Chapter VIII

Rapid Pattern-Oriented Scenario-Based Testing for Embedded Systems

Wei-Tek Tsai, Arizona State University, USA
Ray Paul, Department of Defense, USA
Lian Yu, Arizona State University, USA
Xiao Wei, Arizona State University, USA

Abstract

Systems change often, and each change requires reverification and revalidation. Modern software development processes such as agile process even welcome and accommodate frequent software changes. Traditionally, software reverification and revalidation are handled by regression testing. This chapter presents a pattern-oriented scenario-based approach to rapidly revert and revalidate frequently changed software. Key features of this approach are (1) classifying system scenarios into reusable patterns; (2) application of a formal completeness analysis to identify missing scenarios; (3) identifying scenario patterns (SPs) and corresponding verification patterns (VPs) and robustness patterns (RBPs); (4) rapid test script generation by reusing test script templates. This approach is also compatible with formal approaches such as model checking. This approach has been used at industrial sites to test safety-critical medical devices with significant savings in cost and effort. The chapter presents several examples to illustrate the effectiveness and efficiency of this approach.
Introduction

Software systems are subject to changes throughout development, maintenance, and evolution, due to a variety of reasons such as requirement changes (both functional and nonfunctional), technology changes, and hardware or software platform changes (Eick, 2001; Hartmann, 1988; Lehman, 1985; Sugden, 1996; Tsai, 2000; and Wang, 1996). These changes bring risks to the software system because change propagations may introduce new bugs, sometimes even fatal ones. When the software is updated, it is necessary to reverify and revalidate the software with respect to the changed features as well as those features that are supposed to remain unchanged.

Regression testing is often used to reverify and revalidate software systems, and it has been immensely useful (Onoma, 1998). However, it is limited as follows:

- Regression testing reuses test cases, but existing test cases may not be applicable to verify new features unless the existing test cases are modified.
- Regression testing is most useful when changes are relatively minor, in other words, there are no structural changes in the program. If the concerned program’s structure is drastically changed, many of the existing test cases must be updated before they can be reused for regression testing.

Thus, regression testing may not be suitable for agile software development such as Extreme Programming where the concerned software is still under constant and frequent structural changes (Beck, 1999).

This chapter presents a new pattern-oriented scenario-based approach to rapid and adaptive testing software, even if the software is under constant structural changes as in an agile software development. This approach also complements regression testing; in other words, it can be used together with regression testing. This approach is called Verification Pattern (VP) (Zhu, 2002) and it has been experimented with at several industrial system development sites with significant results. This approach is especially suitable for testing safety-critical and mission-critical embedded systems.

The key concept of VP is that system scenarios can be classified into patterns, and for each scenario pattern (SP), a test script template can be developed for that pattern to test all the scenarios belonging to the same pattern. In other words, test script templates can be reused to verify a large number of system scenarios with minimum incremental cost and effort. Furthermore, if the system is changed, one can use a three-step process to verify the changed system:

1. identify new and/or changed scenarios in the system;
2. for each new scenario, see if it can be classified into the existing SPs. If yes, the existing test script template can be reused to test the new scenario. If the new scenario cannot be matched into the existing patterns, either a new pattern can be recognized and/or a new test script needs to be developed; and