An Approach for Using Mobile Devices in Industrial Safety-Critical Embedded Systems

Ashraf Armoush[1], Dominik Franke[2], Igor Kalkov[2], and Stefan Kowalewski[2]

[1] An-Najah National University
armoush@najah.edu
[2] Embedded Software Laboratory, RWTH Aachen University
Ahornstr. 55, D52074 Aachen, Germany
{franke,kalkov,kowalewski}@embedded.rwth-aachen.de

Abstract. With the booming mobile market and increasing capability of mobile devices, mobile platforms like Android emerge from end-user to industrial application areas. This paper sketches an approach to implement industrial safety-critical embedded systems with fail-safe state on the mobile platform Android. The approach consists of safety-critical software design patterns, a real-time extension to the Android platform and fail-safe application life cycle management.

Keywords: safety-critical, fail-safe, patterns, real-time, life cycle, Android.

1 Introduction

Safety-critical systems are those systems or applications in which failure can lead to serious injury, loss of life, significant property damage, or damage to the environment [3]. The industrial safety-critical embedded systems should be carefully designed with the help of the well-established and common design techniques to fulfill the real-time and safety requirements.

While smartphones and tablets are classified as embedded devices, they serve as the key computing and communication choice for other embedded devices. Modern smartphones contain a wide variety of networking technologies and a set of powerful embedded sensors, such as GPS, accelerometer, gyroscope, compass, and camera. The availability of these capabilities and technologies has opened new applications across a wide variety of domains, and allow them to become an attractive and complementary choice for safety-critical embedded systems. However, their operating systems must be modified and extended to support the reliability and real-time requirements for safety-critical systems.

Fail-safe state in safety-critical system is a state which, in the event of failure, can be identified as being safe without risk. In many design methods, a safety function should be executed in the case of failure to reach a fail-safe state. For smartphones, the application life cycle should be carefully studied and investigated to find the best way to integrate the required safety function.

G. Memmi and U. Blanke (Eds.): MobiCASE 2013, LNICST 130, pp. 294–297, 2014.

2 Applying Safety Patterns on Mobile Platforms

The design of safety-critical embedded systems is considered to be a complex process to fulfill two types of requirements: functional and non-functional requirements. The first part includes the function to be performed, while the non-functional requirements involve the attributes of a system as it performs its job. They include a set of requirements like high reliability and safety, availability, low cost and small size. While the influence of these requirements varies from one domain to another, all safety-critical systems must guarantee an acceptable level of safety. The concept of design patterns, which introduces an abstract representation for how to solve general design problems have been widely used in hardware and software. In the field of safety-critical embedded systems, a collection of common design methods has been presented in a previous work as a catalogue of safety pattern [1].

2.1 Mobile Application Life Cycle and Safety Function

Android schedules its applications and services on application level using states and state transitions. The states and state transitions are often called *life cycles*. Depending of the state an application or service is in, it has access to certain resources (RAM, CPU, ...). For instance, if an Android application is in the state *running* it has access to all resources of the device. If the application is shut down or killed, it has no access to any resources.

Safety-critical system with fail-safe state can be executed on Android as services or applications. In previous work on application and service life cycles, we analyzed which would be the best way to prevent a service from being shut down and how to make sure that an application executes specified actions before being shut down. If a system with fail-safe state is executed on Android as a service, the service should be bound to an Android application and display an icon in the Android notification bar. Both actions increase the priority of the service from, as it is displaying information (in the notification bar or in the bound application), which immediately contribute to the user experience [4].

If a system with fail-safe state is executed on Android as an application, the application life cycle callback methods should be used to trigger fail-safe actions in case of application state changes [5]. As on Android usually only one single application can be active at a time, applications often change their states. An application being responsible for the fail-safe state system can act and react best, if it is in the state *running*. If it changes its state, it might not be able to execute fail-safe actions any more. From our previous work on Android application life cycles, we know that during regular operation a running Android application always calls the life cycle callback method *onPause()*. This method should be used in safety-critical applications to trigger appropriate fail-safe actions.

2.2 Real-Time Requirements and Real-Time Extension

Most safety-critical embedded systems have real-time requirements, as they must react on specific conditions within predefined time intervals. The integration of

smartphones in safety-critical environments urges for modifications and extensions of available platforms to support applications with real-time requirements.

RTAndroid is a modified version of the Android 2.2 platform extended with a real-time capable scheduler, which reduces maximal scheduling latencies for real-time applications from over a second to just a few milliseconds [6]. The approach uses the `RT_PREEMPT` patch to equip Android's Linux kernel with basic real-time support, but additionally allows reliable priority-based process scheduling for background services and a non-blocking automatic memory management. Furthermore, RTAndroid is fully backward compatible to already existing Android components and applications. New features and real-time components can be used in the standard Android manner, e.g. simply by extending introduced classes like a real-time service.

