Лычковский М.С., Белавский А.С.,
Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь
Шнейдеров Е.Н., канд. техн. наук, доцент

МЕТРИКА АНАЛИЗА КОЛИЧЕСТВА СТРОК КОДА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

В приведенной статье приводятся теоретические сведения и ряд рекомендаций, касающиеся метрики анализа количества строк кода.

Количество строк кода (КСК) – это метрика программного обеспечения, используемая для измерения размера программы путем подсчета количества строк в тексте исходного кода программы. КСК, как правило, используется для

прогнозирования объема усилий, которые потребуются для разработки программы, а также для оценки производительности и удобства программирования после выпуска программного обеспечения.

Иногда в метрике КСК есть явно темная сторона, которую многие из нас наблюдали, но мало кто открыто обсуждал. С помощью метрик программного обеспечения, и иногда мы получаем побочные эффекты от метрики, которые затмевают любое значение, которое мы можем извлечь из информации о метриках. Независимо от того, правильны ли наши модели, и независимо от того, насколько хорошо или плохо мы собираем и вычисляем метрики программного обеспечения, поведение людей изменяется предсказуемым образом, чтобы предоставлять ответы, которые запрашивает руководство при применении метрик. По этой причине мы рассмотрим исходные строки метрик кода для нашего анализа. КСК метрика работает с исходными строками кода, но когда мы говорим о разных языках программирования, то строки кода для выполнения задачи меняются от языка к языку, потому что разные языки имеют разную структуру. Наш углубленный анализ в этой области дает исчерпывающее представление о том, что делает показатель КСК и насколько можно рассчитывать на показатель КСК.

Преимущества данной метрики:

1. Область для автоматизации расчета:

Поскольку строка кода является физической сущностью, ручной подсчет усилий может быть легко устранено путем автоматизации процесса подсчета. Для подсчета КСК в программе могут быть разработаны небольшие утилиты.

Однако утилита подсчета кода, разработанная для конкретного языка, не может использоваться для других языков из-за синтаксических и структурных различий между языками.

2. Интуитивная метрика:

Строка кода служит интуитивно понятной метрикой для измерения размера программного обеспечения, потому что его можно увидеть и эффект от него можно визуализировать. Говорят, что функциональные точки являются скорее объективной метрикой, которую нельзя представить как физическую сущность, она существует только в логическом пространстве. Таким образом, количество строк кода пригодится для выражения размера программного обеспечения среди разработчиков с низким уровнем опыта [1].

Недостатками данной метрики являются:

1. Разница в языках:

Рассмотрим два приложения, которые предоставляют одинаковую функциональность (экраны, отчеты, базы данных). Одно из приложений написано на C ++, а другое - на языке, подобном COBOL. Количество функциональных точек будет точно таким же, но аспекты приложения будут другими. Строки кода, необходимые для разработки приложения, конечно, не будут одинаковыми. Как следствие, количество усилий, необходимых для разработки приложения, будет другим (часы на функциональную точку). В отличие от строк кода, количество функциональных точек останется постоянным.

2. Отсутствие стандартов подсчета:

Не существует стандартного определения, что такое строка кода. Учитываются ли комментарии? Включены ли декларации данных? Что произойдет, если утверждение занимает несколько строк? – это вопросы, которые часто возникают. Хотя такие организации, как IEC и IEEE, опубликовали некоторые руководящие принципы в попытке стандартизировать подсчет, их трудно реализовать на практике, особенно в свете того, что с каждым годом вводятся все новые и новые языки[2].

3. Психология программиста:

У программиста, производительность которого измеряется строками кода, будет стимул писать излишне подробный код. Чем больше руководство сосредоточено на строках кода, тем больше у программиста стимулов расширять свой код с ненужной сложностью. Это нежелательно, так как повышенная сложность может привести к увеличению затрат на обслуживание и увеличению усилий, необходимых для исправления ошибок.

Таким образом можно сделать вывод что КСК является наиболее полезной метрикой в оценке программного обеспечения, но некоторые ограничения связанные с этой метрикой делают ее пригодной для использования.

Список использованных источников:

1. M. Kamalrudin International Journal of Soft. Eng. and its Applications / M. Kamalrudin // 9-10, 39-58 (2015)

2. ISO/IEC/IEEE 29148:2011 Systems and software engineering, Life cycle processes, Requirements engineering