Технологии разработки программного обеспечения

Составитель: Эверстов В.В.

Дата составления: 14/05/2014

Дата модификации: 16/05/2014

Метрики ПО

- **Метрика программного обеспечения** — это мера, позволяющая получить численное значение некоторого свойства программного обеспечения или его спецификаций.

Метрики ПО

- порядок роста (имеется в виду анализ алгоритмов в терминах асимптотического анализа и O-нотации),
- количество строк кода,
- цикломатическая сложность,
- анализ функциональных точек,
- количество ошибок на 1000 строк кода,
- степень покрытия кода тестированием,
- покрытие требований,
- количество классов и интерфейсов.

Необходимость метрик

- Для чего нужны метрики?
- Для того чтобы оценить программный продукт
  - Качество,
  - Надежность,
  - Цену и т.д.

Оценка стоимости

- Для оценки стоимости программного продукта в основном используют две метрики, которые мы рассмотрели ранее. Это
  - количество строк кода (Line of Code, LOC),
  - функциональные точки (Function Points, FP)

LOC

- Преимущества использования этого критерия
  - Простота.
- Но он имеет кучу недостатков:
  - Размер проекта в терминах LOC может быть определен только после его завершения,
  - Зависит от языка программирования,
  - Не качественная оценка работы программиста,
  - Не отражает функциональные свойства кода программы.

FP

- Данная метрика ПО была разработана в противовес LOC в 70-х года прошлого столетия А. Дж. Альбректом. Он разработал эту методику для компании IBM для оценки проектов, которая не зависит от языка программирования и среды разработки.

FP

- Методика анализа FP основывается на концепции разграничения взаимодействия. Сущность ее состоит в том, что программа разделяется на классы компонентов по формату и типу логических операций.

Классы компонентов FP

- внутренний логический файл Internal Logical File (ILF) – группа логически связанных данных, находящихся внутри границ приложения и поддерживаемых вводом извне;
- внешний интерфейсный файл External Interface File (EIF) – группа логически связанных данных, находящихся вне границ приложения и являющихся внутренним логическим файлом для другого приложения;
- внешний ввод External Input (EI) – транзакция, при выполнении которой данные пересекают границу приложения извне;
- внешний вывод External Output (EO) – транзакция, при выполнении которой данные пересекают границу приложения изнутри;
- внешний запрос External Inquiry (EQ) – транзакция, при выполнении которой происходит одновременный ввод и вывод.

Сложность

- Классы компонентов оцениваются по сложности. Выделяют три категории сложности:
  - Высокий,
  - Средний и
  - Низкий.

Сложность

- Для транзакций (EI, EO, EQ) уровень определяется по количеству файлов, на которые ссылается транзакция File Types Referenced (FTR) и количеству типов элементов данных Data Element Types (DET).
- Для ILF и EIF имеют значение типы элементов записей Record Element Types (RET) и DET.
- Типы элементов записей это подгруппа элементов данных в ILF или EIF. Типы элементов данных – это уникальное не рекурсивное поле подмножества ILF или EIF. Уровни сложности и соответствующие им значения FTR и DET описаны в FPCPM (*Software* engineering: IFPUG 4.1 Unadjusted functional size measurement method: Counting practices manual.– ISO/IEC.– 2003.)

Нескорректированные функциональные точки

- После того как выделены классы и каждому из них присвоен уровень сложности, производится подсчёт нескорректированных функциональных точек Unadjusted Function Point (UFP) по соответствующей формуле:

$$UFP = \sum_{i=1}^{5} \sum_{j=1}^{3} N_{ij} W_{ij}$$

- N_{ij} – количество экземпляров класса i сложности j, W_{ij} - его весовое значение.

Фактор регулирования стоимости

- При расчете фактора урегулирования стоимости учитываются 14 характеристик системы (GSC). Эти характеристики отражают возможность повторного использования кода, производительность, возможность распределённой обработки, и другие свойства приложения.

Фактор регулирования стоимости

- Каждому из характеристик присваивается значение от 0 до 5. Что является степенью влияния этой характеристики.
- Фактор регулирования стоимости высчитывается по следующей формуле:

$$VAF = 0.65 + [(\sum_{i=1}^{14} C_i) / 100]$$

- C_i – степень влияния i-ой GSC

Количество функциональных точек

$$FP = UAF * VAF$$

Недостатки

- Существуют приложения, в оценке которых использование стандартных функциональных точек не эффективно. Эти приложения следующие:
  - управление процессом в реальном времени,
  - математические вычисления,
  - симуляция,
  - системные приложения,
  - инженерные приложения,
  - встроенные системы.

