ПРОБЛЕМАТИКА ТЕСТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ ВСТРАИВАЕМЫХ СИСТЕМ

Юнусов Э.Э.
Научный руководитель – д.ф.-м.н.,проф. Грицунов А.В.
Харьковский национальный университет радиоэлектроники
(61166, Харьков, просп. Науки,14,каф. МЭПУ, тел.(057)702-13-62)
e-mail: yunusov.elman@gmail.com

Some issues of software testing in embedded systems are discussed. The problems of porting tests to the PC's hardware and the advantages giving by thisareshown. An approach to writing tests with the Gherkin language is described.

В настоящее время при разработке компьютерного ПО используются различные модели, такие как Waterfall, V-образная модель, спираль и другие [1]. Все они предполагают тесную связь между разработкой и одновременным тестированием. Также для автоматизации тестирования существуют различные сервисы, такие как Jenkins или TeamCity. Данные сервисы позволяют производить автоматический анализ, проверку и тестирование ПО.

Однако при тестировании ПО для встраиваемых систем существует ряд проблем, о которых пойдет речь далее.

ПО во встраиваемых системах зависит от характеристик контроллера, что означает, что и тесты имеют такую же зависимость. Также тесты на контроллерах исполняются в реальном времени, сократить которое нельзя.

Если при разработке ПО учесть уровень, абстрагирующий программный уровень от контроллера, то большую часть тестов можно перенести на мощности ПК. Тогда, при портировании тестов на ПК, можно использовать инструментарий доступный для тестирования.

Написание юнит и интеграционных тестов на ПК позволяет опустить зависимость программы от контроллера, а также значительно уменьшает время прохождения тестов.

Отдельный юнит-тест должен тестировать минимально возможную часть кода, также он не должен иметь ветвлений и циклов. В идеале тест должен содержать какое-то количество входных данных, их обработку и данные на выходе.

Для описания сценария теста подходит язык Gherkin. Он имеет следующую структуру: Given (в этой части идет описание исходных параметров теста), When (в этой части запускается тестируемая функция/метод), Then (в этой части проверяются возвращаемые функцией/методом значения), также в сценарии могут присутствовать подчасти And, в которых указываются дополнительные сведения. Такой подход к написанию сценариев позволяет качественно оценить сами тесты,

т.к. если тест не соответствует данной структуре, то его будет сложнее править при изменении функционала, покрытого этим тестом. При неясном объяснении каждой части теста или ее отсутствии появляются сложности в понимании цели тестирования.

Изначально разработчику, не знакомому с тестированием и методом разработки TDD (Test Drive Development), тесты без циклов и ветвлений могут показаться излишне неоптимизированными, однако все это направлено на простоту тестов, что упрощает процесс дальнейшей поддержки их актуальности.

В заключение можно отметить, что портирование тестов имеет большое преимущество: таким образом можно тестировать ПО юнит-тестами и отчасти интеграционными тестами, что является большей частью всех тестов, исходя из пирамиды тестирования Майкла Коэна (рис. 1).

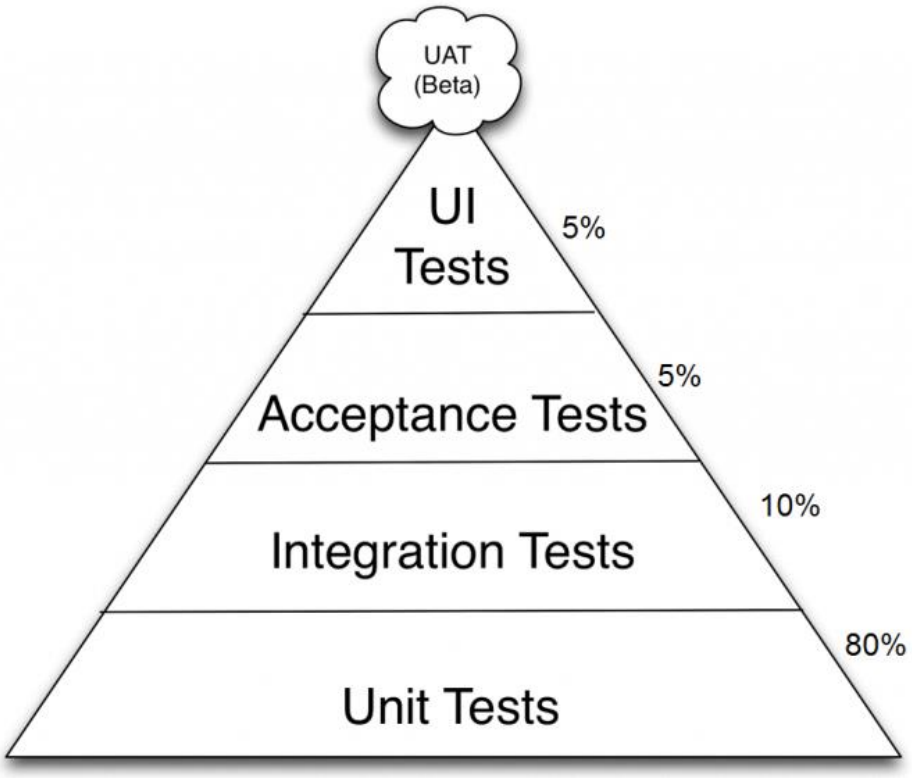

Рис. 1. Пирамида тестирования Майкла Коэна

Литература

1. Эрик Дж. Брауде. Технология разработкипрограммного обеспечения. – СПб.: Питер, 2004. – 655 с.