Viva64.com

ПРИМЕНЕНИЕ ТЕХНОЛОГИИ СТАТИЧЕСКОГО АНАЛИЗА КОДА В СОВРЕМЕННОМ ПРОЦЕССЕ РАЗРАБОТКИ ПРОГРАММ

Авторы: Карпов А.Н., Рыжков Е.А.

Докладчик: Рыжков Е.А.

SECR-2007

Содержание

1. Введение.
2. Традиционный подход к обнаружению ошибок в новом коде.
3. Статический анализ кода в процессе разработки.
4. Возможности диагностики, предоставляемые анализаторами кода.
5. Заключение.

1. Введение

- Современные языки программирования являются достаточно мощными и гибкими, но сложными.
- В одном проекте участвуют, как правило, и опытные программисты, и новички, работу которых надо контролировать.
- Интересы бизнеса – новички меньше ошибаются, опытные меньше отвлекаются на контроль за новичками.

2. Традиционный подход к обнаружению ошибок в новом коде

- Код, разработанный начинающими программистами, всегда просматривается их более опытным коллегой (**code review**).
- Просмотр кода становится мало применим в крупных проектах, в силу их большого объема.
- Просмотр кода часто сводится к нечастым встречам, целью которых ставится обучение новых сотрудников, нежели чем для проверки работоспособностей разрабатываемых модулей.

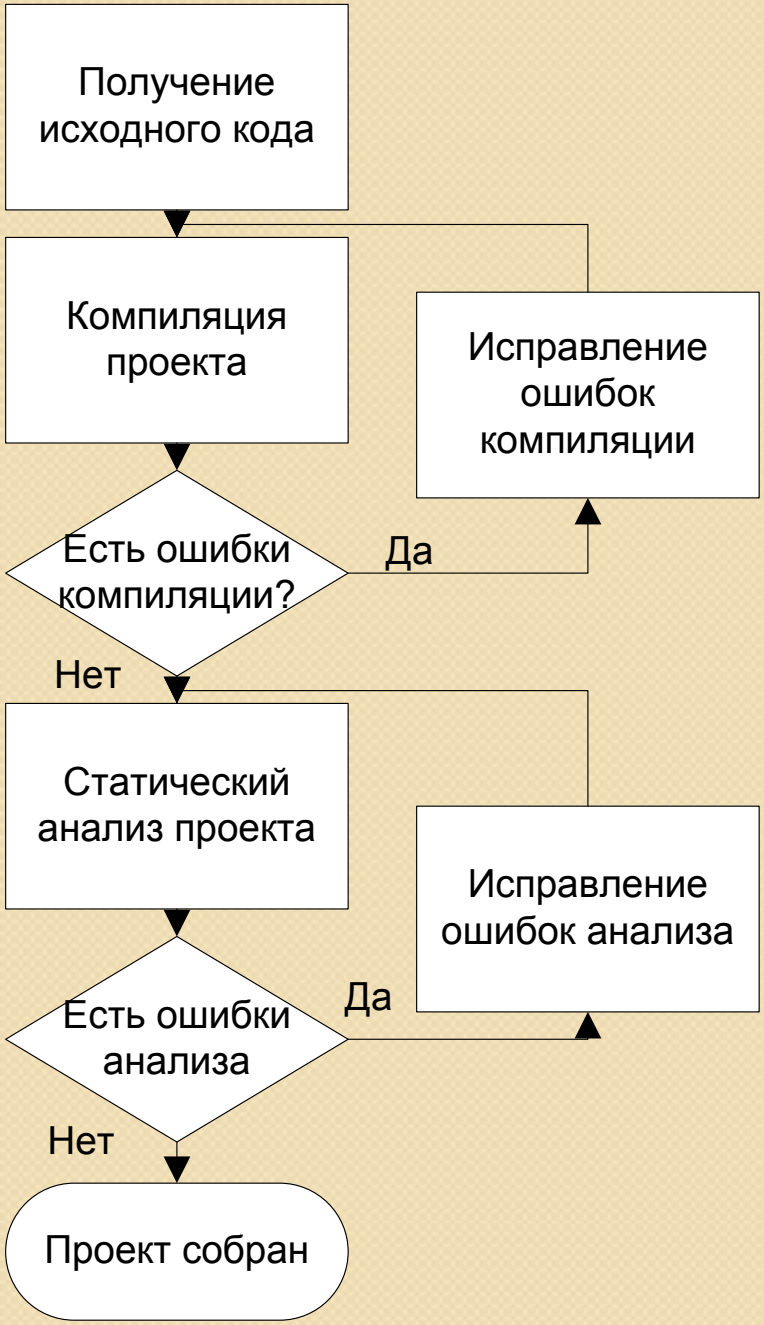
3. Статический анализ кода в процессе разработки

- Статический анализатор кода – инструмент, который разбирает и анализирует исходный код.
- Задача статического анализатора - сокращение объема кода, требующего внимания человека и тем самым сокращение времени его просмотра.