Методы оценки стоимости

- Неалгоритмические методы,
- Алгоритмические методы.

Неалгоритмические

- Price-to-win,
- Оценка по Паркинсону,
- Экспертная оценка,
- Оценка по аналогии.

Price-to-win

- Метод основывается на принципе «клиент всегда прав». Суть метода состоит в том, что независимо от предполагаемых реальных затрат на разработку проекта, оценка стоимости ПО корректируется в соответствии с пожеланиями заказчика.

Оценка по Паркинсону

- Метод основывается на принципе: «Объем работы возрастает в той мере, в какой это необходимо, чтобы занять время, выделенное на ее выполнение».
- В применении к разработке программных проектов, закон Паркинсона используется в виде следующей схемы: **чтобы повысить производительность труда разработчика, необходимо уменьшить время, отведённое на разработку**.

Экспертная оценка

- Метод основывается на принципе экспертной оценки и применяется в проектах использующих новые технологии, новые процессы или решающих инновационные задачи. К процессу оценки привлекаются инженеры-разработчики, которые сами оценивают курируемую ими часть проекта.
- Предположения, на которых основывалась оценка отдельных экспертов, заносятся в протокол и открыто обсуждаются. В результате достигается баланс оценки при интеграции отдельных компонентов в общую систему. Далее следует очередная стадия покомпонентной оценки, и по мере увеличения количества итераций точность оценки увеличивается.

Оценка по аналогии

- Метод основывается на принципе аналогии. Оценка по аналогии, как и алгоритмические модели, использует эмпирические данные о характеристиках завершённых проектов. Ключевое различие состоит в том, что метод оценки по аналогии с помощью эмпирических данных позволяет отобрать схожие проекты.

Оценка по аналогии

- Схема оценки основанная на указанном принципе состоит из нескольких этапов.
  - На первом этапе осуществляется сбор данных по разрабатываемому проекту. В рамках ЖЦ ПО оптимальными формами для этого являются анализ требований и проектирование. На основе экспертной оценки производится отбор характеристик, по которым будут сравниваться проекты.
  - Следующий этап включает в себя поиск и анализ проектов «аналогичных» по выбранным характеристикам разрабатываемому. Результатом данного этапа является, как правило, несколько проектов имеющих наименьшие различия в численных значениях характеристик оценки.
  - Последним этапом является экспертная оценка разрабатываемого проекта, в которой значения, взятые из аналогичного проекта используются как базис оценки.

Алгоритмические методы

- Линейная модель,
- Модель Путнэма (SLIM),
- Модель COCOMO.

Линейная модель

- Самая простая модель, которая существует:

$$\text{Effort} = a_0 + \sum_{i=1}^{n} a_i x_i$$

- a_i выбираются таким образом, чтобы наилучшим образом подходить к характеристикам уже законченных проектов.

Модель Путнэма (SLIM)

- модель основывается на утверждении, что затраты на разработку ПО распределяются согласно кривым Нордена-Рэйли, которые являются графиками функции, представляющей распределение рабочей силы по времени.

$$y = y_0 \cdot \left(1 - e^{-t^2/2t_p^2}\right)$$

Модель Путнэма (SLIM)

$$Size = C \cdot E^{1/3} \cdot t^{4/3}$$

- где Size – размер кода в LOC
- C – технологический фактор
- E – общая стоимость проекта в человеко-годах
- t – ожидаемое время реализации проекта.

$$E = D_0 \times t_d^3$$

$$E = \left(D_0^{4/7} \quad C^{9/7} \right) \quad S^{9/7} \text{ и } t_d = \left(D_0^{-1/7} \cdot E^{-3/7} \right) \cdot S^{-3/7}$$

Модель Путнэма (SLIM)

$$E = 12^{5} \quad B \quad (SLOC/P)^{3} \quad (1/Schedule^{4})$$

- B – фактор специальных навыков,
- P – фактор продуктивности,
- Schedule – время разработки по графику (по плану)

$$E = 56.4 \quad B \quad (SLOC/P)^{\frac{9}{7}}$$

Модель COCOMO

- Семейство моделей COCOMO было создано в 1981 году на основе базы данных о проектах консалтинговой фирмы TRW.
- COCOMO представляет собой три модели, ориентированные на использование в трех фазах жизненного цикла ПО:
  - базовая (Basic) применяется на этапе выработки спецификаций;
  - расширенная (Intermediate) – после определения требований к ПО;
  - Advanced – углубленная используется после окончания проектирования ПО.

