Десятая независимая научно-практическая конференция «Разработка ПО 2014»

23 - 25 октября, Москва

2014 CEE-SEC(R) РазработкаПО

Вопросы эффективности использования статических анализаторов на проектах с большим количеством строк кода

Александр Казанский

ООО АУРИГА

Цели исследования

- Сравнить инструменты между собой по выявляемым ошибкам
- Посмотреть отразится ли на количестве и качестве выявленых ошибок опыт команды

Исследуемый проект

- Исследовались две ревизии написанные разными по опыту командами
- Размер проекта – порядка 900 файлов и 400000 строк кода
- В проекте есть достаточно большое количество кода написанного более 15 лет назад и не удовлетворяющему современным стандартам языка

Используемые инструменты

- cppcheck 1.52
- Clang Static Analyzer based on LLVM 3.3 (из репозитория Ubuntu 12.04)
- Klocwork Insight 10.0.3.10317
- Parasoft C/C++test 9.5.3.61
- gcc 4.6.3

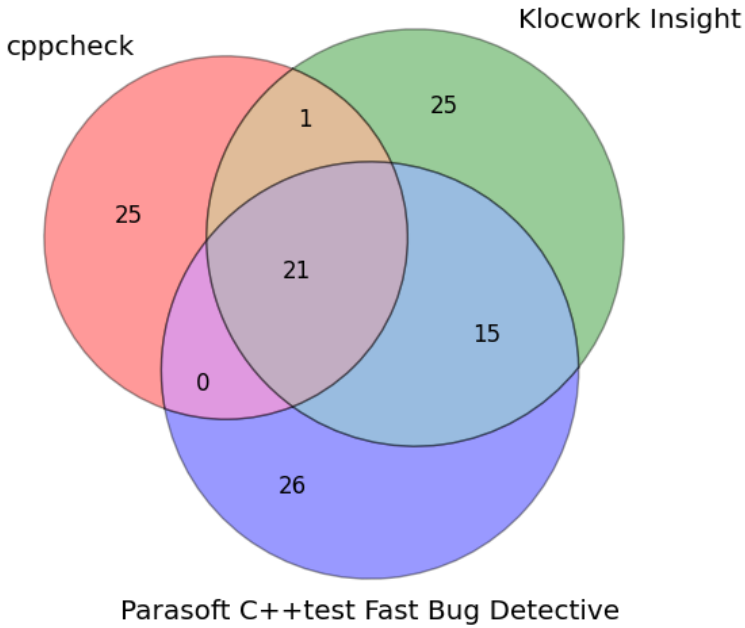
Типы выявленных ошибок

- Проблемы совместимости стандартов
- Неправильные данные в переменных
- Неправильная работа с памятью
- Неправильный формат данных в printf/scanf
- Ошибки при работе с классами или их создании
- Улучшения кода

