Self-Management of External Device Failures in Embedded Software Systems

Michael E. Shin
Department of Computer Science
Texas Tech University
Box 43104, Lubbock, TX 79409
1-806-742-3527
Michael.Shin@ttu.edu

Poonam Mane
Department of Computer Science
Texas Tech University
Box 43104, Lubbock, TX 79409
1-806-742-3527
Poonam.Mane@gmail.edu

Abstract

Embedded software systems interact with several external devices that may come across faults or failures. Most of the approaches have focused on software and hardware systems, but relatively less attention has been given to self-management of faults or failures of external devices in embedded software systems. It is necessary to develop an approach such that faults or failures of external devices in embedded software systems are detected and self-managed.

This paper describes an approach to establishing a framework for detecting and self-managing the failures of external devices for embedded software systems. Failed external devices are identified based on external device types, being self-managed in different ways. The proposed approach in this paper is applied to the use case modeling of an elevator system case study.

1. Introduction

Embedded software systems need to be more reliable against faults or unanticipated events so that the systems become more resilient to the system failures. Embedded software systems may still contain software faults in terms of specifications, semantics, logics, resources, and synchronizations even though the systems are verified and validated during development. Embedded software systems are running on hardware, which may fail due to hardware failures like aging, on-board circuit issues, peripheral or accessory maintenance. Previous approaches [Cheng06, IBM06, Roy07] have focused on failures in software and computer hardware [Coyle04, Pinheiro07, Wikipedia10].

However, not much attention has been given to faults or failures of external devices in embedded software systems, such as elevator systems and robotic systems. An elevator system should control several external devices such as direction lamps, elevator buttons, door, and motor. These external devices can run into some faults or malfunctions, which may lead the systems to failures. The software for embedded systems may not work correctly even though these external devices have minor faults or malfunctions.

This paper describes an approach to establishing a framework for detecting failures in the external devices of embedded software systems and for self-managing the detected failures. This paper develops the detection framework based on the external device types in embedded systems. There are unique communication patterns between embedded software systems and their external devices. These patterns are used to devise failure or fault detection mechanisms. Further, the self-managing framework is developed for handling external device failures from the software perspectives. The self-management is carried out by means of self-healing, self-configuration or adaptation, and self-reporting. The proposed approach in this paper has been applied to the use case modeling of an elevator system.

This paper is organized as follows. Section 2 describes classification of external devices. Section 3 describes the devised framework for fault detection, followed by self-management of external device failures in section 4. Section 5 describes the use case modeling for detecting and self-managing failed external devices. This paper is concluded in section 6.

2. Classification of External Devices

There are different external devices in software systems, which can be classified into several types [Gomaa00]. A type of external devices has the unique communication pattern between the devices and the system. This uniqueness of communication styles is used to develop a framework for detecting faults or failures in external devices. Each device type is categorized by means of device characteristics, such as input/output, periodic/non-periodic, asynchronous, or active/passive.

- Input/output devices: An input or output device can provide input to or receive output from the system respectively. Some devices have only one role of either input or output, or both input and output. For example, an elevator button is an input device and an elevator lamp is an output device in an elevator system.
- Periodic/non-periodic devices: A periodic device interacts with the system periodically, whereas a non-periodic device interacts with the system as needed. For example, a brake is a non-periodic device, and an engine is a periodic device in a cruise control system.
- Asynchronous devices: An asynchronous device generates an interrupt to send a message to the system, which then reads a message from the device. Examples of this type are elevator buttons in an elevator system and cruise control buttons in a cruise control system.
- Active/Passive devices: An active device cannot generate an interrupt to be sent to the system, but it may send directly a message to the system. A passive device cannot generate an interrupt for communication with the system and cannot send a message to the system. The motor and the elevator door in an elevator system are passive.

