The Journal of Systems and Software 88 (2014) 207–230

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

An approach to testing commercial embedded systems

Tingting Yu [a], Ahyoung Sung [b], Witawas Srisa-an [a], Gregg Rothermel [a,*]

[a] Department of Computer Science and Engineering, University of Nebraska-Lincoln, Lincoln, NE, United States
[b] Division of Visual Display, Samsung Electronics Co., Ltd., Suwon, Gyounggi, South Korea

ARTICLE INFO

Article history:
Received 12 January 2013
Received in revised form 16 August 2013
Accepted 19 October 2013
Available online 22 November 2013

Keywords:
Embedded systems
Software test adequacy criteria
Test oracles

ABSTRACT

A wide range of commercial consumer devices such as mobile phones and smart televisions rely on embedded systems software to provide their functionality. Testing is one of the most commonly used methods for validating this software, and improved testing approaches could increase these devices' dependability. In this article we present an approach for performing such testing. Our approach is composed of two techniques. The first technique involves the selection of test data; it utilizes test adequacy criteria that rely on dataflow analysis to distinguish points of interaction between specific layers in embedded systems and between individual software components within those layers, while also tracking interactions between tasks. The second technique involves the observation of failures: it utilizes a family of test oracles that rely on instrumentation to record various aspects of a system's execution behavior, and compare observed behavior to certain intended system properties that can be derived through program analysis. Empirical studies of our approach show that our adequacy criteria can be effective at guiding the creation of test cases that detect faults, and our oracles can help expose faults that cannot easily be found using typical output-based oracles. Moreover, the use of our criteria accentuates the fault-detection effectiveness of our oracles.

1. Introduction

A wide range of commercial consumer devices rely on embedded systems software to provide required functionality. Examples of these devices include mobile phones, digital cameras, smart televisions, gaming platforms, and personal digital assistants. Systems such as these are being produced at an unprecedented rate, with over four billion units shipped in 2006 (LinuxDevices.com, 2008).

In contrast to the software embedded in safety-critical devices such as automobiles, airplanes, and medical devices, which must satisfy hard real-time constraints, the embedded systems software in commercial consumer devices such as those just listed has fewer such constraints. In particular, in commercial embedded systems, timing issues are less critical: a few misses of deadlines are typically considered tolerable (Liu, 2000) and meeting all deadlines is not even the primary consideration. Instead, the faults most common in these systems involve other issues that impact functionality. For instance, one taxonomy names buffer overflow, memory leaks, improperly heeded compiler warnings, race conditions, and memory alignment traps as the "five most destructive bugs" found in such systems (Shalom, 2010). Blackberries and Android phones, for example, are known for being rife with memory leak faults (Forums, 2011, 2006), security faults (Stanley, 2011), and various faults related to functional behavior (CrackBerry, 2008).

There has been a great deal of research performed on techniques for validating embedded systems generally. Much of this work (Section 7 provides details) involves non-testing based approaches. For example, static analysis has been used to enhance the overall quality of software while reducing run-time costs (e.g., Ball et al., 2006; Engler et al., 2000). Formal verification techniques have also been used (e.g., Behrmann et al., 2004; Bodeveix et al., 2005) in validation tasks.

While non-testing-based approaches can provide levels of assurance that are necessary for safety-critical systems, they can also be expensive to apply. Thus, in industrial settings, and in particular where the embedded systems underlying commercial devices are involved, testing-based validation approaches take on great importance. For example, our co-author in the Division of Visual Display at Samsung Electronics Co., Ltd. confirms that testing is the primary methodology used in their division to verify that the televisions, blu-ray disk players, monitors, and other display-devices created by Samsung fulfill their functional requirements. Section 2.2 provides further details on this process, but in brief, at Samsung the most rigorous testing activities are applied after software components have been integrated and are being dynamically exercised on simulators and target boards. Here, system tests are based primarily on expected functional behaviors as inferred from non-formal requirements specifications. Finding approaches by which to better focus this testing on interactions that occur

* Corresponding author. Tel.: +1 4023282796.
E-mail addresses: tyu@cse.unl.edu (T. Yu), ahyoung.sung@samsung.com (A. Sung), witty@cse.unl.edu (W. Srisa-an), grother@cse.unl.edu (G. Rothermel).

0164-1212/$ – see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jss.2013.10.041

among software components and that can be inferred automatically from the code is a high priority.

There has been some research on techniques for testing embedded systems. Some efforts focus on testing applications without specifically targeting underlying software components and without observing specific underlying behaviors (e.g., Tsai et al., 2005), while others focus on testing specific major software components (e.g., real-time kernels and device drivers) that underlie the applications (e.g., Arlat et al., 2002; Barbosa et al., 2007; Koopman et al., 1997; Tsoukarellas et al., 1995). Certainly, applications must be tested, and underlying software components must be tested, but in contexts such as those present at Samsung, methodologies designed to capture component interactions, data-access rules, dependence structures, and designers' intended behaviors are needed. Approaches presented to date do not adequately do this.