Модель COCOMO

$$E = a \cdot S^b \times EAF$$

- где E – затраты труда на проект (в человеко-месяцах),
- S – размер кода (в KLOC),
- EAF – фактор уточнения затрат,
- a и b – зависят от разрабатываемого приложения.

COCOMO

- В базовой модели фактор EAF принимается равным единице.
- Для определения значения этого фактора в расширенной модели используется таблица, содержащая ряд параметров определяющих стоимость проекта.
- При использовании углубленной модели, вначале проводится оценка с использованием расширенной модели на уровне компонента, после чего каждый параметр стоимости оценивается для всех фаз ЖЦ ПО.

COCOMO II

- COCOMO II также является семейством моделей и представляет собой развитие базовой (Basic) модели COCOMO. COCOMO II включает три модели:
  - создания приложений Application Composition Model (ACM),
  - раннего этапа разработки Early Design Model (EDM) и
  - пост-архитектурная Post Architecture Model (PAM).

COCOMO II

- ACM используется на раннем этапе реализации, для того, чтобы оценить:
  - интерфейс
  - пользователя,
  - взаимодействие с системой,
  - производительность.
- За начальный размер принимается количество экранов, отчетов и 3GL – компонентов. Если предположить, что в проекте будет использовано r % объектов из ранее созданных проектов, количество новых объектных точек в проекте Object Points (OP) можно рассчитать, как

$$\mathrm{OP} = (\mathrm{object\ points})\mathrm{x}(100 - \mathrm{r})/100$$

COCOMO II

- Тогда затраты можно определить по следующей формуле:

$$E = OP / PROD$$

- где PROD – табличное значение

COCOMO II

- EDM – это высокоуровневая модель, которой требуется сравнительно небольшое количество исходных параметров. Она предназначена для оценки целесообразности использования тех или иных аппаратных и программных средств в процессе разработки проекта.
- Для определения размера используется неприведенная функциональная точка (Unadjusted Function Point). Для ее преобразования в LOC используются табличные данные. Уравнение модели раннего этапа разработки имеет вид

$$E = a\ LOC\ EAF$$

- где a – константа 2,45. EAF определятся так же, как и в оригинальной модели COCOMO. Параметры для EDM получаются комбинированием параметров для пост-архитектурной модели.

COCOMO II

- РАМ – наиболее детализированная модель, которая используется, когда проект полностью готов к разработке.
- Для оценки стоимости ПО с помощью РАМ необходим пакет описания жизненного цикла проекта, который содержит подробную информацию о факторах стоимости и позволяет провести более точную оценку.
- РАМ используется на этапе фактической разработки и поддержки проекта. Для оценки размеров могут использоваться как строки кода, так и функциональные точки с модификаторами, учитывающими повторное использование кода.
- Модель использует 17 факторов стоимости и 5 факторов, определяющих масштаб проекта (в модели COCOMO масштаб определялся параметрами вида приложения).

COCOMO II

- Уравнение PAM имеет вид

$$E = a \quad LOC^{b} \quad EAF$$

- где *a* принято за 2.55, $b = 1{,}01 + 0{,}01\sum W_i$,
- где W_i – параметры, отражающие свойства проекта, например, схожесть с ранее выполненными проектами, риск выбора архитектуры для реализации, понимание процесса разработки, сработанность команды разработчиков. Значения параметров являются табличными.

Программные средства оценки стоимости ПО

- SLIM Estimate - реализована модель Путнэма,
- Costar – основан на модели COCOMO II
- CoSeekMo -
- Cocomo II Application for Software Cost Estimation (C.A.S.E)
- SEER
- RASS Estimate
- EstimatorPal
- и т.д.

http://www.laatuk.com/tools/effort_estimation_tools.html