3. Detection for External Device Failures
3.1 Failure Detection of Asynchronous Input Devices

An asynchronous input device generates an interrupt and delivers it to the device interface whenever it meets some event interesting to the interface. When the interface receives an interrupt from an asynchronous input device, it may need to read a message from the device. An asynchronous input device is characterized as input, asynchronous, and non-periodic. An asynchronous device may not generate any interrupt if it fails. In that case, the interface cannot detect the failures of the asynchronous input device. Examples of this type of asynchronous external devices are floor buttons and elevator buttons in the elevator system.

However, there are some exceptional asynchronous input devices whose failures may be detected under some conditions. An arrival sensor in the elevator system is an example of the asynchronous input device whose failures are detected using other devices in the system. Figure 1 shows the communication between the arrival sensor and the elevator control. An arrival sensor is installed at each floor, sensing the arrival of the elevator at the floor. As an elevator reaches a specific floor, the arrival sensor sends an interrupt to the arrival sensor interface, which notifies the elevator controller of the message arrival. The elevator controller checks if the elevator should stop at the floor or not. When an elevator is going up, the i-th floor arrival sensor may be presumed that it fails if the elevator controller receives an arrival message from the (i+1)th arrival sensor without a message from i-th arrival sensor. Similarly, this can be applied when the elevator is going down.

A periodic output device receives output from the system periodically and handles the output. A periodic output device is passive, thus it just waits for an output from the system. Failures of a periodic output device may not be detected by the system. This is because the device may not respond to the system when an output is arrived. But failures of periodic output devices may be detected by users. Examples of such devices are a mileage display, a trip average display and a maintenance display in a cruise control system, and a microwave display in a microwave system (Fig. 3). These displays show some information to users in which the information is changing. Users can detect the failures of these displays when the information is not changing.

Some periodic output devices can be monitored by polling the devices. Such an example is the throttle in the cruise control system. The throttle is classified as a periodic output device whose failures may not be detected. But a throttle is a polled device whose failure is detected when the throttle position sensor does not return any value to the system within a defined period of time.

3.2 Failure Detection of Periodic Input Device

A periodic input device delivers inputs to the system periodically, but it is not an active object. A periodic input device is characterized as input, periodic, and passive. A system polls a periodic input device to get an input from the device on a regular basis. When the system polls the device, the device is presumed that it fails if the system cannot read an input from the device upon each poll.

The engine in a cruise control system is the example of this type of device. The engine is polled by the engine interface to read an input at periodic intervals. When the engine does not return its input to the interface, it may meet a failure. Figure 2 shows the engine failure detection in the cruise control system. Also failures of an engine can be detected in another way. If the system does not receive the normal heat temperature values from the heat sensor for a certain period of time, the system detects the failure of engine.

3.3 Failure Detection of Periodic Output Device

A peripheral output device delivers output from the system periodically and handles the output. A peripheral output device is passive, thus it just waits for an output from the system. Failures of a peripheral output device may not be detected by the system. This is because the device may not respond to the system when an output is arrived. But failures of peripheral output devices may be detected by users. Examples of such devices are a mileage display, a trip average display and a maintenance display in a cruise control system, and a microwave display in a microwave system (Fig. 3). These displays show some information to users in which the information is changing. Users can detect the failures of these displays when the information is not changing.

Some periodic output devices can be monitored by polling the devices. Such an example is the throttle in the cruise control system. The throttle is classified as a periodic output device whose failures may not be detected. But a throttle is a polled device whose failure is detected when the throttle position sensor does not return any value to the system within a defined period of time.

3.4 Failure Detection of Passive Output Device

The system sends an output to a passive output device as needed, which handles the output. A passive output device is checked for the status by the system after an output is sent to the device. A passive output device is characterized as output, passive, and non-periodic. The motor, elevator lamp, and elevator door in the elevator system are examples of passive output devices.

