**Измерение размера ПО**

Среди факторов, влияющих на оценку, размер проекта является наиболее важным показателем. Хотя оценки размера недостаточно для понимания общей сложности разрабатываемого продукта, существует явная зависимость между размером проекта и его трудоемкостью.

Чтобы определить, по каким показателям можно сравнивать метрики размера ПО, необходимо рассмотреть основные группы метрик подробнее.

**Количество строк кода**

Количество строк кода – это размерно-ориентированная метрика программного обеспечения, в которой объем ПО рассчитывается исходя из количества строк в тексте исходного кода.

Эта методика возникла в 1950-е годы. Основным носителем информации в те времена была перфокарта, причем на одной перфокарте кодировалась одна строка исходного кода. Поскольку каждая строка кода являлась отдельным физическим носителем, можно было подсчитать число этих объектов и определить трудоемкость и производительность труда программистов.

Среди методик учета числа строк есть две основные:

* по числу физических строк (LOC) – определяется как общее число строк исходного кода, включая комментарии и пустые строки;
* по числу логических строк кода (LLOC) – определяется как общее количество команд и зависит от используемого языка программирования. Если язык поддерживает размещение нескольких команд в одной строке, то одна физическая строка должна быть учтена как несколько логических, если она содержит более одной команды языка.

Также имеются производные от основных методик, которые в зависимости от задачи могут содержать дополнительную информацию по следующим показателям:

* число пустых строк;
* число строк, содержащих комментарии;
* процент комментариев (отношение строк кода к строкам комментария, производная метрика стилистики);
* среднее число строк для функций (классов, файлов);
* среднее число строк, содержащих исходный код для функций (классов, файлов);
* среднее число строк для модулей.

У оценки по количеству строк кода есть несколько минусов. В современных высокоуровневых языках одну и ту же функциональность можно описать различным числом строк кода, поэтому для них данная метрика может слабо коррелировать с реальными трудозатратами.

Кроме того, при использовании современных средств разработки ПО часто используется генерация кода для определенных действий, что может еще сильнее усложнить итоговую взаимосвязь между числом строк и трудоемкостью.

При этом у измерений в строках кода есть ряд преимуществ. Например, данные по количеству строк в завершенных проектах или модулях программ могут быть легко собраны при помощи служебных средств интегрированных сред разработки (IDE) или специальных программ.

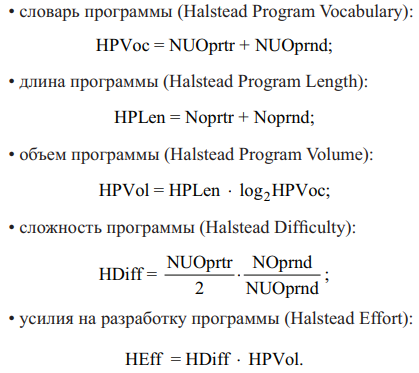
**Метод Холстеда**

Метрики, основанные на анализе числа строк и синтаксических элементов исходного кода программы, были предложены М. Холстедом (MauriceHalstead) в 1977 г.. Метрики Холстеда (Halstead complexity measures) частично позволяют учесть возможность реализации одной и той же функциональности разным числом строк и операторов. Наиболее частым сценарием использования этих метрик является оценка сложности промежуточных продуктов разработки, однако набор метрик также содержит оценки размера.

Основу метрики Холстеда составляют четыре измеряемые характеристики программы:

* NUOprtr (Number of Unique Operators) – число уникальных операторов программы, включая символы-разделители, имена процедур и знаки операций (словарь операторов);
* NUOprnd (Number of Unique Operands) – число уникальных операндов программы (словарь операндов);
* Noprtr (Number of Operators) – общее число операторов в программе;
* Noprnd (Number of Operands) – общее число операндов в программе.

На основе этих характеристик вычисляют различные метрики размера, сложности и качества программ, такие как:



Хотя метрики Холстеда позволяют использовать дополнительные возможности по анализу исходного кода, которые отсутствуют в метрике числа строк кода, с точки зрения оценки размера проекта большая часть задач остается нерешенной. Оценка общего числа операторов и их операндов до завершения этапа разработки программы может оказаться еще более сложной задачей, чем оценка числа строк кода.

**Функциональные точки**

Анализ функциональных точек (Function points) – это метод измерения размера программного обеспечения с точки зрения пользователей системы [8]. Метод был разработан Аланом Альбрехтом (Alan Albrecht) в середине 1970-х годов, впервые опубликован в 1979 г. Широкое распространение эта методика получила в середине 1980-х годов, после того как была сформирована организация IFPUG, занимающаяся развитием метода, а также публикующая новые версии [9].

Данный метод предназначен для оценки объема программного продукта по функциональной модели, т. е. оценивается объем функций разрабатываемой системы. Основная цель этого метода – в декомпозиции системы таким образом, чтобы обеспечить приемлемую сложность анализа. Базовой единицей измерения, на которой основывается данный метод, является функциональная точка. Оценка размера в функциональных точках базируется на количестве и сложности следующих элементов:

• внешних входных элементов – всех элементов, предназначенных для ввода или управления данными, которые поступают в систему;

• внешних выходных элементов – всех элементов для ввода и управления данными, которые выходят за внешние границы системы;

• внешних запросов – комбинации входных и выходных элементов, в которых входной элемент сопоставляется с простой выходной формой;

• внутренних логических файлов – каждого логического файла (группы данных), который создается или используется в системе; •

• внешних интерфейсных файлов – каждого файла под управлением другой системы, с которым взаимодействует измеряемая программа

Анализ функциональных точек включает в себя следующие этапы [10]:

1) определения типа оценки;

2) определения области оценки и границ продукта;

3) подсчета функциональных точек, связанных с данными;

4) подсчета функциональных точек, связанных с транзакциями;

5) определения суммарного количества не выровненных функциональных точек (UFP);

6) определения значения коэффициента выравнивания (FAV);

7) расчета количества выровненных функциональных точек (AFP).

