**Лекция**

# ОЦЕНКА ТРУДОЕМКОСТИ И СРОКОВ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Король Иван Андреевич –

доцент, канд. физ.-мат. наук

**ВОПРОСЫ:**

- 1. Оценка – вероятностное утверждение
- 2. Негативные последствия «агрессивного» расписания
- 3. Прагматичный подход. Метод PERT
- 4. Обзор метода функциональных точек
- 5. Обзор методики COCOMO II
- 6. Выводы
- 7. Контрольные вопросы

- Оценка трудоемкости разработки программного обеспечения должна **быть вероятностным утверждением** [1].
- Это означает, что для нее существует некоторое **распределение вероятности**, которое может быть очень широким, если неопределенность высокая, или достаточно узким, если неопределенность низкая.
- Использование собственного опыта или опыта коллег, полученного в похожих проектах – это наиболее **прагматичный подход**, который позволяет получить достаточно реалистичные оценки трудоемкости и срока реализации программного проекта, быстро и без больших затрат. **Прагматичный подход (метод PERT)**


- 1. Макконнелл, С. Сколько стоит программный проект / С. Макконнелл. – С.-Пб. : «Питер», 2007. – 297 с.

- Если собственный опыт аналогичных проектов отсутствует, а коллеги-эксперты недоступны, то необходимо использовать **формальные методики, основанные на обобщенном отраслевом опыте**.
- Среди них наибольшее распространение получили **два подхода:**
  - - FPA IFPUG – метод функциональных точек;
  - - COCOMO II – модель издержек разработки.
- В Беларуси введена своя методика, утвержденная Постановлением Министерства труда и социальной защиты Республики Беларусь от 27.06.2007 № 91 «**Об утверждении укрупненных норм затрат на разработку программного обеспечения**».

- Program (Project) Evaluation and Review Technique (PERT) – это **метод анализа задач**, необходимых для выполнения проекта, в особенности, анализа времени, которое требуется для выполнения каждой отдельной задачи, а также определение минимального необходимого времени для выполнения всего проекта.
- PERT предназначен для очень **масштабных, единовременных, сложных и нерутинных проектов**.
- Метод подразумевал **наличие неопределённости**, давая возможность разработать рабочий график проекта без точного знания деталей и необходимого времени для всех его составляющих.

- Самой популярной частью PERT является метод критического пути, опирающийся на построение сетевого графика (сетевой диаграммы PERT). Метод критического пути – инструмент планирования расписания и управления сроками проекта.
- В основе метода лежит определение наиболее длительной последовательности задач от начала проекта до его окончания с учетом их взаимосвязи.
- Задачи, лежащие на **критическом пути** (*критические задачи*), имеют нулевой резерв времени выполнения, и, в случае изменения их длительности, изменяются сроки всего проекта.

- В связи с этим, при выполнении проекта критические задачи требуют более **тщательного контроля**, в частности, своевременного выявления проблем и рисков, влияющих на сроки их выполнения и, следовательно, на сроки выполнения проекта в целом.
- В процессе выполнения проекта **критический путь проекта может меняться,** так как при изменении длительности задач некоторые из них могут оказаться на критическом пути.
- Входом для данного метода оценки служит **список элементарных пакетов работ (задач)**, которые нецелесообразно дальше декомпозировать.

- При этом нет необходимости точно знать **закон распределения нашей оценки трудоемкости** каждого такого элементарного пакета.
- Диапазон неопределенности некоторого проекта *i* достаточно охарактеризовать **тремя оценками**:
  - $M_i$ – **наиболее вероятная оценка трудозатрат**;
  - $O_i$ – **минимально возможные трудозатраты** на реализацию пакета работ; ни один риск не реализовался; быстрее точно не сделаем; вероятность того, что мы уложимся в эти затраты, равна 0;
  - $P_i$ – **пессимистическая оценка трудозатрат**; все риски реализовались.