The status of the motor (Fig. 4) in the elevator system is determined by checking the rotation/motion sensor after an output, such as motor start or stop, is being delivered to the motor. When the rotation/motion sensor does not return a valid response to the elevator system within a defined period of time, it leads to a conclusion that the motor has failed. Similarly, the elevator direction lamp is monitored by the voltage sensor, and the elevator door’s failures are detected by the rotation sensor status.
4. Self-management of External Device Failures

4.1 Self-healing

Failures detected in external devices may be resolved by retrying the communication with the devices. Similar to software, an external device may meet a failure attributed to some minor mechanical error or instant error. For failures of a periodic input device, the system reads an input from the devices again. For instance, an engine interface retries to read an input from an engine if there is no response from the engine. A passive output device interface retries to send an output to the device, and polls the status of the device again. For example, the motor in the elevator system may be retried if it meets some failures.

Failed external devices can be reinitialized by calling the initialize() operation, which initializes the devices and its related variables. A device interface object has the initialize() operation that is called at initialization time. However, the system calls the initialize() operation if it meets failures in the devices, and the system retries to communicate with the failed devices.

The system controller switches failed external device to another device if a secondary device is available. The secondary device has the same functionality as the primary device. The switch between a primary device and a secondary device is carried by the device interface, but the system does not require reconfiguration of software components constituting the system. An external device may not have its secondary device if the device is not critical to the system or if the secondary device is not cost-effective.

The self-healing approach can be applied to the external devices that are non-time critical. A failure of external devices can be categorized with response time, such as time critical or non-time critical. A time critical device’s failure is intolerant if the device does not work on time. A time critical device does not have enough time to self-repair the failures. A failure of time critical device may shut down the entire system. The brake or engine in a cruise control system is an example of the time critical device. A non-time critical device is not a time bound device whose failure may be fixed by self-repair. A floor button in an elevator system is an example of non-time critical device.

The self-healing approach for external devices has some limitations that may not fix failures attributed to real mechanical problems. An external device with a mechanical problem needs to be replaced physically by hardware engineers if the secondary device is not installed with the primary device. For example, an elevator direction lamp may be burn out due to the life time, and the motor of an elevator may meet a mechanical problem that cannot be fixed by retrying or initializing it.

4.2 Self-configuration or adaptation

Failure analysis enables us to analyze the impact and the consequences of the failed device in the embedded system. The failures can be classified as tolerable, serious and catastrophic. A failure of external devices that does not disrupt the normal functioning of the system is termed as a tolerable failure. An example is a failure of a direction lamp of an elevator. Even though a direction lamp fails, the elevator works fully without degrading its services to passengers. A serious failure in the external devices degrades the functions of a running system, but it may not necessarily bring the system to a total failure. For example, an elevator button’s failure makes some passengers not reach to a specific floor, but the elevator still works. Another example is a failure of arrival sensor at a specific floor in an elevator. A catastrophic failure is irrecoverable, and it brings the entire system to a sudden halt. The examples are failures of the motor in an elevator system, and failures of the brake in a cruise control system.

The embedded software may need to be reconfigured or adapted against tolerable or serious failures of external devices so that the system prevents from the ripple effect in terms of the failures – worse performance or additional failures. When a device meets either a tolerable or serious failure that cannot be self-repaired, the system should not communicate with the device in order to send an output to or read some input from the device. This communication can make the device interface meet the same failure, and try to self-repair it using retry or re-initialization. Even the same failure may be notified to the system controller, which already knows the device status. For example, an elevator system should not send an “on or off” message to an elevator direction lamp if this lamp is already detected to be failed. Otherwise, the elevator direction lamp will try to self-repair the same failure again and the system’s performance will be getting worse.

A failure of external device may bring another failure to the system if the system is not adapted against the failed device. Suppose that multiple elevators are operating in a building and an arrival sensor for a specific floor of an elevator fails. The scheduler of the elevator system should not patch the elevator to the floor with a failed arrival sensor. The elevator cannot stop at the floor due to the failure of arrival sensor. If the elevator is patched to the floor, the elevator may continue to go up or down repeatedly to reach the floor.

