Autonomous Recovery Technique of Software Bus Based on VxWorks Operating System

Xing Weiyan, Liu Dong, Li Ming, Jin Pengfei, *Member, IAENG*

Abstract—**In the paper, an autonomous recovery mechanism based on VxWorks operating system was introduced so as to improve the reliability of embedded software. The mechanism uses software bus technique to manage the software components in VxWorks. A component is defined as a software module, which is usually a process. All software components are mounted on the software bus. And the components are divided into two types according to the way that they influence the system, namely, cold backup components and hot backup components. The autonomous recovery mechanism assures that at least one component is in work state. The software bus architecture and the reusable component technology can improve the reliability, scalability, and maintainability of embedded systems.**

Index Terms—**VxWorks, autonomous recovery, software bus, signal mechanisms.**

I. Introduction

WITH the development of computer technology and the expansion of its applications, the reliability requirements of computer systems are increasing. In the aerospace, aviation, finance and other important areas, the huge economic losses and the disastrous consequences maybe occur since the work cannot be achieved in specified time [1]. Therefore, in order to ensure the work (or the program) of embedded systems to be done normally, the reliability of the systems should be improved. Software method is usually introduced in embedded systems to tolerate the hardware faults and the software faults, which may break the systems' work down [2][3].

VxWorks is a high-performance, real-time, reliable embedded operating system, which has become widely used as the operating system for embedded systems. However, the applications in operating system maybe introduce errors because of the bugs in the programs. Therefore, it is important to use some special methods to maintain the reliability of the VxWorks based embedded systems in a complex application environment.

This paper puts forward a software bus based method to improve the embedded systems. On studying the common faults of embedded systems [4][5], the paper introduces an autonomous recovery management mechanism, where the application programs, or components, are especially designed and can be autonomously recovered. The mechanism ensures the embedded systems to work normally by shielding the failures when fault occurs in the systems.

In this paper, the redundancy scheme, the component-based software bus architecture, the autonomous recovery technology based on software bus and some other technologies are designed.

Section 2 and section 3 introduce the design and the implementation of the software bus respectively. In section 4, the paper tells how to develop the components using autonomous recovery management. And the conclusion is made in section 5.

II. The Architecture of the Component-Based Software Bus

The fault-tolerant technology is one of the ways to provide the autonomous recovery ability of the embedded systems [6]. It often means that the embedded systems can provide the continuous services when a fault occurs by using redundant resources. In this paper, two kinds of redundancy schemes are taken into account, namely cold backup and hot backup. The two redundancy schemes are managed by autonomous recovery management software.

The autonomous recovery management software monitors the faults in the system and assigns redundant resources according to the different requirements of applications. The management software takes the appropriate strategy according to the type of the faults monitored to ensure the applications to be recovered.

Usually, some principles for the design of embedded systems should be taken into account. Firstly, the general important specification should be recognized so as to help understanding the advanced relationships in the system, which is useful to establish new system on the basis of the old system. Secondly, the right architecture is the key element that ensures the success of software design, and it also can be used as a framework which meets the demand of analyzing and managing the system. Thirdly, the architecture can be used as the medium for different developers in the system development process. Finally, the architecture can help overcome the obstacles of software reuse, which means that it provides a road to reuse software based on software architecture.

In this paper, we provide a component-based software bus architecture. The architecture is scalable, maintainable and reusable. It supports the "plug and play" of components and the dynamic configuration of service interface, which make

Manuscript received February 15, 2014.

Xing Weiyan is with the Equipment Academy, Beijing, 101416 China (e-mail: xingweiyanld@163.com).

Liu Dong is with the Equipment Academy, Beijing, 101416 China (e-mail: LD5M@163.com).

Li Ming is with the Military Economics Academy, Wuhan, Hubei Province, 430035 China (email: 13007143832@163.com).

Jin Pengfei is with the Equipment Academy, Beijing, 101416 China (e-mail: 18700401381@163.com).

the target system have a good flexibility. At the same time, due to the reduction of the coupling among the various components, each component can be independently developed or modified as long as the interface provided is not changed. This is especially helpful for parallel development. Since the problem of component integration is not considered, it allows developers to focus on business logic implementation, which improves development efficiency. In addition, the use of the component-based software bus architecture makes a clear structure, which is conducive to system maintenance and modification.

Therefore, on the basis of the above mentioned principles as well as the advantages of software bus, the component-based software bus architecture is put forward as the autonomy recovery management software architecture in this paper.

