ГОСТ Р 71207–2024 — Статический анализ программного обеспечения. Процессы

Вебинар от команды PVS-Studio

Андрей Карпов

DevRel

Андрей Карпов

- Один из основателей проекта PVS-Studio — pvs-studio.ru
- Пишу и рассказываю про статический анализ и качество кода

ГОСТ Р 71207–2024. Статический анализ ПО

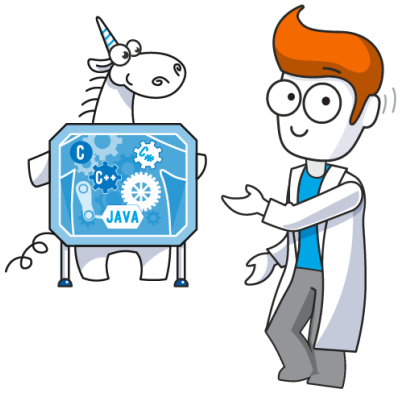
Предыдущие части:

Общее описание и актуальность	https://pvs-studio.ru/ru/blog/video/11050/
Терминология	https://pvs-studio.ru/ru/blog/video/11066/
Критические ошибки	https://pvs-studio.ru/ru/blog/video/11084/
Технологии анализа кода	https://pvs-studio.ru/ru/blog/video/11092/

Внедрение: подготовительный этап

Выбор инструмента статического анализа

- Приоритет следует отдавать инструментам, демонстрирующим лучшие показатели.
- Можно использовать совместно несколько инструментов, чтобы покрыть все используемые языки программирования.
- Следует использовать регулярно обновляемые анализаторы, поддерживающие:
  - современные стандарты языков программирования;
  - используемые в проекте системы сборки.



Подготовка сборочной среды

- Следует использовать среду анализа, позволяющую выполнить проверку ПО выбранным инструментом (инструментами) с учётом временных ограничений:
  - анализ проекта (**с учётом заимствованных компонент**) должен выполняться **не более чем 2 суток;**
  - простыми словами: всё должно проанализироваться за выходные ☺

Подготовка сборочной среды

- Допустимо разворачивать анализатор в виртуальной инфраструктуре или использовать в виде сервиса.
- P.S. **PVS-Studio работает в закрытом контуре**, но совместим с
  - Docker
  - WSL
  - CircleCI
  - Travis CI
  - GitLab
  - Azure DevOps
  - GitHub Actions

Внедрение: начальный этап

Начальный этап

- Настройка инструмента статического анализа для данного ПО.
  - Выбор и включение типов предупреждений, соответствующий списку **критических ошибок**.
  - Можно включить и другие типы предупреждений.
- Выполнение первичного статического анализа кода.
- Разметка полученных результатов и формирование начальной базы предупреждений о потенциальных ошибках.

Первичная разметка предупреждений

- Полученный набор предупреждений должен быть размечен.
- Результаты разметки должны быть сохранены для возможности сравнения результатов последующих запусков.
- После первичной разметки конфигурация статического анализатора может быть доработана.
- Примеры доработок:
  - подавление предупреждений в макросах (C, C++);
  - сокращённый набор правил для тестов.

PVS-Studio и первичная разметка предупреждений

1. Можно сохранить первоначальный отчёт и сравнивать с последующими, используя:
   - утилиту PlogConverter, входящую в дистрибутив PVS-Studio;
   - внешние платформы контроля качества кода:

     интеграция с SonarQube;

     интеграция с DefectDojo.

PVS-Studio и первичная разметка предупреждений

2. Использовать немного другой удобный подход
   - Просмотреть предупреждения про критические ошибки
     Ошибки исправить / Ложные разметить
   - Остальное скрыть созданием baseline-уровня сообщений (база разметки)
   - Возвращаться к техническому долгу и разбирать оставшиеся предупреждения.

Как внедрить статический анализатор кода в legacy проект и не демотивировать команду

После начального этапа проведения статического анализа

- Может быть принято решение о выборе другого анализатора.

Ещё про первичную разметку предупреждений

- Все предупреждения о критических ошибках должны быть отнесены к одной из категорий:
  - истинные предупреждения;
  - истинные предупреждения, но не требующие исправления кода;
  - ложные предупреждения.
- Допускается расширять перечень категорий.