4.3 Self-reporting

The embedded software systems can be divided as unmanned control or manned control systems. The elevator system is an example of unmanned control system, whereas the microwave and the cruise control systems are manned control systems. This criteria is made based on to whom the non-self managed failures should be reported. In the case of the unmanned control systems, the maintenance centers are notified while the users are notified in the manned control systems. When a device failure is detected, the failure can be either detected by the device interfaces, or users. If a catastrophic device in the unmanned control system fails and there is no way to self-manage, the system should stop and report to appropriate person or maintenance center.

5. Detection and Self-management in Use case Modeling

The proposed approach for detection and self-management of failed external devices is applied to the use case modeling for the elevator system, cruise control system and microwave system [Mane10]. The following is the Stop Elevator at Floor use case [Gomaa00] in the elevator system, which is extended to
detect and self-manage failed external devices – arrival sensor, motor, floor direction lamp, and elevator door.

Name: Stop Elevator at Floor Use Case

Actors: Arrival sensor

Precondition: Elevator is moving.

Description:
As the elevator moves between floors, the arrival sensor detects that the elevator is approaching a floor and notifies the system. The system checks whether the elevator should stop at this floor. If so, the system commands the motor to stop and turns on the floor direction lamp. When the elevator has stopped, the system commands the elevator door to open.

Alternative for Application:
• The elevator is not required to stop at this floor and so continues past the floor.

Postcondition: Elevator has stopped at floor with door open.

Detection and Self-management of External Device Failures:
• Arrival sensor (asynchronous input device). Detection - the system may not detect the failure, but it may detect it using neighbor sensor input.

Self-management - if an arrival sensor fails and the elevator needs to stop at the floor, the system stops the elevator at the next (up or down) floor to drop passengers. The system self-reports this failure to the maintenance center if the arrival sensor does not send the signal to the system continuously.

• Motor (passive output device). Detection - if the motor does not return the status within a defined time or return an abnormal door sensor value, the system detects the failure of motor. Self-management – the system reinitializes the motor and commands it to start moving the elevator. If the motor is still not moving the elevator and the elevator is located at a floor, the system commands the elevator door to open so that passengers can go out of the elevator. The motor is a critical device, so the system self-reports this failure to the maintenance center. The system does not allocate this failed elevator to user requests.

• Floor direction lamp and Elevator door (passive output devices). Detection - if the floor direction lamp (elevator door) does not return the normal value, the system detects the failure of the floor direction lamp. Self-management – the system reinitializes the floor direction lamp (elevator door) and goes to the next use case description step because it is not a critical device. Since it is a non-critical, tolerable failure, the system reconfigures the failed floor direction (elevator door) lamp interface so that the system does not send messages to the failed device. The system self-reports this failure to the maintenance center.

The use cases for the elevator system with detection and self-management of failed external devices are implemented to validate the proposed detection and self-managing framework. This implementation is to develop a simulator for the elevator system in which the external devices are implemented as software objects rather real devices. The software architecture for self-managing external devices of the elevator system is designed as a layered architecture structured with the service layer and self-management layers [Mane10]. The service layer has application objects for the elevator system, whereas the self-management layer has objects – detection, planning, and execution modules - related to detection and self-management of device failures.

6. Conclusions

This paper has described an approach to establishing a framework for detection and self-management of failed external devices in embedded software systems. This paper has developed the detection framework based on the external device types in embedded software systems. The communication patterns between external devices and the system are used to detect failure or fault of external devices. The self-managing framework handles external device failures from the software perspectives. The proposed approach in this paper has been applied to the use case modeling of the elevator system.

This paper can have further research. This research has been applied to three embedded software systems - elevator system, cruise control system and microwave system. This research may need to other embedded software systems so that we can enable to identify more device types. Also this paper does not include the relationship between an external device and its sensor. An external device may have its sensor that is used to monitor the device. When an external device fails, the failure may be in the device or its sensor. This relationship needs to be addressed in our future research.

References


