Статический анализ качества кода

Статический анализ кода - это процесс выявления ошибок и недочётов в исходном коде программ. Статический анализ можно рассматривать как автоматизированный процесс инспекции кода. Он выполняется без исполнения исследуемого кода.

Статический анализ кода решает 3 задачи:

- Выявление **ошибок** и недочётов в программах.
- Проверка кода на соответствие рекомендациям по **оформлению кода** (применение отступов в различных конструкциях, использование пробелов/символов табуляции, регистра букв и так далее).
- **Подсчет метрик** качества кода (связность объектов, дублирование и так далее).

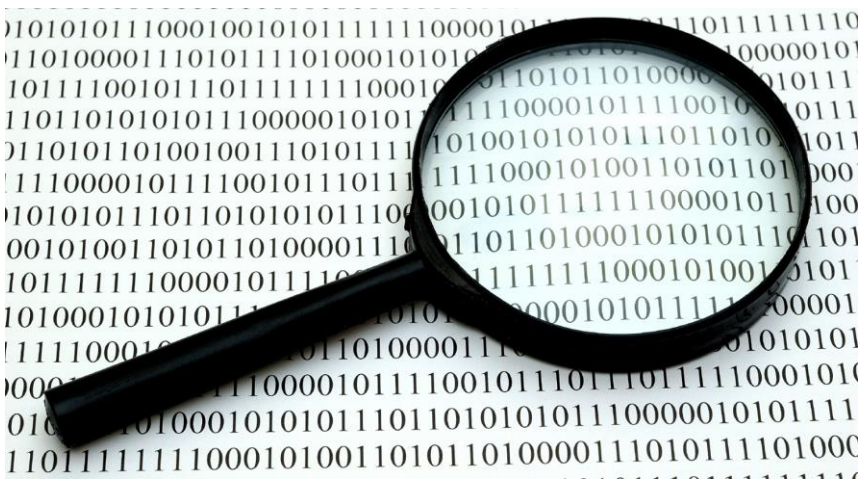
https://ru.wikipedia.org/wiki/Стандарт_оформления_кода

https://m.habrahabr.ru/company/pvs-studio/blog/327388/

https://ru.wikipedia.org/wiki/Метрика_программного_обеспечения

https://msdn.microsoft.com/ru-ru/library/bb385914.aspx

https://sbtatlas.sigma.sbrf.ru/wiki/pages/viewpage.action?pageId=43238082



Статический анализ качества кода. Ограничения

Результат работы статического анализатора — это список обнаруженных в коде потенциальных проблем с указанием имени файла и конкретной строки. Другими словами, это список ошибок, очень похожий на тот, что выдает компилятор. Термин «потенциальные проблемы» используется здесь не случайно. К сожалению, статический анализатор не может абсолютно точно сказать, является ли эта потенциальная ошибка в коде реальной проблемой. Это может знать только программист.

Статический анализ кода **целесообразно** применять не во всех проектах. Эффект от статического анализа будет очевиден на проектах примерно **от 3х человек** и **от полугода года** длительностью. Если программный проект меньше указанного размера, то вместо использования статического анализа выгоднее иметь в проекте нескольких квалифицированных разработчиков.

https://habrahabr.ru/company/intel/blog/103776/

Статический анализ качества кода. Практика Джона Кармака

«Самой важной вещью, которую я сделал как программист за последние годы — это начал агрессивно применять статический анализ кода. Не так важны сотни серьёзных багов, которые удалось предотвратить, сколько смена ментальности и моего отношения к надёжности ПО и качеству кода. Я всегда старался писать хороший код, поскольку одной из моих внутренних мотиваций было то, что как мастер своего дела я постоянно должен совершенствоваться.»

Джон Кармак в течение нескольких месяцев работал, постепенно исправляя найденные баги: сначала свой код, потом другой системный код, потом игровую часть. Эффект был колоссальным, нашлись очень важные ошибки, о которых в своё время попросту забыли, а также новые.

В конце концов, Джон Кармак сделал несколько выводов, которые он советует понять и другим разработчикам:

- каждый должен признать, что в его коде существует множество ошибок. Это горькая пилюля, которую требуется проглотить каждому программисту.

- ревизия кода должна быть обязательно автоматизирована. Приятно видеть ошибочные срабатывания автоматических систем на вашем коде, но на каждую ошибку автоматической системы приходится с десяток человеческих ошибок. Советы «писать лучший код», работать парами и так далее, тут не работают, особенно когда десятки программистов находятся в условиях нехватки времени.

Кармак также отмечает, что инструмент после каждого апдейта находит новые ошибки. То есть ошибки в коде вы никогда не сможете устранить до конца. В большом проекте они всегда будут, как статистические отклонения в свойствах физического материала.
Вы можете только попытаться минимизировать их негативный эффект.

“The first step is fully admitting that the code you write is riddled with errors” (John Carmack)

Статический анализ качества кода. Метрики качества

Контроль качества исходного кода предполагает наличие метрик, которые позволяют оценить достижение того или иного уровня качества программного проекта.

Метрика программного обеспечения (software metric) – численная мера, позволяющая оценить определенные свойства конкретного участка программного кода. Для каждой метрики обычно существуют ее эталонные показатели, указывающие, при каких крайних значениях стоит обратить внимание на данный участок кода.

Метрики кода разделяются на категории и могут оценивать совершенно различные аспекты программной системы: сложность и структурированность программного кода, связность компонентов, относительный объем программных компонентов и др.

Основные метрики *:

- Покрытие кода юнит тестами (Code coverage)
- Комментарии (Comments)
- Отсутствие связности внутри методов (LCOM4)
- Индикатор спутанности пакетов (Package tangle index)
- Процент дублирования кода (Duplications)
- Цикломатическая сложность (Complexity)

https://www.osp.ru/os/2009/08/10748698
https://dou.ua/lenta/articles/code-metrics/
https://ru.wikipedia.org/wiki/Метрика_программного_обеспечения
https://habrahabr.ru/post/205342/
http://cyberleninka.ru/article/n/problemy-izmereniya-kachestva-programmnogo-koda

** Не все метрики возможно измерить в рамках используемых технологий и средств*