IJCSC 0973-7391 Volume 7 • Number 1 Sept 2015 - March 2016 pp. 270-272 available online at www.csjournalss.com

Built-In Testing For Component Based Software Development in Embedded Systems

Divya Shree

Abstract: Component-based software development (CBSD) has made it easier to develop an embedded system. Many approaches have been proposed to create embedded systems through components. While creating a component-based embedded software system it is difficult to test the functionality of the component. This paper proposes a method to enhance testability of an embedded component using PECOS model approach of component-based software development. Built-in test cases have been incorporated into the component and a case study is shown using implementation in java. The method makes the task of integration testing of embedded components easier.

Keywords: Component-based Software Engineering, Embedded System, Field Devices, Built-in, COTS, Component model.

1. Introduction

During the past decade software engineering community has learned to design software for embedded systems using Component-based technology. Software forthe embedded systems isplatform dependent. It is difficult to maintain, upgrade and port to other platforms but development using component-based technology has made it easier and faster. The components trustworthiness is an issue of vital importance. The component being developed for an embedded software system must be compatible with the main system to which it will be ported.Various approaches have been adopted to build a component-based embedded software system.

The commonly used approaches or the component models arepecos, saveCCM, procom, comdes II etc.

Testing of the embedded software components is a difficult as well as a tedious task because of many test-cases involved for checking component compatibility and behavior. So built-in test cases can be introduced into the components to check the compatibility and also make component behavior adjustments if possible.

The organization of rest of the paper is as follows: firstly in section 2 we discuss the need for built-in test-cases in the embedded software system and the principles of an embedded system along with the built-in. Section 3 exposes the new approach that we have proposed considering the example application which we developed to our view point clear. The section 4 includes the implementation work and at last in section 5 the paper closes with the final results.

2. Need for Built-in in Embedded Software System

Embedded software system's components are expected to be of high quality, but in certain cases due to lack of relevant information (inaccurate application environment, insufficient documentation of the component) provided by the developer can mislead the component user [1]. Hence testing the components has become a major concern while developing complete embedded software.The CBSE approach is inefficient in testing and verification of the components because the black-box testing approach used here is concerned only with the specifications and not the design or code of the component [4]. Also generating test-cases for each component separately becomes a difficult task. Hence to overcome the problem of component testability and verification problems BIT (Built-in tests) are taken into consideration.

2.1. Embedded Software System

Embedded software systems are microprocessor based systems that are used in large range of computer systems for monitoring and controlling complex processes. Such systems consist of hardware and software integrations in which the software reacts to the environment.An embedded component is developed such that it provides specific functionality to the system and testing the component integration into the embedded system is of vital importance.

2.2. Built-in-Testing (BIT) in Embedded Software System

With the increased use of CBSD approach for developing software system and disadvantage of the testing methodologies followed, Built-in-Testing (BIT)has been adopted as a component testing technique which not only improves the quality of the system but also helps in checking the compatibility of

the software components with each other. According to Binder [2] "Built-in-test refer to code added to an application that checks the application at runtime". BIT is a Design-for-Testability (DFT) technique; they contain test cases or possess facilities for generating the test cases into the component. The test cases are explicitly described in the software source code as member function forenhancing the software testability and maintainability [3].

3. New approach

In embedded system it is very necessary to check the compatibility of the components of an embedded software system with the main system in which it will be used in future. Not only reliability but also software quality is of vital importance which can be quantified in terms of testability and maintainability. Till date off-the-shelf (COTS) components had undergone unit testing by the manufacturer where no testability and maintainability at unit level is available to the system integrator. This level of trust cannot be afforded when dealing with the embedded or the safety critical applications. So we have developed a method to enhance the testability and monitoring of an embedded component considering a component model [5]. Various component models have been developed for embedded systems such asrubus, comdes II, pecos, saveCCM, procometc [8]. For our method we have used the pecos component model of embedded software development.

PECOS component model was originally developed for field device systems in which Component Composition (CoCo) description language was used for specifying the components [6]. The implementation for this model can be in any of the object-oriented languages (c, c ++, java etc). Our method makes use of java language for the implementation of the component as well as the BIT.

