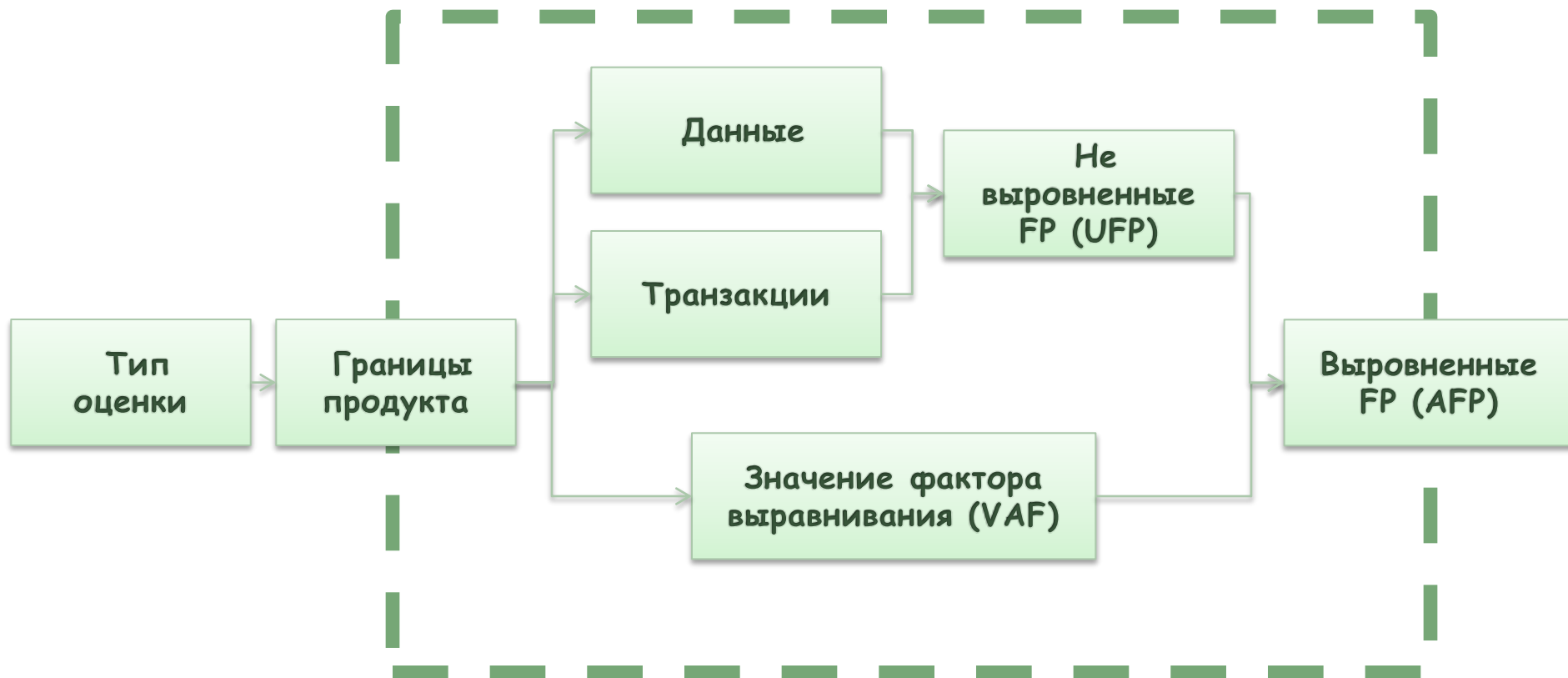
## Достоинства

- Измерения не зависят от технологической платформы.
- Метод обеспечивает единообразный подход к оценке всех проектов в компании.

# История

Метод функциональных точек разработан Аланом Альбрехтом (Alan Albrecht) в середине 70-х. Метод был впервые опубликован в 1979 году. В 1986 году была сформирована Международная Ассоциация Пользователей Функциональных Точек (International Function Point User Group - IFPUG), которая опубликовала несколько ревизий метода.