Fail-safe state systems can be executed on RTAndroid using reliable background services. Instantiating a new service based on the new `ServiceRT` class in RTAndroid prevents the corresponding process from being shut down or killed by the system. Thus, safety-critical applications can be executed in real-time prioritized Linux processes with non-blocking, concurring garbage collector. In this case, the application stays responsive and precise scheduling guarantees bounded reaction times to upcoming events, such that a fail-safe state can be entered.

3 Example Pattern

Chemical process control is considered as one of common examples for real-time and safety-critical system with multiple inputs and outputs. Let us have a small experimental reactor to perform a specific chemical reaction. It is clear that such an application includes a set of measured values like temperature, pressure, gas concentration and many others. Any failure in such system could lead to critical situation. The Monitor-Actuator Pattern [2] is suitable for this case due to the low availability requirement and the existing fail-safe state. As shown in Fig.1, the pattern consists of two heterogeneous channels that run independently: The main actuation channel can be a microcontroller that reads the measured values from input sensors and provides a check on the input data and the system itself to ensure the correct processing of the desired operations. The Monitoring channel is used to improve the safety of the system by providing a continuous monitoring for the actuation channel. It takes the information from the set point source and the actuator sensors and compares it with the provided set points to detect possible faults in the actuation channel. In the case of improper operation, it forces the actuation channel to enter the fail-safe state. The monitoring channel should be independent from the actuation channel, so Android device is selected in our approach to perform this task. The heterogeneous redundancy in the two channels reduces the possible concurrent failure. Finally, the real-time extension and the life cycle assessment ensure the real-time requirement for this application and the proper time to call the safety function to enter the fail-safe state.

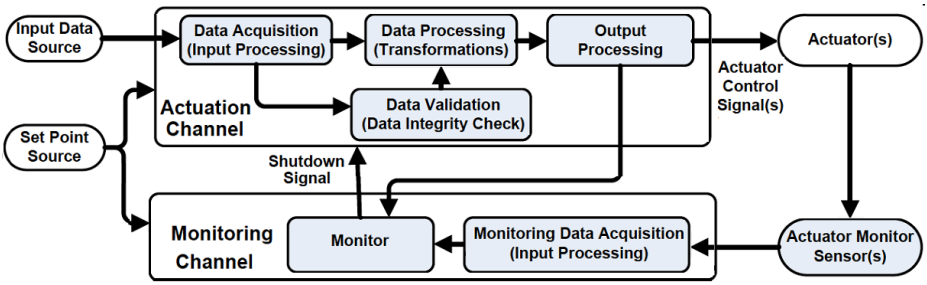

Fig. 1. The Monitor-Actuator Pattern

4 Conclusion

The design of safety-critical embedded systems requires the integration of common design methods with the recent advances in information technologies. In this paper we presented a conceptual approach to implement safety-critical systems with fail-safe state on Android devices by using appropriate safety design patterns. Moreover, a real-time extension and a corresponding life cycle management have been used for Android to ensure the non-functional requirements in the developed application. As shown in the example, the proposed approach is more suitable for the cases with low availability and clear fail-safe sate.

This work was supported by the UMIC Research Centre, RWTH Aachen University, Germany.

References

1. Armoush, A.: Design patterns for safety-critical embedded systems. PhD thesis, RWTH Aachen University (2010)
2. Douglass, B.P.: Real-time design patterns: robust scalable architecture for real-time systems. Addison-Wesley Professional (2003)
3. Dunn, W.R.: Designing safety-critical computer systems. Computer 36(11), 40–46 (2003)
4. Franke, D., Elsemann, C., Weise, C., Kowalewski, S.: Reverse Engineering of Mobile Application Lifecycles. In: Proc. 18th Working Conference on Reverse Engineering (WCRE 2011), pp. 283–292. IEEE (2011)
5. Franke, D., Kowalewski, S., Weise, C., Prakobkosol, N.: Testing Conformance of Lifecycle-Dependent Properties of Mobile Applications. In: Proc. 5th International Conference on Software Testing, Verification and Validation (ICST 2012), pp. 241–250. IEEE (2012)
6. Kalkov, I., Franke, D., Schommer, J.F., Kowalewski, S.: A Real-time Extension to the Android Platform. In: Proceedings of the 10th International Workshop on Java Technologies for Real-time and Embedded Systems (JTRES 2012), pp. 105–114 (2012)