The method provides an idea of better testing and check the compatibility of the components of an embedded software system by incorporating the test cases into the source code of the components at the development stage. The built-in-tests make the software testing self-contained [7]. The method proposed includes case study in which component have been developed using java language. The test-cases have been built inside the source code itself in the form of a function.

3.1. Example application

The application that we have developed is based on Pecos component model. The model works basically for field devices. A field device is an embedded system which uses sensors to continuously gather data, such as temperature, pressure or rate of flow and then react to the environment by controlling the actuators [5]. The application generated by us consists of a component or the class (tempcontrol2()) coded in java language. The basic purpose of this component is to switch its task by behaving as an air-conditioner or a heater according to the temperature taken as an input from the user. The component developed is such that it represents an important characteristic of a high-quality software component i.e. Reusability. Reusability in the sense that the single component developed performs the task of both an air-conditioner and the heater. The functionality of the component remains the same i.e. reading the temperature as an input from user and then reacting to the environment accordingly which in turn shows the reusability characteristic. Hence it solves the purpose of being an embedded component with a specific functionality with reusability considered.

Component-based usability testing is a major concern when software components directly interact with the users while developing the component. This has been done by incorporating the test-case (testcases()) as a member function into the component. The component itself is a class that constitutes various data members and the member functions. In our application the class named "tempcontrol2()" consists of a member function-"control()" which provides the conditions on when to switch between the two tasks according to the temperature change. Another member function-"testcases()" have been coded which is the built-in test-case to check whether the embedded component is compatible with the external environment or not. The test-case checks for different conditions and the action that need to take place likewise.

4. Implementation Code

The code for the testcases() is as follows:

```
Publicvoidtestcases()
{
Scanner sc = newScanner(System.in);
Scanner sc1 = newScanner(System.in);
System.out.println("enter the high temperature");
booleanhh = sc.hasNext();
System.out.println("enter the room temperature");
floatrr=sc1.nextFloat();
System.out.println("enter the user temperature");
floatuu = sc1.nextFloat();
hightemp = hh;
usertemp=uu;
roomtemp=rr;
control();

if((hh==true)&&(uu>rr))
{
if(x=='3')
System.out.println("verified");
else
System.out.println("error");
}
```

```
elseif((hh==true)&&(uu<=rr))
{
if(x=='0')
System.out.println("verified");
else
System.out.println("error");
}
elseif((hh==false)&&(uu<rr))
{
if(x=='2')
System.out.println("verified");
else
System.out.println("error");
}
else
{
if(x=='0')
System.out.println("verified");
else
System.out.println("error");
}
```

}Will have to put the code in a box after the complete text is written

The above component along with the built-in test-case helps in testability of the compatibility of the embedded software system with the environment or the main system in which it has to be plugged in.

5. Conclusions

The method proposed here can simplify the method testing the compatibility of an embedded software system (component) with the external environment into which it has to be plugged in. the test-cases incorporated into the component source code are check the behavior of the embedded software system. These test-cases are behavioral in nature that predicts the behavior of the component [2]. Hence no additional resources are required for the test case generation. This approach makes it easier for the system developer to integrate the components with each other with no harm to functionality of each component as well as the compatibility of such system with the main system is given importance.

6. References

[1] Sami Beydeda, Bundesamt fur Finanzen; Research in Testing COTS Components – Built-in Testing Approaches; IEEE; 2005.

[2] Franck Barbier, Nicolas Belloir; Component Behavior Prediction and Monitoring through Built-In Test; IEEE; 2003.

[3] Yingxu Wang, Graham King, HakanWickburg; A Method for Built-in Tests in Component-based Software Maintenance; Research Centre for Systems Engineering, Southampton Institute.

[4] Irena Pavlova, AleksandarDimov; Advanced Approach foe Effective Verification of Component Based Software Systems; 2007.

[5] Michael Winter, Oscar Nierstrasz, Peter Muller; Components for Embedded Software – The PECOS Approach.

[6] DayandNorhayatiAbangJawawi, SafaaiDeris, RosbiMamat; A Component-Oriented Programming Framework for Developing Embedded Mobile Robot Software using PECOS Model.

[7] Eric Piel et al; On Coping with real-time software dynamic inconsistency by built-in-tests.

[8] Kung-Kiu Lau, Zheng Wang; A Taxonomy of Software Component Models; IEEE; 2005.