A. *Component-based development method*

Software components have different meanings in different contexts. For example, it can be functional modules, classes, objects, or a group of related functions. Other definitions of software component include standard library, framework, reusable software products, and so on. Usually, the component has the following characteristics: 1) component is self-contained software which has one or more well-defined interfaces; 2) it is binary reusable, complies with certain component specification, and can combine with other components into a system where it is replaceable.

Component-based development method contains two life cycles which are the development of reusable software components and the development based on reusing component. The development process is very different from the traditional software development. In the various stages of the life cycle, the development focuses on the software reuse idea. Based on the reusable software component, the component production and reusing are interrelated, which fully implement the idea of software reusing in the various stages of the software development process. In this paper, the component based method is used in the process of developing autonomous recovery management software, which improves the maintainability of the system.

B. *The architecture based on software bus*

1. Component design

1) Business component

The business components in the business module are responsible for the normal working of the business in embedded systems. The idea of business component redundancy in this paper is that two copies of the component perform the same function, thereby increasing the availability of the business component. In this paper, we provided two schemes, namely cold backup redundancy scheme and hot backup redundancy scheme.

In cold backup redundancy scheme, a copy of the business component is used as the main working copy to perform business function. The other copy is used as the cold backup which is not running when the main working copy is working. When a fault occurs in system, the autonomous recovery component discovers the fault in time and starts the cold backup copy that takes over the business of the fault component, in which the fault occurs. During the process, the work of the embedded system is not disturbed.

In hot backup redundancy scheme, the two components copies are running at the same time, one of them is used as the main working copy to perform business function. The other copy is used as the hot backup, which is running but not performs business. When a fault occurs in system, the autonomous recovery component discovers the fault in time and makes the hot backup copy take over the business of the fault component in which the fault occurs. During the process, the work of the embedded system is not disturbed.

2) The basic fault-tolerance component

In this paper, the basic fault-tolerance components in cold backup redundancy scheme and hot backup redundancy scheme are designed. The basic fault-tolerance components include basic message requesting component, basic request responding component, basic fault detection component, basic response detection component and dynamic reconfiguration component. The business component is mounted on the software bus by the basic request responding component. And other basic components are assembled into the autonomous recovery component.

Usually, the requirements of the embedded system are ever-changing, so the number of the business components and the type of redundancy scheme are also changing in the system, which requires that the system should have a good scalability. In this paper, the software bus and the component-based technology are combined in the process of the development of autonomous recovery management software, which solves the above problem. The process of reusing the components being is shown in the Fig. 1.

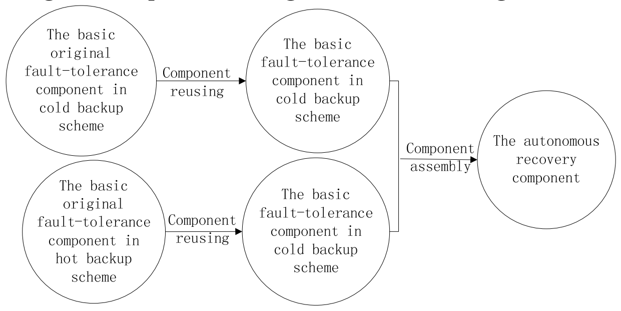
The basic original fault-tolerance component in cold backup scheme
Component reusing
The basic fault-tolerance component in cold backup scheme
Component assembly
The autonomous recovery component
The basic original fault-tolerance component in hot backup scheme
Component reusing
The basic fault-tolerance component in cold backup scheme

Fig. 1 The process of reusing components

3) The autonomous recovery component

The interaction between the autonomous recovery component and the business components in cold backup redundancy scheme and hot backup redundancy scheme make the business component have the ability of recovery. In autonomous recovery component, the invocation process to basic tolerant component is described as follows:

(a) Send the request message to the business component by the message requesting component;

(b) Business component responds by request responding component, and the response message is sent to the fault detection component;

(c) After the fault detection component get the response message from the business component, it invokes the response detection component to detect the message.

(d) If the business component is found bad, the fault detection component invokes the dynamic reconfiguration component to recovery the bad component by autonomous recovery.

4) Communication component

In this paper, there are two types of communication components, one is responsible for the interaction between business components and the autonomous recovery component, the other is responsible for the interaction between the business components.

2. The design of software bus architecture

As shown in Fig. 2, any business components in redundant resource repository (such as A or B) is regardless of what kind of the function that they have, as long as they follow the interface standard provided by the software bus. And the business components can be directly plugged into the software bus, by which they can interact with the autonomous recovery component so as to have the ability of autonomous recovery. Similarly as long as the business components follow the communication interface, which the software bus provided, they can be integrated directly into the system environment and have the ability of communication between the components by the service provided by data communications components in software bus. Autonomous recovery component and autonomous recovery backup component can be independently developed. As long as they follow the interface standard provided by the software bus, they can provided the autonomous recovery service for the business components. Autonomous recovery backup component is responsible for monitoring the autonomous recovery component and taking over the autonomous recovery component when a fault is found in the latter.

