SelfTester – новая система тестирования на замену Viva64SelfTester

**Автор:** Евгений Рыжков

**Дата:** 01.07.2014

# Состояние дел

Существующая система Viva64SelfTester – самый технически успешный внутренний программный проект компании. В том смысле, что он живет и ежедневно используется нами больше любого другого проекта, включая основные наши продукты. Отдадим дань уважения этому проекту и похороним его в ближайшее время, так как он имеет несколько серьезных проблем:

* Невозможность "в один клик" выполнить все необходимые проверки. Например, хочется сначала проверить на базовом наборе тестах во всех версиях студии, а потом на наборе тестов для VS2010 запустится.
* Сложность дальнейшего расширения из-за очень большого количества изменений в проекте.

Поэтому необходимо разработать новый инструмент тестирования SelfTester (это новое название) с примерно такой же функциональностью.

Этот документ - не техническое задание, а описание концепции инструмента. На основе него уже надо будет разработать детальное техническое задание.

# Основные положения

* Инструмент работает со всеми версиями Visual Studio, начиная с VS2005.
* Инструмент НЕ будет работать с Embarcadero RAD Studio, так как мы планируем отказаться от поддержки этой среды в обозримом будущем.
* Инструмент НЕ будет уметь проверять MinGW проекты, т.к. это неоправданное сильное усложнение.
* Поддерживается только Visual Studio и все.
* Инструмент позволяет тестировать и PVS-Studio, и CppCat. Как по очереди, так и вместе.
* Инструмент содержит одну базу проверяемых проектов, часть из которых дает одинаковый .plog на всех версиях Visual Studio, а часть – проверяется только в конкретных версиях Visual Studio. База проектов содержит проекты из src-for-Viva64SelfTester.7z и src-for-Viva64SelfTester\_VS10.7z. Остальные проекты выкидываем.
* Среда разработки – VS2012, C#, WPF.

# Внешний вид

В программе поменяется способ представления информации о проектах и версиях Visual Studio.

Вместо списка проектов будет таблица, большинство ячеек которой можно выбирать.

|  |  |  |  |
| --- | --- | --- | --- |
| **Solution** | **VS2005** | **VS2008** | **VS2010** |
| **Solution1.sln** | - | - | - |
| **Solution2.sln** | - | - | - |
| **Solution3.sln** | - | - | - |

В качестве имени проекта выступает short filename, так как у нас много версий Visual Studio и надо экономить место.

После того, как пользователь выбрал проекты и версии VS, он нажимает кнопку СТАРТ. Таблица меняет вид. Например, если пользователь выбрал только VS2005 и VS2008, то таблица примет вид:

|  |  |  |  |
| --- | --- | --- | --- |
| **Solution** | **VS2005** | **VS2008** | **VS2010** |
| **Solution1.sln** | Pending | Pending | - |
| **Solution2.sln** | Pending | Pending | - |
| **Solution3.sln** | Pending | Pending | - |

После проверки Solution1 и Solution2 в VS2005, таблица примет вид:

|  |  |  |  |
| --- | --- | --- | --- |
| **Solution** | **VS2005** | **VS2008** | **VS2010** |
| **Solution1.sln** | OK | Pending | - |
| **Solution2.sln** | Diff | Pending | - |
| **Solution3.sln** | Pending | Pending | - |

После того как все три проекта проверятся и в VS2005 и в VS2008:

|  |  |  |  |
| --- | --- | --- | --- |
| **Solution** | **VS2005** | **VS2008** | **VS2010** |
| **Solution1.sln** | OK | Diff | - |
| **Solution2.sln** | Diff | Diff | - |
| **Solution3.sln** | OK | OK | - |

Таблица обновляется после проверки каждого проекта, таким образом, являясь общим индикатором прогресса. Естественно, что и сравнение результатов анализа выполняется после проверки каждого проекта.

Сначала проекты проверяются в первой среде, затем во второй и т.п. То есть проверка делается вертикально по версиям VS. Это позволит понять еще при прогоне на VS2005, что есть ошибка.

Если кликнуть мышкой по одной из надписей Diff, то откроется проект и файл отличий именно для той версии Visual Studio, в какой колонке кликнули на Diff.