# Процедура оценки



# Три типа оценок

1. Проект  
разработки



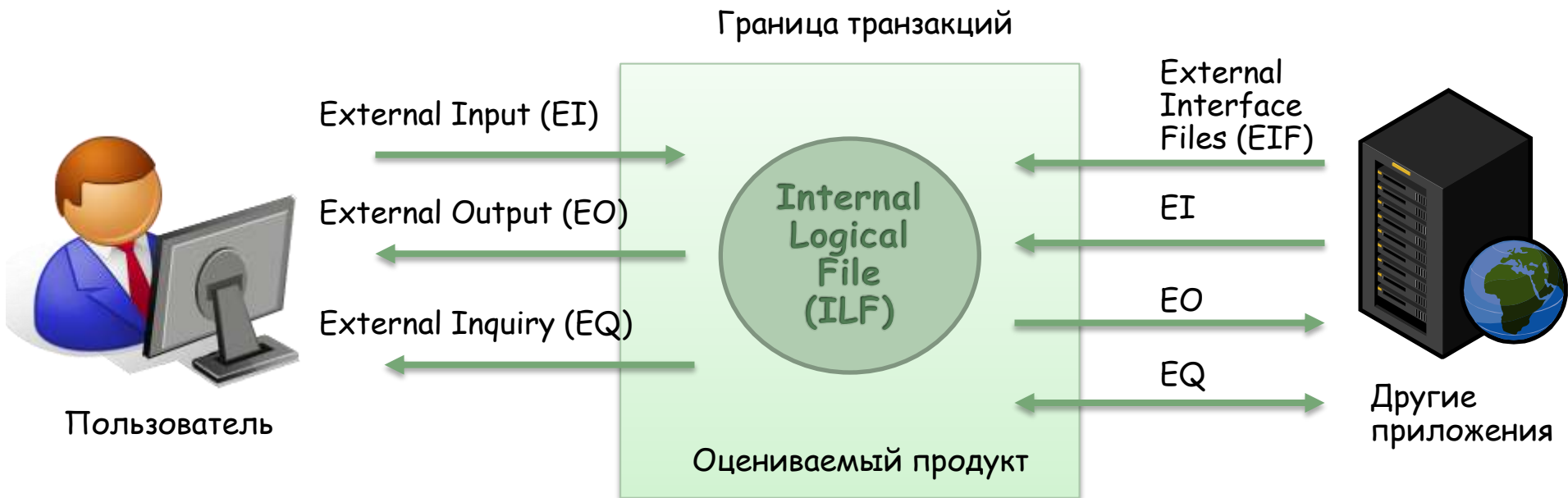
3. ГОТОВЫЙ  
продукт



2. Проект  
развития



# Границы продукта



# Оценка сложности данных

Характеристики сложности данных :

- DET (data element type) – неповторяемое уникальное поле данных.  
Например, Имя Клиента – 1 DET; Адрес Клиента (индекс, страна, область, район, город, улица, дом, корпус, квартира) – 9 DET's
- RET (record element type) – логическая группа данных, например, адрес, паспорт, телефонный номер.

Сложность данных:

	1-19 DET	20-50 DET	50+ DET
1 RET	Low	Low	Average
2-5 RET	Low	Average	High
6+ RET	Average	High	High