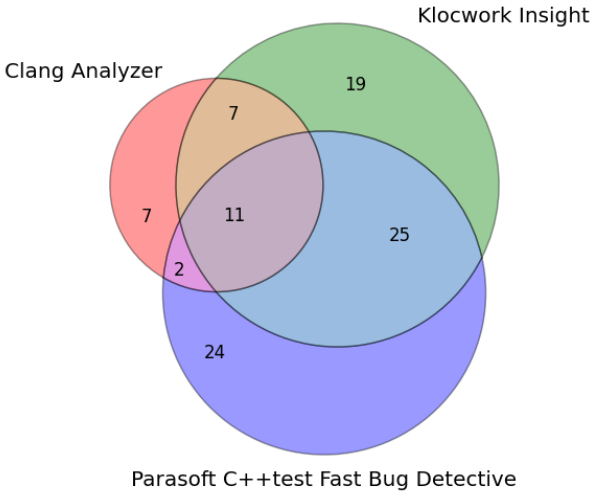
Неправильные данные в переменных

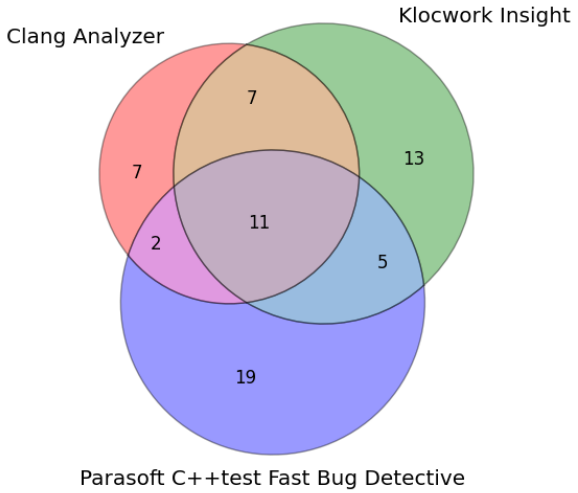
	Clang Static Analyzer	cppcheck	Klocwork Insight	Parasoft C++test FBD	предупреждения компилятора
Clang Static Analyzer	27	3	18	13	8
cppcheck	3	47	22	21	3
Klocwork Insight	18	22	62	36	10
Parasoft C++test FBD	13	21	36	63	6
предупреждения компилятора	8	3	10	6	11

Неправильные данные в переменных
Пересечение сообщений cppcheck, Parasoft C/C++test и Klocwork Insight

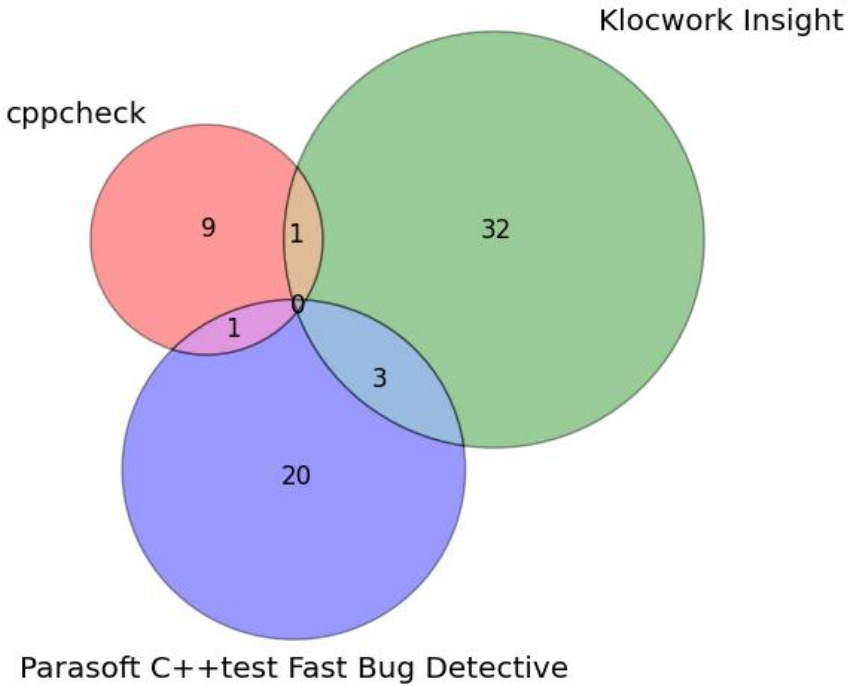


Неправильные данные в переменных
Пересечение сообщений Clang Static Analyzer, Parasoft C/C++test и Klocwork Insight