Не все проекты проверяются во всех версиях VS. Тогда ячейка становится неактивной и при ее выборе проверка не запускается.

|  |  |  |  |
| --- | --- | --- | --- |
| **Solution** | **VS2005** | **VS2008** | **VS2010** |
| **Solution5.sln** |  |  | OK |
| **Solution6.sln** |  |  | Diff |
| **Solution7.sln** |  |  | OK |

В качестве интерфейсных элементов кроме таблицы нужны еще три кнопки: запуск тестов, прерывание тестов и принять результаты тестов как эталоны. Эти кнопки для наглядности можно сделать рисунками (запуск – символ "воспроизведение, |>", остановка – символ "стоп, □", принять результаты – "зеленая галочка, V") с всплывающими подсказками.

**Обновлено:**

**Должна быть кнопка "Open html log", которая ведет себя так же, как в старой версии тестера.**

Кнопка "Сгенерировать эталоны" не нужна. Если логов нет, то надо выдать запрос на принятие нового лога как эталонного. Только не для каждого файла отдельно такой запрос.

**Обновлено:**

**Эталоны (проекты, для которых нет лог-файлов) подсвечиваем оранжевым перед approve.**

Кнопка "Принять результаты" работает с выделенными ячейками. Те ячейки, которые выделены, те и применяются. Важный момент – "принять результаты" можно только для первой версии Visual Studio. Т.е. при наличии VS2005 и VS2008 мы можем принять только от VS2005 результаты. Но если есть только запуск VS2010, то результаты принять можно и нужно.

Также еще надо в окне программы иметь часы, которые показывают, сколько времени прошло с момента запуска. То есть сначала они показывают 00:00:00, а потом при старте начинают идти и по завершении тестов, к примеру, показывают 01:25:32. Длительность работы тестов часто важный параметр, так как может показать, что в программе что-то стало работать очень медленно, хоть и корректные результаты сравнения.

Элемент переключения результатов запуска (Testing Run в текущей версии) очень удачное решение и его надо сохранить. Только combobox, без кнопок Import и Erase.

Какой-либо log-файл не сохраняем.

Консоль с результатами не нужна.

В главном окне можно выбрать – только PVS-Studio запускается, только CppCat или все вместе. Это реализуется с помощью checkbox'ов видимо (ну или посовременней контролы, но смысл тот же).

Очень важный момент – выбор проектов по умолчанию (при запуске программы). Выбираются всегда первые имеющиеся проекты (см. ниже). При этом выбран только запуск PVS-Studio.

Возможно для удобства выбора только первых проектов стоит таблицу делать чуть в другом формате и без заголовков:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| **Solution5.sln** | VS2005: OK | VS2008:Diff |  |
| **Solution6.sln** | VS2005: Diff | VS2008:Diff |  |
| **Solution7.sln** | VS2010:OK |  | - |

Тогда клик по первому столбцу выберет все проекты в "первой доступной" версии студии. Но этот вопрос – обсуждаемый.

**ОБНОВЛЕНО:**

**Решили сделать так. Первый столбец – VS2010. Соответственно клик по столбцу выбирает основной набор. По умолчанию при запуске уже выбирается первый столбец (VS2010). Мотивация такая. Наш основной прогон должен состоять из запуска VS2010, а не VS2005, которой никто не пользуется. Это конечно повлияет на время работы. Сейчас базовые тесты VS2005 проходят за 40 минут, базовые VS2010 – за 50 минут, дополнительные VS2010 – 60 минут. Соответственно теперь основной прогон будет занимать около двух часов (базовые VS2010 и дополнительные VS2010), но это необходимо. По умолчанию выбирается PVS-Studio.**

Программа должна поддерживать запуск из командной строки. Но пока это можно не уточнять, так как еще не понятно, что конкретно мы будем проверять при запуске из командной строки и в каком виде.

Конфигурационный файл программы будет использоваться также утилитой CLMonitoring Tester, но при этом его можно менять, это не проблема.

# Этапы работ

1. Разработка технического задания.
2. Создание макета пользовательского интерфейса.
3. Разработка болванки программы без внутренней логики, но с работоспособным интерфейсом. Все кнопки нажимаются, тесты выполняются, прогресс идет (с эмуляцией запуска).
4. Реализация внутренней логики программы; подготовка базы тестов на основе существующего набора.
5. Разработка инсталлятора.
6. Тестирование.
7. Внедрение вместо существующей системы Viva64SelfTester.