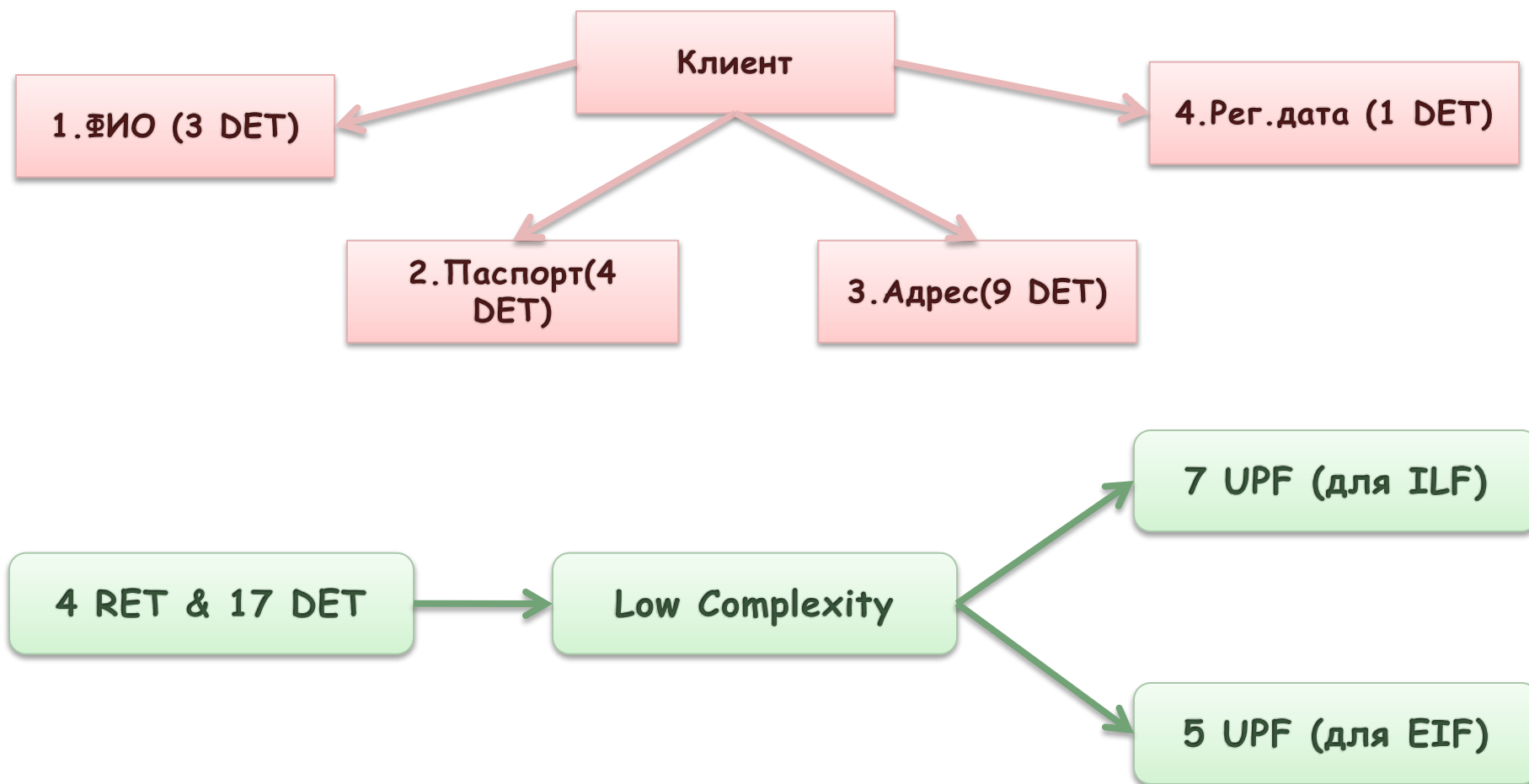
# Оценка данных в не выровненных функциональных точках (UFP)

Сложность данных ILF	Количество UFP
Low	7
Average	10
High	15

Сложность данных EIF	Количество UFP
Low	5
Average	7
High	10



# Пример. Оценка объекта данных «Клиент»



# Отличия между типами транзакций

## Легенда

О – основная;

Д – дополнительная;

NA – не применима.

Транзакция – это элементарный неделимый замкнутый процесс, представляющий значение для пользователя и переводящий продукт из одного consistente состояния в другое.

Функция	Тип транзакции		
	EI	EO	EQ
Изменяет поведение системы	О	Д	NA
Поддержка одного или более ILF	О	Д	NA
Представление информации пользователю	Д	О	О

# Характеристики сложности транзакции

*FTR* (file type referenced) – позволяет подсчитать количество различных файлов (информационных объектов) типа ILF и/или EIF модифицируемых или считываемых в транзакции.

*DET* (data element type) – неповторяемое уникальное поле данных. Примеры. *EI*: поле ввода, кнопка. *EO*: поле данных отчета, сообщение об ошибке. *EQ*: поле ввода для поиска, поле вывода результата поиска.

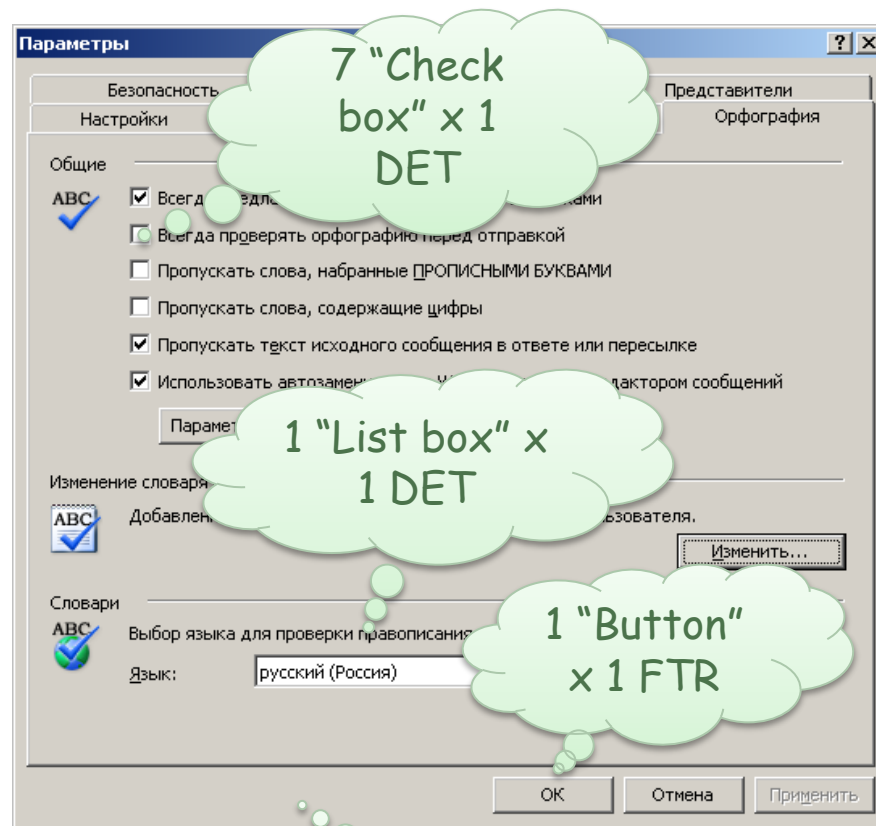
EI	1-4 DET	5-15 DET	16+ DET
0-1 FTR	Low	Low	Average
2 FTR	Low	Average	High
3+ FTR	Average	High	High