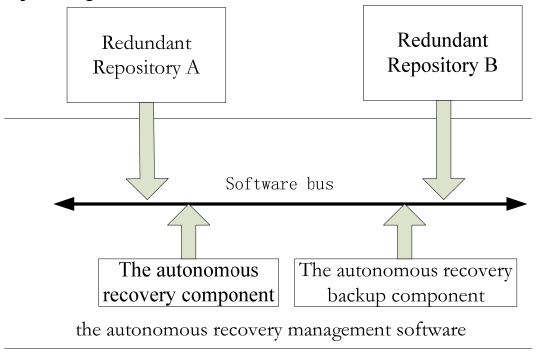
Redundant Repository A
Redundant Repository B
Software bus
The autonomous recovery component
The autonomous recovery backup component
the autonomous recovery management software

Fig. 2 Software bus architecture

The software bus provides some necessary functions such as component management, data management, component register management, message management, and so on.

Component management: providing the dynamic reconfiguration management for the business components and the autonomous recovery component in the form of the task in VxWorks.

Data management: providing the management for the communication among the business component and the communication between the business component and the autonomous recovery component.

Component register management: the information about the business component and the auxiliary information about the autonomous recovery function are registered with the software bus. The register information contain the scheme that includes the cold backup and the hot backup the business component adopts, the number of the business component in the corresponding redundant scheme, the autonomous recovery strategy and so on. The above information is written in the file, which is loaded when the system is started, and the signal and memory resources are allotted.

Message management: providing management for signal mechanism in VxWorks.

III. THE IMPLEMENT OF THE AUTONOMOUS RECOVERY TECHNOLOGY BASED ON THE SOFTWARE BUS

A. The type of fault

In “no response” failure, a request is sent from the business component to the autonomous recovery component (see Fig. 3). If the business component doesn't response, the “No Response” fault happens in the business component.

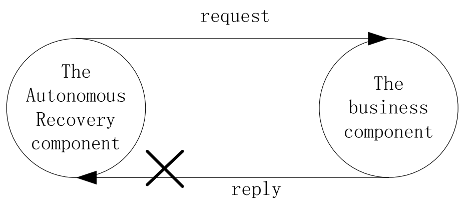

Fig. 3 The show of the fault named “no response”

B. Resource Allocation

At the beginning of the system, the system provides some resources that are signal and the shared memory for the business component, which can accept the service of the autonomous recovery component. The interaction between the business component and the autonomous recovery component by the method, which the software bus provides in the form of the interface, makes that the autonomous recovery component can monitor the business component.

1. Response structure

The software bus provides a response structure for the business component which is plugged into the software bus by the interface provided by the bus. The structure is shared by the business component and the autonomous recovery component. The response structure contains the ID of the business component and the system time named Now_time.

2. Signal resources

Autonomous recovery component sends requests to the business components through the signal mechanism. The business components response after they received the requests. And the response message of the process is written to the response structure which is checked by the autonomous recovery component.

C. The autonomous recovery of software bus

Shown in Fig. 4, autonomous recovery component sends a signal to the business component, the business component response after receiving the signal and waits for the autonomous recovery component checking up the information which is written in the shared structure. If the time in the shared structure is between t1 and t2, the reaction of the business component is normal, or the reaction is not normal. If the reaction of the business component is not normal, the autonomous recovery component provides failure recovery service for the business component by invoking dynamic reconfiguration component. In the cold backup scheme, the method of failure recovery is to replace the main work component by starting the cold backup. In the hot backup scheme, the method about failure recovery is that the hot backup takes over the main work component in real time.