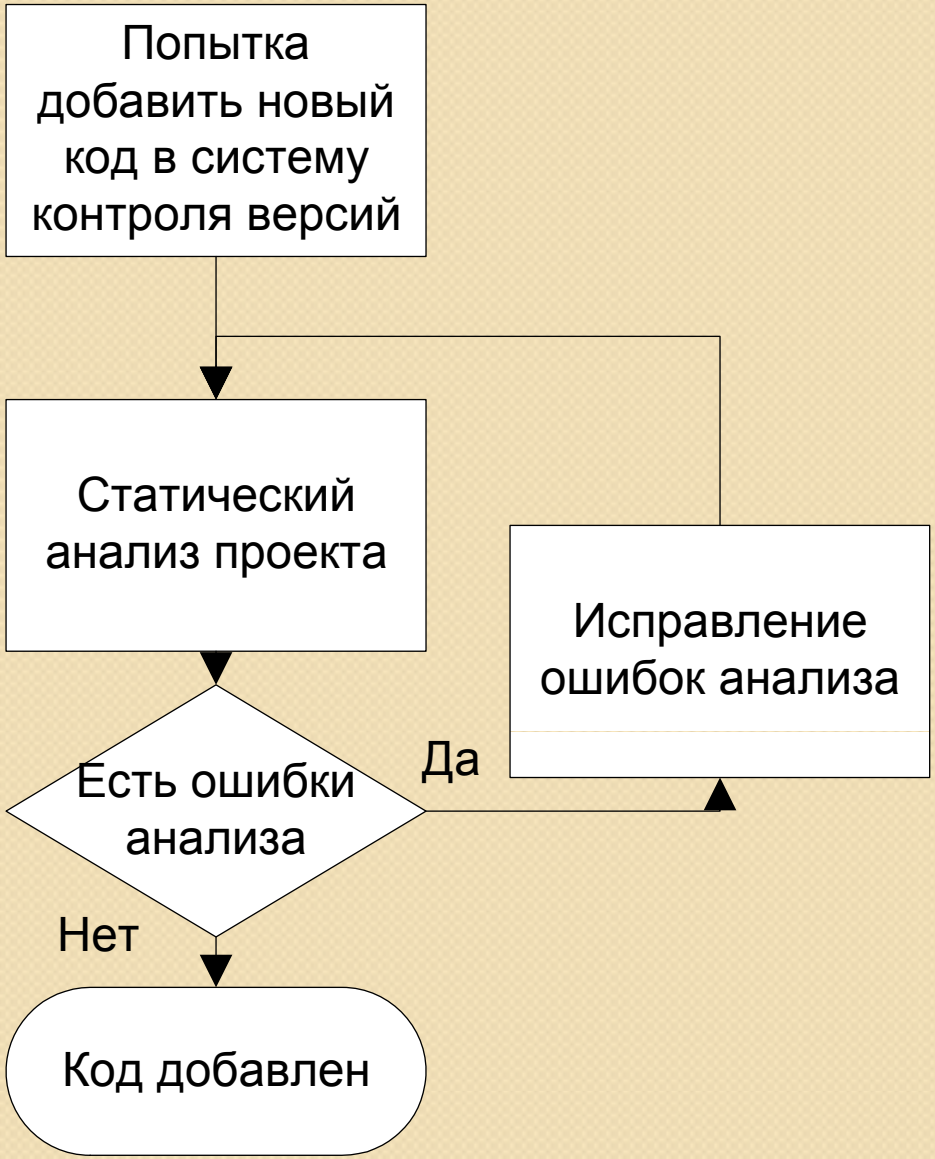
Интеграция статического анализатора в процесс разработки

- Интеграция статического анализатора в автоматическую систему сборки проекта.
- Автоматический запуск статического анализатора при добавлении кода в систему контроля версий.

Интеграция статического анализатора в автоматическую систему сборки



Интеграция статического анализатора в систему контроля версий



4. Возможности диагностики, предоставляемые анализаторами кода

- Возможности классических анализаторов кода.
- Возможности современных анализаторов кода.

Традиционные диагностические возможности статических анализаторов кода

- переполнение буфера;
- некорректная работа с типами данных, некорректное приведение типов;
- обращение к нулевым указателям;
- некорректная работа с памятью;
- использование переменных без инициализации;
- поиск неиспользуемого кода;
- …

(на примере PC-lint, C++test, FxCop, Klockwork Developer)

Новые возможности статических анализаторов кода

- Статический анализ для поддержки кросс-платформенной разработки.
- Статический анализ для миграции кода на новую платформу.
- Статический анализ многопоточных и многопроцессорных систем.

Статический анализ для поддержки кросс-платформенной разработки

- Кросс-платформенная разработка – это процесс разработки программ, которые способны работать в различных системах, например, и в Microsoft Windows, и в Unix.
- Статический анализатор может обнаруживать непереносимый код, а также разделять код на переносимый и непереносимый

Коммерческих решений пока нет

Статический анализ для миграции кода на новую платформу

- В настоящее время начинается процесс переноса приложений на 64-битные платформы.
- Статический анализатор кода, предназначенный для обнаружения потенциальных ошибок на 64-битных платформах, позволяет обнаружить некорректные конструкции до стадии собственно тестирования готового программного обеспечения.

Примеры: PC-lint, C++test, Viva64

Статический анализ многопоточных и многопроцессорных систем

- Анализаторы кода могут ориентироваться именно на ошибки многопоточных приложений: одновременный доступ к данным из разных потоков, взаимная блокировка, состояние гонки и так далее.

Коммерческих решений пока нет

5. Заключение

- Использование статических анализаторов не позволяет отказаться от тестирования разрабатываемых приложений, от создания юнит-тестов и других традиционных подходов к процессу разработки.
- Однако можно сэкономить время, которое с большей пользой стоит потратить не на поиск ошибок в коде, а на дополнительный контроль качества.