EO & EQ	1-5 DET	6-19 DET	20+ DET
0-1 FTR	Low	Low	Average
2-3 FTR	Low	Average	High
4+ FTR	Average	High	High

# Оценка транзакций в точках UFP

Сложность транзакций EI & EQ	Количество UFP
Low	3
Average	4
High	6

Сложность транзакций EO	Количество UFP
Low	4
Average	5
High	7



# Определение суммарного количества UFP

Объем продукта в не выровненных функциональных точках (UFP) определяется путем суммирования по всем информационным объектам (ILF, EIF) и элементарным операциям (транзакциям EI, EO, EQ).

$$UFP = \sum_{ILF} UFP_i + \sum_{EIF} UFP_i + \sum_{EI} UFP_i + \sum_{EO} UFP_i + \sum_{EQ} UFP_i$$

# Определение значения фактора выравнивания

- Значение фактора выравнивания,  $VAF$  зависит от 14 параметров, которые определяют системные характеристики продукта.
- Суммарное влияние процедуры выравнивания лежит в пределах +/- 35% относительно объема рассчитанного в  $UFP$ .
- Последовательность операций расчета:
  - Оценка 14 системных характеристик по шкале от 0 до 5 (degree of influence,  $DI$ ).
  - Расчет суммарного эффекта 14 системных характеристик (total degree of influence,  $TDI$ ).

$$TDI = \sum DI$$

- Расчет значения фактора выравнивания:

$$VAF = (TDI * 0.01) + 0.65$$

# 14 системных характеристик

## *Обмен данными*

- 0 – продукт представляет собой автономное приложение;
- 5 – продукт обменивается данными по более, чем одному телекоммуникационному протоколу

## *Распределенная обработка данных*

- 0 – продукт не перемещает данные;
- 5 – распределенная обработка данных выполняется несколькими компонентами системы

## *Производительность*

- 0 – пользовательские требования по производительности не установлены;
- 5 – время отклика сильно ограничено критично для всех бизнес-операций, для удовлетворения требованиям необходимы специальные проектные решения и инструменты анализа.

## *Ограничения по аппаратным ресурсам*

- 0 – нет ограничений;
- 5 – продукт целиком должен функционировать на определенном процессоре и не может быть распределен

## *Транзакционная нагрузка*

- 0 – транзакций не много, без пиков;
- 5 – число транзакций велико и неравномерно, требуются специальные решения и инструменты

## *Интенсивность взаимодействия с пользователем*

- 0 – все транзакции обрабатываются в пакетном режиме;
- 5 – более 30% транзакций – интерактивные

## *Эргономика*

- 0 – нет специальных требований;
- 5 – требования по эффективности очень жесткие

# 14 системных характеристик

## *Интенсивность изменения данных (ILF) пользователями*

- 0 – не требуются;
- 5 – изменения интенсивные, жесткие требования по восстановлению

## *Сложность обработки*

- 0 – обработка минимальна;
- 5 – требования безопасности, логическая и математическая сложность, многопоточность.