- **Оценка средней трудоемкости** по каждому элементарному пакету $E_i$ определяется по формуле:

- $E_i = (P_i + 4M_i + O_i)/6.$

- Для расчета **среднеквадратичного отклонения** $\boldsymbol{CKO_i}$ используется формула:

- $CKO_i = (P_i - O_i)/6.$

Если оценки трудоемкости элементарных пакетов работ статистически независимы, и не испорчены, например, необоснованным оптимизмом программистов, то согласно центральной предельной теореме теории вероятностей **суммарная трудоемкость проекта (*E*)** может быть рассчитана по формуле:

$$E = \sum_{i=1}^{n} E_i \ .$$

А **среднеквадратичное отклонение (*СКО*) для оценки суммарной трудоемкости** будет составлять:

$$СКО = \sqrt{\sum_{i=1}^{n} СКО_i^2} \ .$$

- Тогда для оценки **суммарной трудоемкости проекта,** которую мы не превысим с вероятностью 95 %, можно применить формулу:

- $E_{95\%} = E + 2 * СКО.$

- Это значит, что вероятность того, что проект превысит данную оценку трудоемкости, **составляет всего 5 %.**
- А это уже вполне **приемлемая оценка**, под которой может расписаться профессиональный менеджер.

- Анализ функциональных точек – стандартный метод измерения размера программного продукта с точки зрения пользователей системы.
- Метод разработан **Аланом Альбрехтом** (Alan Albrecht) в середине 70-х годов XX в. и впервые опубликован в 1979 г.
- В 1986 г. была сформирована **Международная ассоциация пользователей функциональных точек** (International Function Point User Group – IFPUG), которая опубликовала несколько ревизий данного метода [2].


- 2. Function Point Counting Practices Manual. – Release 4.2. – Princeton Junction : IFPUG, 2004. – 150 p.

- **Основная идея метода** – максимальный отказ от деталей реализации программного обеспечения и перенос оценки в область функциональности, наблюдаемой пользователем.
- Метод предназначен для оценки на основе **логической модели объема программного продукта количеством функционала**, востребованного заказчиком и поставляемого разработчиком.
- **Несомненным достоинством метода** является то, что измерения не зависят от технологической платформы, на которой будет разрабатываться продукт, и он обеспечивает единообразный подход к оценке всех проектов в компании.

- При анализе методом функциональных точек надо выполнить следующую **последовательность шагов**:
  - - определение типа оценки;
  - - определение области оценки и границ продукта;
  - - подсчет функциональных точек, связанных с данными;
  - - подсчет функциональных точек, связанных с транзакциями;
  - - определение суммарного количества не выровненных функциональных точек;
  - - определение значения фактора выравнивания;
  - - расчет количества выровненных функциональных точек.

- COnstructive COst MOdel (COCOMO – модель издержек разработки) – это **алгоритмическая модель оценки стоимости разработки программного обеспечения**, разработанная Барри Боэмом (Barry Boehm).
- Модель использует простую формулу регрессии с параметрами, определенными из данных, собранных по ряду проектов. Впервые опубликована в 1981 г. в [3] в виде результата анализа 63 проектов компании TRW Aerospace.
- В 1997 г. методика была усовершенствована и получила название COCOMO II [4]. Калибровка параметров производилась по 161 проекту разработки.


- 3. Boehm, B. Software engineering economics / B. Boehm. – Englewood Cliffs, NJ : Prentice-Hall, 1981. – 767 p.

- Различаются **две стадии оценки** проекта: ***предварительная*** оценка на начальной фазе и ***детальная*** оценка после проработки архитектуры.
- Формула **оценки трудоемкости проекта** в чел.*мес. имеет вид:

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

- где $A$ = 2,94; $E = B + 0{,}01 \times \sum_{j=1}^{5} SE_j$ $B$ = *0,91;*
- ***SIZE* – размер программного продукта в тысячах строках исходного кода (Kilo Source Lines of Code – KSLOC);**
- $EM_i$ – множители трудоемкости; *S*
- $E_j$ –факторы масштаба; n = 7 (для предварительной оценки); n = 17 (для детальной оценки).