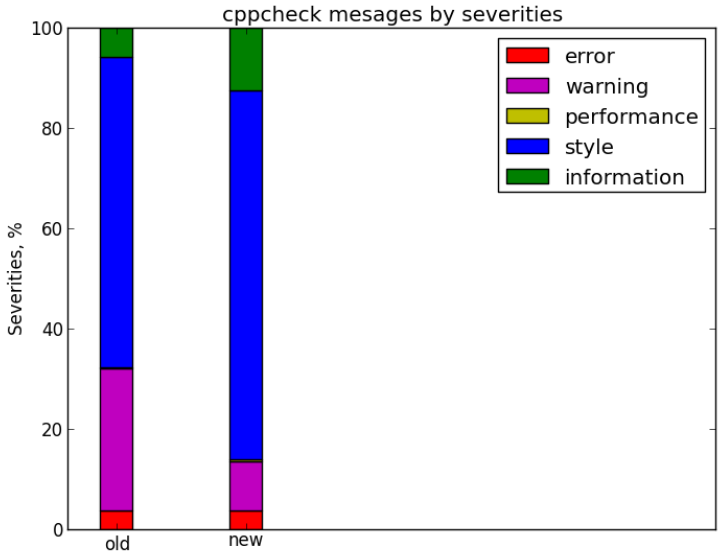
Неправильная работа с памятью

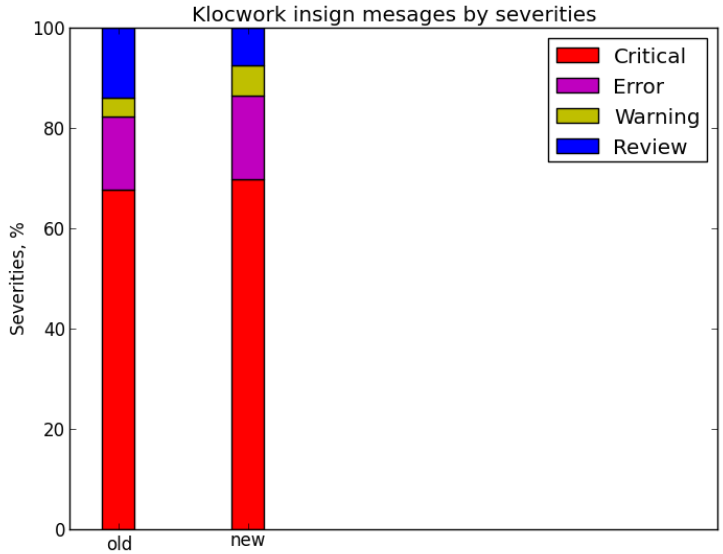


Сравнение ревизий между собой

	Ревизия 1 – команда джуниоров	Ревизия 2 – более опытная команда	Новое
Количество строк кода	393427	413373	
Файлов	881	965	
Предупреждений компилятора (-Wall)	2489	0	
cppcheck	1317	683	72
Clang Static Analyzer	101 + 10 файлов вызвали ошибку анализатора	66 + 2 файла вызвали ошибку анализатора	42, из них 7 в файлах которые в первой ревизии анализатор не смог проверить
Klocwork Insight	338	186	61
Распределение для KW по опасности сообщения, critical/error/warning/review	229/49/13/47	130/31/11/14	
Parasoft C/C++test, набор правил Recomended Rules	1700	1805	
строк кода на сообщение, набор правил Recomended Rules	231.43	229.02	
Parasoft C/C++test, набор правил BugDetective Fast	265	258	28

Сравнение ревизий между собой — распределение по уровням опасности





Ложноположительные срабатывания

Вынесены отдельно потому что формально это верное срабатывание, но по какой-то причине не важные в данном случае

Виды срабатываний:

- Специфика проекта (например, бесконечный цикл чтения из порта В отдельном потоке)
- Легко избегаемые программистом за счет именования

Пример кода с понятным человеку именованием:

```
int HR_action(HR_STAGE currentStage){
    static int staticValue = 0;
    int result = 0;

    switch (currentStage) {
    case HR_STAGE_INIT:{
        staticValue = 1;
        break;
    }
    case HR_STAGE_RUNNING:{
        result = 10/staticValue; //Возможное деление на нуль
        ++staticValue;
        break;
        }
    case HR_STAGE_FINAL:{
        staticValue = 0;
      }
    }
    return result;
}
```

Выводы

- Каждый из анализаторов имеет свои сильные и слабые стороны, лучше использовать как можно больше инструментов, причем желатительно из разных категорий (статический анализ, DRY, динамический анализ).
- Лучше всех и по количеству и по качеству показал себя Klocwork Insight
- Результаты проверки кода требуют ручной перепроверки ошибок программистом
- Более опытная команда допускает меньше ошибок, но качественное разделение их остается примерно тем же