## *Повторное использование*

- 0 – не требуется;
- 5 – продукт разрабатывается как стандартный многоразовый компонент

## *Удобство инсталляции*

- 0 – нет требований;
- 5 – установка и обновление ПО производится автоматически

## *Удобство администрирования*

- 0 – не требуется;
- 5 – система автоматически самовосстанавливается

## *Портируемость*

- 0 – продукт имеет только 1 инсталляцию на единственном процессоре;
- 5 – система является распределенной и предполагает установку на различные «железо» и ОС

## *Гибкость*

- 0 – не требуется;
- 5 – гибкая система запросов и построение произвольных отчетов, модель данных изменяется пользователем в интерактивном режиме



# Расчет количества выровненных функциональных точек

Проект разработки (development functional point)

$$DFP = (UFP + CFP) * VAF$$

Проект развития (enhancement functional point)

$$EFP = (ADD + CHGA + CFP) * VAFA + (DEL * VAFB)$$

Здесь:

**CFP** – дополнительные функциональные точки, которые потребуются при установке, например, миграция данных;

**ADD** - функциональные точки для добавленной функциональности;

**CHGA** - функциональные точки для измененных функций, рассчитанные после модификации;

**VAFA** – величина фактора выравнивания рассчитанного после завершения проекта;

**DEL** – объем удаленной функциональности;

**VAFB** - величина фактора выравнивания рассчитанного до начала проекта.

# **ОСНОВЫ МЕТОДИКИ СОСОМО II**

# История

SOCOMO впервые опубликована Бари Боэмом в 1981, как результат анализа 63 проектов компании «TRW Aerospace». В 1997 была усовершенствована и получила название SOCOMO II. Калибровка параметров производилась по 161 проекту разработки

Два подхода к  
оценки проекта

Предварительная  
оценка на  
начальной фазе.

Детальная оценка  
после проработки  
архитектуры

# Формулы оценки трудоемкости проекта

$$PM = A \times SIZE^E \times \prod_{i=1}^n EM_i$$

$$A = 2,94$$

$$E = B + 0,01 \times \sum_{j=1}^5 SF_j$$

$$B = 0,91$$

где

$SIZE$  - размер продукта в KSLOC

$EM_i$  - множители трудоемкости

$SF_j$  - факторы масштаба

$n = 7$  - для предварительной оценки

$n = 17$  - для детальной оценки

# Оценки количества SLOC на 1 UFP

Язык программирования	Среднее	Оптимистичная	Пессимистичная
Assembler	172	86	320
C	148	9	704
C++	60	29	178
C#	59	51	66
J2EE	61	50	100
JavaScript	56	44	65
PL/SQL	46	14	110
Visual Basic	50	14	276

«Function Point Programming Languages Table», Quantitative Software Management, Inc. , 2005.

# Список факторов масштаба, $SF_{j=1...5}$

## 1. PREC – прецедентность (опыт аналогичных разработок).

- Very Low – опыт в продукте и платформе отсутствует.
- Extra High – продукт и платформа полностью знакомы.

## 2. FLEX – гибкость процесса разработки.

- Very Low – процесс строго детерминирован.
- Extra High – определены только общие цели.

## 3. RESL – архитектура и разрешение рисков.

- Very Low – риски неизвестны/не проанализированы.
- Extra High – риски разрешены на 100%.

## 4. TEAM – сработанность команды.

- Very Low – формальные взаимодействия.
- Extra High – полное доверие, взаимозаменяемость и взаимопомощь.

## 5. PMAT – зрелость процессов.

- Very Low – CMM Level 1.
- Extra High – CMM Level 5.

# Значения факторов масштаба, $SF_{j=1...5}$

Scale Factors	Very Low	Low	Nominal	High	Very High	Extra High
1.PREC	6.20	4.96	3.72	2.48	1.24	0.00
2.FLEX	5.07	4.05	3.04	2.03	1.01	0.00
3.RESL	7.07	5.65	4.24	2.83	1.41	0.00
4.TEAM	5.48	4.38	3.29	2.19	1.10	0.00
5.PMAT	7.80	6.24	4.68	3.12	1.56	0.00

# Влияние факторов масштаба

Оцениваем проект в 100 KSLOC

Все факторы масштаба  
оценены как «Extra High»

$$E = 0,91 + 0,01 * 0,00; PM = 2,94 * (100)^{0,91} = \mathbf{194} \text{ (чел.*мес.)}$$

Все факторы масштаба  
оценены как «Very Low»

$$E = 0,91 + 0,01 * 31,6; PM = 2,94 * (100)^{1,226} = \mathbf{832} \text{ (чел.*мес.)}$$

Факторы масштаба могут  
изменять оценку  
трудоемкости в 4,3 раза!