- Главной особенностью методики является то, что для того, чтобы оценить трудоемкость, **необходимо знать KSLOC**.
- Размер программного продукта может быть, например, **оценен экспертами** с применением метода PERT.

- Рассмотрим следующие **восемь особенностей** методики оценки трудоемкости разработки ПО,
- утвержденной Постановлением Министерства труда и социальной защиты Республики Беларусь от 27.06.2007 № 91
- «**Об утверждении укрупненных норм затрат на разработку программного обеспечения**».

- 1) Укрупненные нормы затрат труда (далее – укрупненные нормы) на разработку ПО являются основой для **определения общей трудоемкости разработки ПО**, **объемов финансирования** на стадии его технико-экономического обоснования.
- На основе общей трудоемкости разработки ПО составляется **смета затрат**, а также определяется **численность исполнителей** (соисполнителей) и **трудоемкость выполняемых ими работ по этапам** разработки ПО.
- Укрупненные нормы могут использоваться для разработки в организациях **локальных нормативных правовых ак**тов об утверждении норм затрат труда на разработку ПО с учетом конкретных организационных и технологических особенностей разработки ПО.

- 2) В **основу разработки укрупненных норм положены**:
  - - результаты анализа фактических затрат труда на разработку ПО;
  - - экспертные оценки;
  - - данные оперативного учета и отчетности;
  - - результаты анализа ранее действовавших и действующих в настоящее время норм труда по разработке ПО в Российской Федерации и других странах.

- 3) **Стадиями разработки ПО** согласно ГОСТам Единой системы программной документации являются:
  - - техническое задание (ТЗ);
  - - эскизный проект (ЭП);
  - - технический проект (ТП);
  - - рабочий проект (РП);
  - - ввод в действие (ВН).

- 4) Каждая стадия разработки ПО предусматривает выполнение **следующих видов работ**:
- - **ТЗ** – постановку задачи; сбор исходных материалов; выбор и обоснование критериев эффективности и качества разрабатываемой программы; обоснование необходимости проведения научно-исследовательских работ; определение структуры входных и выходных данных; предварительный выбор методов решения задачи; обоснование целесообразности применения ранее разработанных программ; определение требований к техническим средствам; обоснование принципиальной возможности решения поставленной задачи; определение требований к программе; определение стадий, этапов и сроков разработки программы и документации на нее; выбор средств программирования; согласование и утверждение ТЗ;

- **ЭП** –
  - уточнение методов решения задачи;
  - разработку общего описания алгоритма решения задачи, общей структуры и компонентов;
  - разработку пояснительной записки, включая внешние интерфейсы и базы данных;
  - согласование и утверждение ЭП;

- **ТП** – уточнение структуры входных и выходных данных, логической структуры базы данных, внешних интерфейсов; разработку алгоритма решения задачи; определение формы представления входных и выходных данных; разработку структуры программы, уточнение структуры компонентов на уровне программных модулей; окончательное определение конфигурации технических средств; разработку плана мероприятий по разработке и внедрению программ; определение требований к испытанию (тестированию) программных модулей; разработку пояснительной записки; согласование и утверждение ТП;

- **РП** – программирование и отладку программы; изготовление программы-ориги-нала; разработку программных документов в соответствии с требованиями ГОСТов; разработку, согласование и утверждение порядка и методики испытаний; проведение испытаний (тестирование) программных модулей, базы данных; корректировку программы и программной документации по результатам испытаний;
- **ВН** – проведение приемосдаточных испытаний программы; оформление и утверждение акта о передаче программы в постоянную эксплуатацию.

