A Hooking Method running on MHAP Environment

Eung-Nam Ko
Division of Information & Communication
Baekseok University
115 Anseo-dong, Dongnam-gu, Cheonan, Chungnam, Korea, 330-704
Korea
ssken@bu.ac.kr

Abstract: This paper proposes performance analysis of an error detection method by comparison of hooking with snatch method for MHAP(MOM-based on Home Automation Platform). A snatch is a method which intercepts all GDI(Graphic Device Interface) event. GDI snatching method is an event communication of hardware oriented between application program and graphic device driver. A hooking is a method which intercepts a communication message between window processes. This paper explains a performance analysis of an error detection system running on MHAP using the rule-based DEVS(Discrete Event System Specification) modeling and simulation technique.

Key-Words: error detection method, snatch method, GDI, hooking method, DEVS

1 Introduction

With the rapid development of multimedia and network technology, more and more digital media generated. With such huge amount of data, efficient means to index the content for future retrieval have to be made available. Efforts have been made in the literature along both directions, some in single media domain and some in multimedia domain[1-5]. These components are not always guaranteed to support enough reliability and availability for the applications. It is critical to discuss how to make and keep the systems so reliable and available that even fault-tolerant applications could be computed in the systems[6]. One of conventional methods detects an error by intercepting all GDI event. This paper proposes a method of detection to find an error by hooking only the process. In this paper, we would like to discuss a new method of performance analysis of error detection system running on MHAP environment using the rule-based SES and DEVS modeling and simulation technique. In DEVS, a system has a time base, inputs, states, outputs, and functions. The purpose of this paper is to compare and analyze a performance of proposed method with conventional method by using DEVS formalism for a fault detection system running on MHAP environment.

The rest of this paper is organized as follows. In section 2, Discrete event modeling, DEV(S(Discrete Event System Specification) formalism ,and SES(System Entity Structure) are reviewed. In section 3, we show hooking technique of EDA running on MHAP environment. In section 4, we propose a modeling for an error detection and performance analysis of proposed method with conventional method by using DEVS formalism. Finally, in section 5, we summarize our paper.

2 Related Works

The DEVS-Scheme environment is based on two formalism: discrete event-system specification(DEV)[7,8]. In this section, DEVS are reviewed.

2.1 Discrete Event Simulation

Differential equations employed to describe continuous systems have a long history of development whose mathematical formalization came well before the advent of the computer. In contrast, discrete event simulations were made possible by, and evolved with, the growing computational power of computers. Since the early 70’s work has been proceeding on a mathematical formalism for modeling discrete event systems. One approach, inspired by the systems theory concepts of Zadeh and Dossoer(1963), Wymore(1967), Mesarovic and Takahara(1975), and Arbib and Padulo(1974), attempted to cast both continuous and discrete event models within a common systems modeling framework. This approach was elaborated in a number of publications primarily summarized in the books(Zeigler, 1976) and (Zeigler, 1984a), and is reviewed in (Zeigler, 1984b). Systems modeling concepts were an important facet in a movement to develop a methodology under which
simulation could be performed in a more principled and secure manner. The recent advent of high performance artificial intelligence software and hardware has facilitated the transfer of this simulation methodology from research to practice (Elzas et al., 1986)[8,11].

2.2 DEVS formalism
The DEVS formalism introduced by Zeigler provides a means of specifying a mathematical object called a system. The DEVS formalism is a theoretical, well grounded means of expressing hierarchical, modular discrete event models. Basically, a system has a time base, inputs, states, and outputs, and functions for determining next states and outputs given current states and inputs. In the DEVS formalism are defined by the structure[9-12].

\[ M = \langle X, S, Y, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, t_s \rangle \]

where
- \( X \): a set of input events,
- \( S \): a set of sequential states,
- \( Y \): a set of output events,
- \( \delta_{\text{int}} : S \rightarrow S \): internal transition function,
- \( \delta_{\text{ext}} : Q \times X \rightarrow S \): external transition function,
- \( \lambda : S \rightarrow Y \): output function,
- \( t_s : \) time advance function.

Basic models may be coupled in the DEVS formalism to form a multi-component model which is defined by the structure[9-12].

\[ DN = \langle D, \{M_i\}, \{I_i\}, \{Z_{ij}\}, \text{select} \rangle \]

where
- \( D \): Diagraph Network,
- \( \{M_i\} \): a component basic model
- \( \{I_i\} \): a set, the influences of I and for each j in \( I_i \),
- \( \{Z_{ij}\} \): a function, the I-to-j output transition,
- select: a function, the tie-breaking selector.

3 An Error Hooking Technique
This paper is described a performance analysis of an error detection system running on MHAP environment using the rule-based DEVS modeling and simulation technique.

3.1 MHAP Environment
As shown in figure 1, MHAP has four layered architecture[13].

![Fig. 1 The organization of MHAP with four layered architecture](image)

The physical device and network layer consists of any home network and physical device supporting any home networking technology. The infrastructure layer introduces infrastructure such as OSGi to provide service management and deployment functions for MHAP services. The MHAP layer consists of MHAP services and provides functionalities constructing HA, which includes event notification, appliance control, HA rule configuration and device management. It uses MOM to support event-driven HA in heterogeneous environment. Facilitating Home Automation needs many different kinds of applications. There are DOORAE agent layer between application layer and MHAP service layer.

3.2 Hooking Technique for EDA based on MHAP
EDRA consists of EDA (Error Detection Agent) and ERA (Error Recovery Agent). EDA consists of ED (Error Detector), EC (Error Classifier) and EL (Error
Learner). EDA is an agent which plays a role in detecting, and classifying errors. ED is an agent which plays a role as an interface to interact among an application, EC and EL. As shown in figure 2, ED has functions which detect an error by using hooking technique.