# Множители трудоемкости для предварительной оценки, $EM_{i=1...3}$

## 1. PERS - квалификация персонала

- Extra Low - аналитики и программисты имеют низшую квалификацию, текучесть больше 45%
- Extra High - аналитики и программисты имеют высшую квалификацию, текучесть меньше 4%

## 2. RCPX - сложность и надежность продукта

- Extra Low - продукт простой, специальных требований по надежности нет, БД маленькая, документация не требуется.
- Extra High - продукт очень сложный, требования по надежности жесткие, БД сверхбольшая, документация требуется в полном объеме.

## 3. RUSE - разработка для повторного использования

- Low - не требуется
- Extra High - требуется переиспользование в других продуктах.

# Множители трудоемкости для предварительной оценки, $EM_{i=4...7}$

## 4. PDIF - сложность платформы разработки

- Extra Low - специальные ограничения по памяти и быстродействию отсутствуют, платформа стабильна.
- Extra High - жесткие ограничения по памяти и быстродействию, платформа нестабильна.

## 5. PREX - опыт персонала

- Extra Low - новое приложение, инструменты и платформа.
- Extra High - приложение, инструменты и платформа хорошо известны.

## 6. FCIL - оборудование.

- Extra Low - инструменты простейшие, коммуникации затруднены.
- Extra High - интегрированные средства поддержки жизненного цикла, интерактивные мультимедиа коммуникации.

## 7. SCED - сжатие расписания.

- Very Low - 75% от номинальной длительности.
- Very High - 160% от номинальной длительности.

# Значения множителей трудоемкости

$EM_i$	Extra Low	Very Low	Low	Nominal	High	Very High	Extra High
1. PERS	2.12	1.62	1.26	1.00	0.83	0.63	0.5
2. RCPX	0.49	0.60	0.83	1.00	1.33	1.91	2.72
3. RUSE	n/a	n/a	0.95	1.00	1.07	1.15	1.24
4. PDIF	n/a	n/a	0.87	1.00	1.29	1.81	2.61
5. PREX	1.59	1.33	1.22	1.00	0.87	0.74	0.62
6. FCIL	1.43	1.30	1.10	1.0	0.87	0.73	0.62
7. SCED	n/a	1.43	1.14	1.00	1.00	1.00	n/a

# Оценка длительности проекта

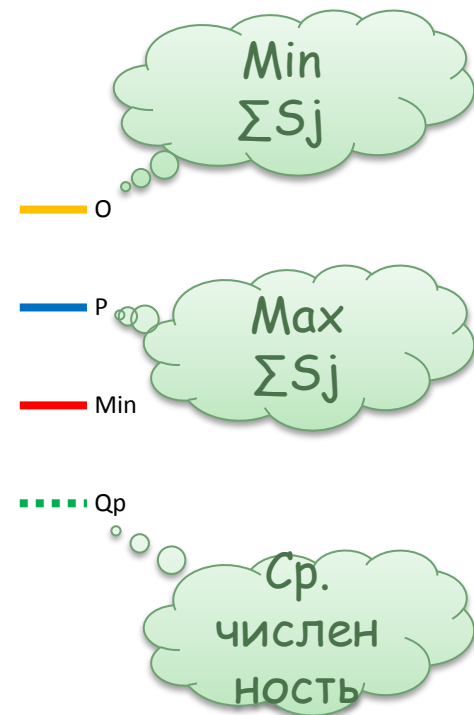
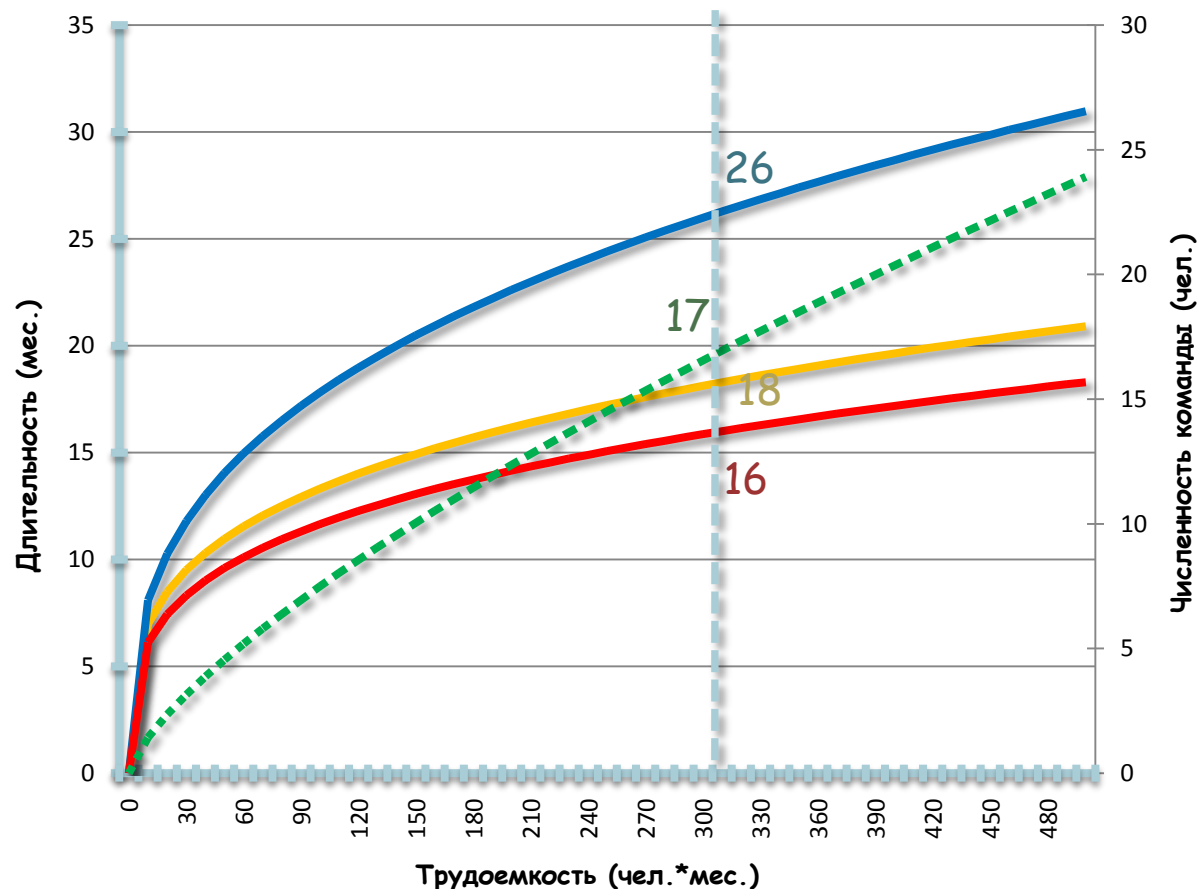
$$TDEV = C \times (PM_{NS})^{D+0,2 \times 0,01 \times \sum_{j=1}^5 SF_j} \times \frac{SCED}{100}$$