На двух первых этапах анализируется предмет оценки, определяются границы продукта, выявляется разрабатываемая функциональность.

На этапах 3–5 происходят общий подсчет и суммирование функциональных точек без учета коэффициента выравнивания.

При выполнении 6-го пункта в методе расчета вводятся общесистемные требования, которые накладывают различные ограничения и увеличивают сложность разработки.

Сложность этих требований оцениваются коэффициентом выравнивания (FAV), который зависит от 14 общих системных характеристик (total degree of infl uence, TDI) и вычисляется по формуле



В целом измерение размера проекта функциональными точками является более актуальной метрикой, чем измерение числом строк кода. Ключевое преимущество этого подхода – оценка основана на требованиях к продукту, что позволяет оценить трудоемкость на самых ранних этапах работы над проектом, сразу после выявления необходимых требований. На последующих этапах работы оценку можно уточнить, поэтому этот метод можно применять при использовании гибких методологий разработки ПО. Также надо отметить, что декомпозиция системы, выполняемая на начальных этапах расчета, может в дальнейшем стать основой для документирования функциональности системы, а значит, снизить трудозатраты проектной команды. Однако использование функциональных точек в качестве единиц измерения имеет ряд существенных недостатков. Для вычисления функциональных точек необходимо детально изучить спецификацию требований и подсчитать все входные и выходные элементы, файлы, транзакции, что может быть весьма трудоемко. При этом полученная оценка весьма субъективна, поскольку некоторые бизнес-процессы могут иметь высокую алгоритмическую сложность, но обладать достаточно простыми внешними вводами и выводами. По мере развития методологий разработки программного обеспечения эксперты предложили новые методы оценки, основанные на методе функциональных точек.

**Объектные точки**

В современных методологиях разработки объектно-ориентированное программирование (ООП) занимает особое место, фактически является наиболее широко распространенной методологией.

Поскольку в изначальном варианте метода функциональных точек не было предусмотрено применения объектно-ориентированного подхода, был разработан адаптированный вариант, оперирующий терминами ООП.

Его принципиальным отличием от других вариаций метода функциональных точек является то, что он не расширяет стандартный набор типов элементов, а использует совершенно иные [13]:

• формы (Screens definitions);

• отчеты (User reports);

• модули (3GL Modules).

Фактически в данной метрике каждому уникальному классу или объекту назначается одна объектная точка. В целом оценка производится примерно по тем же этапам, что и функциональные точки: 1) подсчет количества форм, отчетов и компонентов; 2) классификация каждого экземпляра объекта по уровню сложности; 3) определение веса для объектов; 4) суммирование взвешенных объектов; 5) определение процента повторного использования кода; 6) определение уровня продуктивности; 7) подсчет значения длительности работы в человекомесяцах.

По сравнению с методом функциональных точек ощутимые различия есть при использовании факторов преобразования на этапах 5–7. Во-первых, подсчитывается процент повторного использования, исходя из него количество объектных точек пересчитывается

Этапы:

1) Подсчет количества форм, отчетов и компонентов;

2) Классификация каждого экземпляра объекта по уровню

сложности;

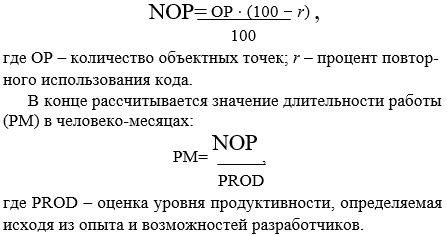
3) Определение веса для объектов;

4) Суммирование взвешенных объектов;

5) Определение процента повторного использования кода;

6) Определение уровня продуктивности;

7) Подсчет значения длительности работы в человекомесяцах.



**Метод UCP**

Метод UCP (Use-case points) представляет собой оценку размера проектов на основе диаграмм UML (Unified Modeling Language) и методологии RUP (Rational Unified Process). Как и многие другие современные методы оценки, UCP базируется примерно на тех же принципах, что и метод функциональных точек. Главное различие заключается в замене единиц измерения с функциональных точек на варианты использования (Use Cases).

Оценка по методу UCP складывается из следующих эта-пов [14]:

1. Оценка акторов. На этом шаге определяются все акторы системы (сущности, взаимодействующие с системой извне). После определения для каждого актора в соответствии с его оценкой устанавливается вес.
2. Нескорректированная оценка вариантов использования. Рассчитывается исходя из количества транзакций.
3. Оценка технических факторов. Используется для определения сложности архитектуры приложения и степени влияния нефункциональных требований.
4. Оценка внешних факторов. Используется для опреде-ления коэффициента влияния организационных рисков на разработку.
5. Окончательный подсчет. Оценивается общее число вариантов с учетом прочих факторов по формуле



где UUCW – нескорректированная оценка вариантов исполь-зования; UAW – оценка акторов; TCF – оценка технических факторов; ECF – оценка внешних факторов.