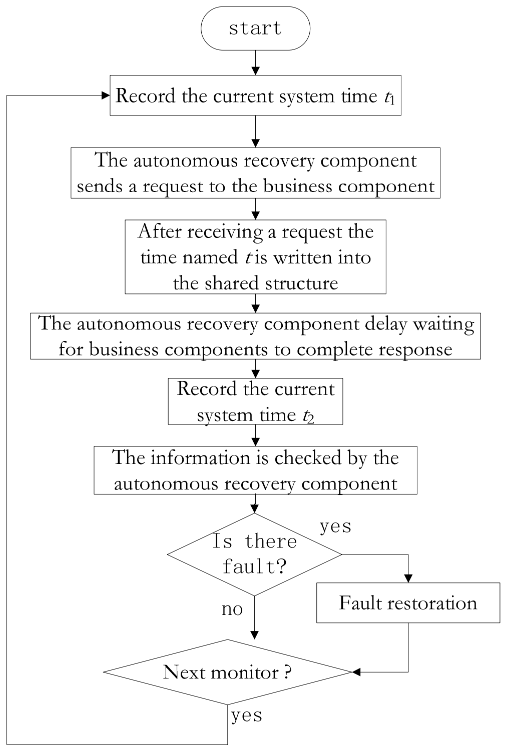
start
Record the current system time t_1
The autonomous recovery component sends a request to the business component
After receiving a request the time named t is written into the shared structure
The autonomous recovery component delay waiting for business components to complete response
Record the current system time t_2
The information is checked by the autonomous recovery component
Is there fault?
yes
Fault restoration
no
Next monitor ?
yes

Fig. 4 The process of autonomous recovery

D. *The design of the autonomous recovery management software*

Since the autonomous recovery component is central to the autonomous recovery software, it should be especially designed so that it can be replace by a spare component if it fails. The implementation scheme is list below:

1) Start the autonomous recovery component prior to a spare one, and the time interval between these two components is *t*.

2) Update t_1 and t_2 using the systematic time in the period of *Ta* and *Tb*. If the autonomous recovery component is in working state, the time space between t_1 and t_2 should be less than *Tb*. Otherwise, the autonomous recovery component works improperly. The relationship among t_1, t_2, *t*, *Ta* and *Tb* is shown in Fig. 5.

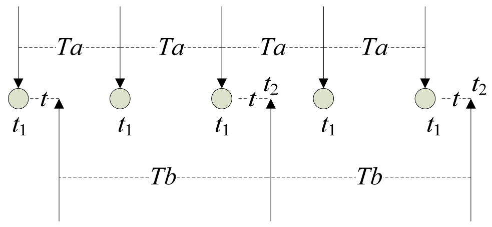

Fig. 5 The relationship of different time

IV. THE DEVELOPMENT PROCESS BASED ON AUTONOMOUS RECOVERY MANAGEMENT SOFTWARE

The steps of system development based on autonomous recovery management software in this paper are list below:

1. The determination of demand

1) Determine the number of business component in the system.

2) Determine the redundancy scheme of various business components, namely whether the cold backup or the hot backup scheme is needed to be adopted in the system.

3) Determine the possible fault types of each business component.

4) Determine the communication relationships among business components.

2. Component reusing

1) According to requirements, increase or decrease the number of the resource allocated in software bus such as sharing structure, signal resource, and so on.

2) According to the requirements, the original basic fault-tolerant component is modified to make it meet the new demand.

3) According to the communication relations among the business components identified in the requirements, modify the communication components.

3. The assembly of components

1) The basic fault tolerance components are assembled into autonomous recovery component according to the certain logic.

2) Various modules are associated by software bus, and then, it can be downloaded to the target machine after it is compiled.

V. CONCLUSIONS

In order to improve the reliability of the embedded software, this paper puts forward the component-based software bus as the autonomous recovery management software architecture, where the autonomous recovery technique is realized by redundant technology.

The software bus, designed in this paper, is more the underlying communication platform, since is uses the signal mechanisms and the shared memory mechanism of the underlying library functions of VxWorks operating system. Therefore, compared with the communication function which is designed by common software developer, the software bus has better security and reliability.

REFERENCES

[1] W. Shuda, "Study and design of the structure of on-board system software based on software bus," Computer Engineering, vol. 29, pp. 39-41, 2003.
[2] S. Changai, J. Maozhong, L. Chao, "Overviews on software architecture research," Journal of Software, vol. 13, pp. 1228-1235, July 2002.
[3] H. Huimin, L. Qiurang, Z. Kailong, "Hybrid self fault-tolerant mechanism of embedded multi-tasks software," Computer Engineering, vol. 18, pp.47-49, 2011.
[4] L. Zhengkui, Y. Deli, "Software component reused technology overview," Computer Engineering and Design, vol. 25, pp. 877-880, June 2004.
[5] G. Deconinck, "Software-implemented fault tolerance arid separate recovery strategies enhance maintainability," IEEE Trans. on Reliability, vol. 51, pp. 158-165, Feb. 2002.
[6] M. Oussalah, A. Smeda, and T. Khammaci, "An explicit definition of connectors for component based software architecture," in Proceedings of the 11th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems, 2004