In this article, we present an approach meant to help system testers of the embedded systems that lie at the core of commercial consumer devices perform testing more effectively. Our approach is composed of two techniques, each of which focuses on an aspect of the testing task: (1) helping engineers create adequate test cases, and (2) improving the observability of failures that result from testing. Our first technique involves test adequacy criteria that test embedded systems software from the context of applications, focusing on faults that are not related to timing, and targeting interactions between system layers and tasks; this is the first work to address embedded system dependability in this context. Our second technique involves the use of test oracles that utilize instrumentation to record various aspects of embedded system execution behavior and compare observed behavior to certain intended system properties that can be derived through program analysis.

We report results of empirical studies focusing on three embedded system applications built on a commercial embedded system kernel, that show that our adequacy criteria can be effective at guiding the creation of test cases that can detect faults, and our oracles can help expose faults that cannot easily be found using typical output-based oracles. Even more important, our results show that our adequacy criterion and oracles are appropriate partners: the use of our criteria accentuates the fault-detection effectiveness of the oracles.

The remainder of this article is organized as follows. Section 2 provides background information. Section 3 describes our technique for helping testers create adequate test suites and Section 4 presents our empirical study of that technique. Section 5 describes our technique for providing test oracles and Section 6 presents our empirical study of those oracles. Section 7 describes related work, and Section 8 presents conclusions.

2. Background and motivation

We begin with an overview of architecture of the embedded systems found in commercial consumer devices. Then we describe several characteristics and properties of these systems and the processes by which they are developed and tested, that motivate our approach.

2.1. Commercial embedded system architecture

The embedded systems that underlie commercial consumer devices are designed to perform specific tasks in particular computational environments consisting of software and hardware components. Fig. 1 illustrates the typical structure of such a system in terms of five layers – the first four consisting of software and the fifth of hardware.

An application layer consists of code for specific applications. Two types of applications may be supported: built-in and external. Built-in applications are written by the engineers who create the devices, while external applications (when supported) can be created by anyone (e.g., applications created in Android SDK.) Both types of applications utilize services from underlying layers including Middleware, the Operating System (OS) and the Hardware Abstraction Layer (HAL). The gray boxes represent interfaces between these layers; these provide access to, and observation points into, the internal workings of each layer.

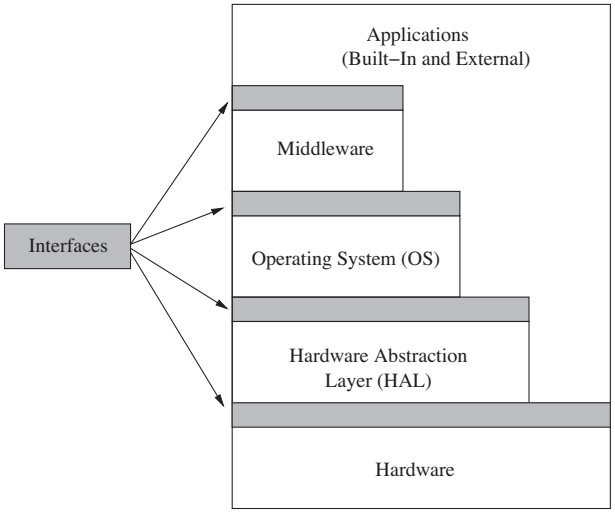

Fig. 1. Architecture of a commercial embedded system.

The middleware layer consists of various runtime components (e.g., graphic engine, font engine, multimedia file parser, and voice recognition engine). Middleware is designed to support the common functions for built-in or external applications. As an example, any application in a mobile phone requires support from a graphic engine in the form of drawing APIs that render icons and characters on the screen. Similarly, a voice recognition engine provides a set of APIs that developers can use to include voice command capability in their applications.

The Operating System (OS) layer consists of task management, context-switching, interrupt handling, inter-task communication, and memory management facilities (Labrosse, 2002) that allow developers to create commercial embedded system applications that meet functional requirements and deadlines using provided libraries and APIs. The kernel is designed to support applications in devices with limited memory; thus, the kernel footprint must be small. Furthermore, to support real-time applications, the execution time of most kernel functions must be *deterministic* (Labrosse, 2002). Thus, many kernels schedule tasks based on priority assignment.

The hardware abstraction layer is a runtime system that manages device drivers and provides interfaces by which higher level software components can obtain services. A typical HAL implementation includes standard libraries and vendor specific interfaces. With HAL, applications are more compact and portable and can easily be written to directly execute on hardware without operating system support.

2.2. Characteristics of commercial embedded systems and their development processes

The embedded systems underlying commercial consumer devices, and the processes by which they are developed, have certain characteristics that must be considered when creating testing techniques that target them.