где

$$C = 3,67; \quad D = 0,28;$$

$PM_{NS}$  – трудоемкость без учета сжатия( $SCED$ )

# Зависимость длительности проекта от трудоемкости



# Необоснованный оптимизм

- Разработчики всегда называют слишком оптимистичные сроки.
- «Над этим проектом мы будем работать более эффективно, чем на предыдущем».
- «Если проектом грамотно управлять, то можно сделать быстрее».
- Необоснованные ожидания на применение новых технологий и средств разработки. Средняя производительность в отрасли растет только на 3-5% в год (Демарко).



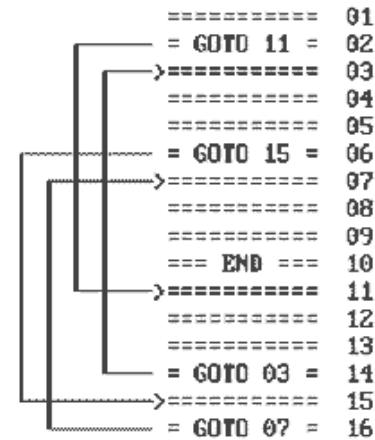
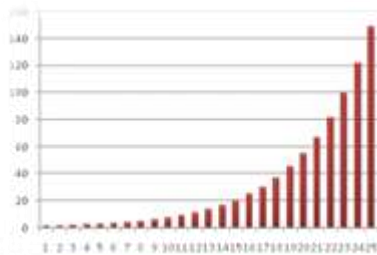
# Директивное занижение срока

- Боязнь переоценки со стороны руководства и/или Заказчика.  
Закон Паркинсона.
- «Студенческий синдром».  
Опасение, что если разработчикам будет выделено слишком много времени, то в начале они будут работать спустя рукава, а перед завершением начнется аврал.



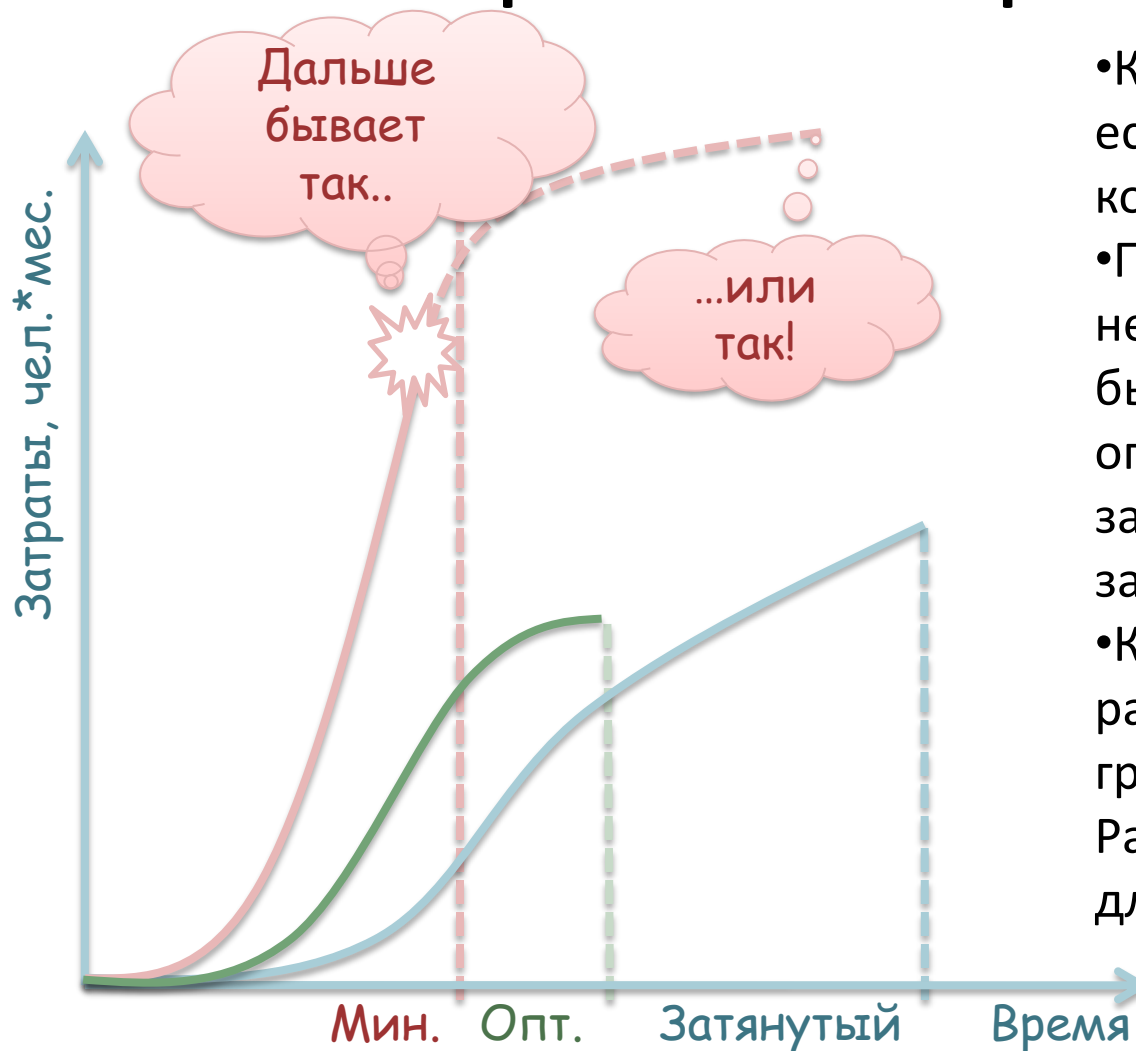
# Последствия недооценки

- Демотивация
- 40% ошибок из-за стресса.
- Отсутствие анализа и проектирования.
- Решение проблем наспех, обходными путями.
- Большой проблемный код.
- Большие затраты на исправление ошибок и внесение изменений.
- В случае недооценки проекта *потери растут нелинейно и неограниченно.*





# Правила Бари Боэма



- Кривая стоимости резко растет, если запланированный график короче номинального.
- Практически ни один проект невозможно завершить быстрее, чем за 3/4 расчетного оптимального графика вне зависимости от количества занятых в нем!
- Кривая стоимости медленно растет, если запланированный график длиннее номинального. Работа занимает все отведенное для нее время.

# Источники и дополнительная литература

- *С. Макконнелл, «Сколько стоит программный проект», «Питер», 2007.*
- *Function Point Counting Practices Manual, Release 4.2, IFPUG, 2004.*
- *Barry Boehm, et al. «Software cost estimation with COCOMO II». Englewood Cliffs, NJ:Prentice-Hall, 2000.*
- *С. Архипенков “Лекции по управлению программными проектами”, М., 2009 (<http://www.arkhipenkov.ru>)*