- 5) **Укрупненные нормы определены** на одно ПО и указаны в человеко-днях при пятидневной рабочей неделе с продолжительностью рабочего дня восемь часов с учетом времени на подготовительно-заключительные работы, обслуживание рабочего места, отдых (включая физкультурные паузы) и личные надобности в размере 10 % от нормируемых затрат труда, а также следующих **факторов, влияющих на трудоемкость разработки ПО**:
  - - объема ПО в строках исходного кода (LOC);
  - - сложности разрабатываемого ПО;
  - - степени новизны разрабатываемого ПО;
  - - степени использования в разработке стандартных модулей;
  - - условий и средств разработки ПО.
- В случае применения иных режимов рабочего времени нормы затрат труда **пересчитываются**.

- 6) В качестве единицы измерения объема ПО используется **строка исходного кода (LOC)**. **Преимущества** использования строки исходного кода (LOC) как единицы измерения заключаются в том, что эта единица измерения:
  - - отражает содержание труда программистов;
  - - позволяет выполнять сопоставление размеров ПО и производительности в различных группах разработчиков;
  - - непосредственно связана с разрабатываемым ПО;
  - - может использоваться для оценки работ до завершения проекта;
  - - позволяет автоматизировать сбор данных о количестве строк исходного кода (LOC) от начала до конца проекта;
  - - дает возможность учитывать мнение разработчика об объеме ПО на основе количества написанных строк исходного кода.

- 7) При **подсчете строк исходного кода (LOC) следует придерживаться следующих рекомендаций**:
  - - учитывать «строку исходного кода (LOC)» как одну, если в ней содержится лишь один оператор (если в одной строке содержатся два выполняемых оператора, разделяемых точкой с запятой, то нужно считать как две строки, а если один выполняемый оператор разбит на две «физические» строки, то он будет учитываться как один оператор);
  - - учитывать все имеющиеся выполняемые операторы, поддерживаемые данным продуктом;
  - - не учитывать строки, содержащие комментарии;
  - - не учитывать отладочный код или другой временный код (пробное ПО, средства тестирования, инструменты разработки и прототипирования, другие инструментальные средства);
  - - учитывать каждую инициализацию, вызов или включение макроса в качестве части исходного кода;
  - - не учитывать повторно используемые операторы исходного кода.

- 8) На работы, не предусмотренные укрупненными нормами, **нормы затрат труда разрабатываются организациями на основании методов технического нормирования** и утверждаются в установленном порядке.

- **Нереалистичность оценок трудоемкости** и сроков разработки программной системы – один из серьезнейших демотивирующих факторов для разработчиков.
- **Недооценка** приводит к ошибкам планирования и неэффективному взаимодействию.
- Агрессивные сроки, постоянное давление, сверхурочные, авралы служат причиной того, что **затраты на проект растут экспоненциально и неограниченно**.

- Использование собственного опыта или опыта коллег, полученного в похожих проектах, это наиболее **прагматичный подход**, который позволяет получить достаточно реалистичные оценки трудоемкости и срока реализации программного проекта, быстро и без больших затрат.
- Если собственный опыт аналогичных проектов отсутствует, а коллеги-эксперты недоступны, то необходимо использовать **формальные методики, основанные на обобщенном отраслевом опыте**.

- Среди них наибольшее распространение получили **два подхода**:
- - FPA IFPUG – **метод функциональных точек**;
- - COCOMO II – **модель издержек разработки**.

- В Беларуси целесообразно использовать методику оценки трудоемкости разработки ПО, утвержденную Постановлением Министерства труда и социальной защиты Республики Беларусь от 27.06.2007 № 91 «**Об утверждении укрупненных норм затрат на разработку программного обеспечения**».

- 1. Что такое оценка трудоемкости разработки программного обеспечения?
- 2. Негативные последствия «агрессивного» расписания?
- 3. Расскажите о прагматичном подходе в методе PERT оценки трудоемкости проекта.
- 4. Обзор метода функциональных точек оценки трудоемкости проекта?
- 5. Обзор методики COCOMO II оценки трудоемкости проекта?

**Лекция**

# ОЦЕНКА ТРУДОЕМКОСТИ И СРОКОВ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Король Иван Андреевич –

доцент, канд. физ.-мат. наук