Проведение регулярного статического анализа ПО

Важность регулярного использования

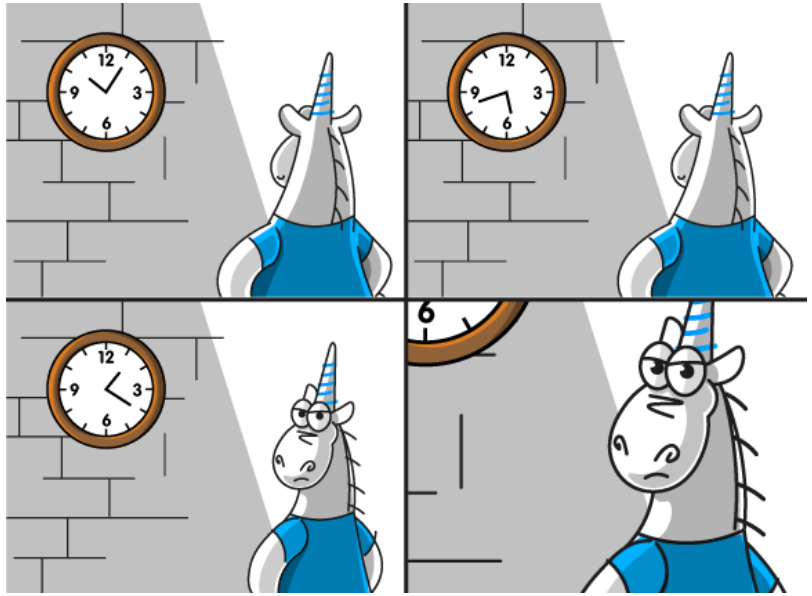
- Накопление непроанализированных изменений ухудшает качество проводимой экспертизы результатов анализа.
- Чаще возникают ситуации, при которых истинное предупреждение неверно классифицируется как ложное срабатывание.
- Увеличивается длительность проводимой экспертизы.
- Усложняется исправление ошибок в силу увеличившегося временного промежутка между внесением ошибки и правкой кода.

Регулярность

- Статический анализ всего ПО следует **проводить не реже одного раза в 10 рабочих дней, если код был изменён**.
- Статический анализ добавленных/изменённых частей следует проводить после каждого изменения.

Просмотр предупреждений

- При анализе **изменённых** частей ПО — не позже, **чем через три рабочих дня** после выполнения анализа.
- При анализе ПО целиком — не позже, **чем через 10 рабочих дней** после выполнения анализа.

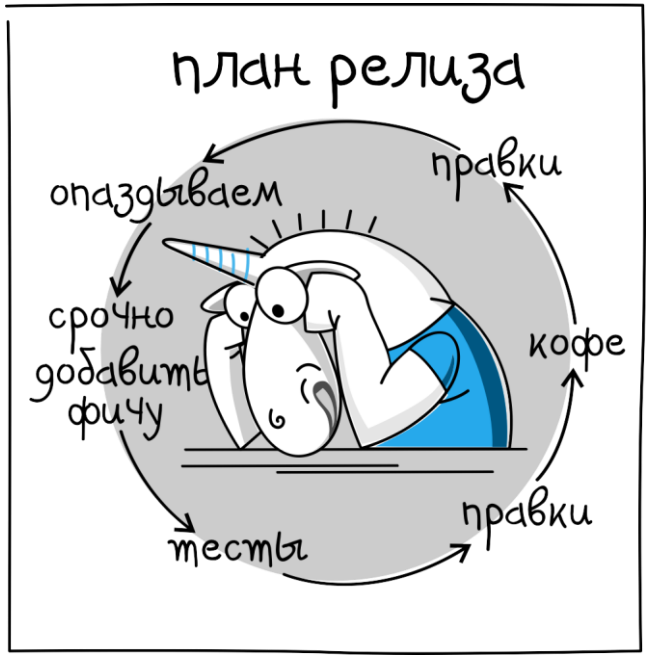


Правка ошибок

- Потенциальные ошибки, которые были отнесены к истинным, не позже, чем через **10 рабочих дней** после выполнения разметки, следует **исправить**.
- Либо запланировать сроки их устранения в соответствии с принятыми процессами разработки ПО.
- Если применяется гибкая методология разработки, то план по устранению ошибок включается в ближайший цикл разработки.

Правка ошибок

- При анализе проекта целиком, устранение найденных критических ошибок должно быть выполнено не позднее начала квалификационного тестирования.



Контроль

- Контроль над ходом устранения выявленных потенциальных ошибок следует выполнять **не реже одного раза в месяц**.
- Для контроля следует привлекать специалистов, которые не участвовали в процессах: анализа, просмотра предупреждений и устранения ошибок.

Что ещё

- Конфигурация и настройки анализатора должны пересматриваться при
  - изменении архитектуры;
  - добавлении сторонних компонент;
  - выходе новой версии статического анализатора.

Заключение

Статический анализ по ГОСТ — это хорошо

- Правильный выбор анализаторов кода
- Правильный процесс внедрения
- Правильный процесс регулярного использования

- Стандарт практичный и написан (достаточно) понятным.

- Считается, что сделанное по ГОСТ — это хорошее. За всё не скажу, но для статического анализа это подходит. Пользуйтесь.

Попробуйте PVS-Studio

PVS-Studio – pvs-studio.ru

- Разрабатывается с учётом требований, предъявляемых к статическим анализаторам в ГОСТ Р 71207–2024.
- Более 15 лет на рынке.
- C, C++, C#, Java.
- Классификация предупреждений: CWE, MISRA, AUTOSAR, OWASP, CERT.
- Работа в закрытом контуре.
- Входит в реестр отечественного ПО: №9837.
- Можно использовать для сертификации в испытательных лабораториях для проведения исследований до 4 уровня контроля.

Q&A

Андрей Карпов
DevRel