EDA detects an error by using a hooking method in MS-Windows API(Application Program Interface). When an error occurs, A hook is a point in the Microsoft Windows message-handling mechanism where an application can install a subroutine to monitor the message traffic in the system and process certain types of messages before they reach the target window procedure. Windows contains many different types of hook. The roles of error and application program sharing are divided into two main parts; Abstraction and sharing of view generation. Error and application program sharing must take different from each other according to number of replicated application program and an event command. This proposed structure is distributed architecture but for error and application program sharing, centralization architecture is used. Error and application program sharing windows perform process communication of message form. In the middle of this process, there are couple ways of snatching message by error and application sharing agent. ED informs EC of the results of detected errors. ED inspects applications by using a hooking technique to find an error. EC and EL deal with learning in reactive multi-agent systems. Generally, learning rules may be classified as supervised or unsupervised. KB has a registration information of creation of service handle and session manager handle by Daemon and GSM. EC can decide whether it is hardware error or software error based on learning rules by EL. In case of hardware error, it cannot be recoverable. In case of software error, it can be recoverable. This approach is based on the idea of comparing the expected error type which is generated by an EL with the actual error occurred from sites.

4 Performance Analysis by DEVS
To evaluate the performance of the proposed system, an error detection method was used to compare the performance of the proposed model against the conventional model by using DEVS formalism.

(Simulation 1)
In the first simulation, we have considered composition component as shown in Table 1. The atomic models are EF, RA1, and ED1. The combination of atomic models makes a new coupled model. First, it receives input event, i.e., polling interval. The value is an input value in RA1. An output value is determined by the time related simulation process RA1. The output value can be an input value in ED1. An output value is determined by the time related simulation process ED1. We can observe the result value through transducer.

<table>
<thead>
<tr>
<th>Component</th>
<th>State Variable</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>EF(genr)</td>
<td>Poll_int</td>
<td>Polling interval</td>
</tr>
<tr>
<td>RA1</td>
<td>Ra_re_time</td>
<td>Response time</td>
</tr>
<tr>
<td></td>
<td>App_cnt</td>
<td>The number of application program</td>
</tr>
<tr>
<td></td>
<td>Ra_re_t_a</td>
<td>Accumulated response time</td>
</tr>
<tr>
<td>ED1</td>
<td>Tat_t_a</td>
<td>RA accumulated response time</td>
</tr>
</tbody>
</table>

(Simulation 2)
In the second simulation, we have considered composition component as shown in Table 2. The atomic models are EF, RA2, and ED2. The combination of atomic models makes a new coupled model. First, it receives input event, i.e., polling interval. The value is an input value in RA2. An output value is determined by the time related simulation process RA2. The output value can be an input value in
An output value is determined by the time related simulation process ED2. We can observe the result value through transducer.

<table>
<thead>
<tr>
<th>Component</th>
<th>State Variable</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>EF (genr)</td>
<td>Poll_int</td>
<td>polling interval</td>
</tr>
<tr>
<td>RA2</td>
<td>Ra_re_time</td>
<td>Response time</td>
</tr>
<tr>
<td></td>
<td>App_cnt</td>
<td>The number of application program</td>
</tr>
<tr>
<td></td>
<td>Ra_re_t_a</td>
<td>Accumulated response time</td>
</tr>
<tr>
<td>ED2</td>
<td>Tat_t_a</td>
<td>RA accumulated response time</td>
</tr>
</tbody>
</table>

Table 2  Atomic Model and State Variable

We can observe the following. The error detected time interval is as follows.

Conventional method:
\[ 2 \times \text{Poll_int} \times \text{App_cnt} \]

Proposed method:
\[ 1 \times \text{Poll_int} \]

Therefore, proposed method is more efficient than conventional method in error detected method because of App_cnt \( \geq 1 \). We have compared the performance of the proposed method with conventional method.

The problem of rapid increase in communication load due to one transmission even with presence of many users, using simultaneous broadcasting. Application program sharing windows perform process communication of message form. In the middle of this process, there are couple ways of snatching message by application sharing agent, windows hooking method and GDI snatching method.

For example, we have compared the performance of hooking with snatch method during some interval time by using MS-Word and MS-Powerpoint as shown table 3. Therefore, proposed method is more efficient than conventional method in error detected method because of function number.

Table 3. Comparison of Hooking with Snatch Method during Some Interval

<table>
<thead>
<tr>
<th>Event</th>
<th>Hooking</th>
<th>GDI snatch</th>
<th>Event</th>
<th>Hooking</th>
<th>GDI snatch</th>
</tr>
</thead>
<tbody>
<tr>
<td>MS-Word</td>
<td>129,395</td>
<td>44,668</td>
<td>MS-Powerpoint</td>
<td>887</td>
<td>361</td>
</tr>
</tbody>
</table>

5 Conclusion

The main idea is to detect an error by a hooking method. Also it is to classify the type of errors by using learning rules. The merit of this system is to use the same method to recovery it as it creates a session. EDA is a system that is able of detecting software an error based on MHAP environment. Conventional method detects an error by snatching it. This paper explained a performance analysis of an error detection system running on MHAP environment using the rule-based DEVS modeling and simulation technique. The weak point of this system is limited to DOORAE.

Our future work is to extend to autonomous agents for detecting and recovering error and to generalize it to adjust any other system. The focus of situation-aware ubiquitous computing has increased lately. An example of situation-aware applications is a multimedia education system. The development of multimedia computers and communication techniques has made it possible for a mind to be transmitted from a teacher to a student in distance environment. This paper proposes an Adaptive Fault Tolerance (AFT) algorithm in situation-aware middleware framework and presents its simulation model of AFT-based agents.

References:


