Research on Assembly and Fault-tolerant of Interface Component in Distributed Human-computer Interactive System

Ming-chuan Zhang, Hong-qi Wang, Shi-bao Sun, Qingtao Wu, Guan-feng Li
College of Electronic and Information Engineering Henan University of Science and Technology, Luoyang, China
zhlzmc@163.com, haustwhy@gmail.com, sunshibao@126.com, wqt8921@126.com, haustlgf@gmail.com

Abstract—In order to realize the human-computer interactive interface fault-tolerant, dynamic reconfiguration and uniform control of interface switching of each human-computer interactive node in distributed system, a rapid developing model is proposed to build human-computer interactive system recurring to an assistant developing tool designed self-owned. Subsequently, a fault-tolerant scheduling model is presented as well as corresponding scheduling algorithms based on the rapid developing model. This model is made up of Center Scheduling Component and many human-computer interactive nodes. Each human-computer interactive node includes a Local Scheduling Component, an Interface Generating Component and some human-computer interactive interface components. Finally, feasibility of the model and correctness of the algorithms are proved by the simulation experiment.

Index Terms—distributed system, interface component, human-computer interactive system, component assembly, scheduling model

I. INTRODUCTION

With the development of computer technology, users have higher expectations for computer system, especially to the human-computer interactive system (HCIS). Not only do users require that the human-computer interactive interfaces (HCIIIs) are beautiful, easy to use, response sensitive, but also require HCIS possess reliability, ability to resist trouble and dynamic reconfiguration etc. The main purposes this paper studies are to solve three problems in distributed system, which are the fault-tolerant (FT), the dynamic reconfiguration and the uniform control of the human-computer interactive interface components.

Recent years, the dynamic reconfiguration and FT problems are paid attention to by more and more researchers in distributed system. Remote dynamic component configuration is discussed in reference, which greatly improves system flexibility using configuration files [1]. The dynamic deployment and re-configuration of pervasive service components in a self-controlled manner are researched. In particular, a service component self-deployment algorithm using partitioning techniques and a simple service re-configuration algorithm are proposed and evaluated. The effectiveness of the proposed mechanisms is proved by the experiment results [2]. A new method of QoS-aware and dynamic configuration for Web services composition is presented to improve the adaptive capacity to both the QoS variability of component services and the failure-prone environment [3]. There are two topics which are researched in reference [4]. First, it describes optimizations applied to an implementation of the OMG’s Deployment and Configuration of Components specification that enable performance trade-offs between QoS aspects of DRE systems. Second, it compares the performance of several dynamic and static configuration mechanisms to help guide the selection of suitable configuration mechanisms based on specific DRE system requirements. Two methods of dynamic reconfiguration are introduced. First, using configuration file, this method belongs to static configuration ways. Second, utilizing configuration operation in the program, this method belongs to dynamic reconfiguration ways, which can adapt to some configuration situations that can't be estimated in advance. The software architecture supporting dynamic reconfiguration is studied in reference [5]. It is solved with the graph-oriented programming method, which realizes dynamic reconfiguration and the description of software architecture based on components in the uniform way. Certain problems of components dynamic reconfiguration are researched in references [3-5], and some achievements are got. However, there are some difficulties in practical application. Furthermore, systemic fault-tolerant has not been considered.

In references [6-8], FT and dynamic reconfiguration are studied in distributed system. Among them, the FT and dynamic reconfiguration are realized in different technology. However, the FT and dynamic reconfiguration have not been considered in the high level, as well as uniform control of distributed system. Component frameworks provide the strategy for the development and deployment of complex multiphysics applications to satisfy the need [9]. In order to build dynamically adaptable applications, the service-oriented component models supporting the dynamic availability of components at run-time are researched as well as offering the possibility in reference [10].

This paper considers dynamic reconfiguration and FT of human-computer interactive interface components, as well as uniform control of interface switching of each node synthetically. A rapid developing model, a fault-
tolerant scheduling model and corresponding scheduling algorithms are presented to resolve the problems.

II. THE ASSEMBLY MODEL OF INTERFACE COMPONENT

There are two kinds of components in the human-computer interactive system. One is framework component. The other is business component (interface component). There is at least a component that is called main framework for one application. The main framework denotes the business logic relation of the application. The business components and framework components are assembled by the main framework to realize the application business logic. The assembly architecture of component and framework is shown as Figure 1. Because the main framework is the first entity that is assembled and the function of other components is to cooperate with the main framework, its actual function is a component container. The component assembly architecture is shown as Figure 2.

Figure 1. the assembly of component and framework

Figure 2. the architecture based on component assembly

The assembly characteristic of the business component and framework component are illuminated as follows.

(1) Business components can only be assembled to framework components. Business components can not assemble business components.

(2) Framework components can be assembled to framework components. That is to say, simple business logic combines with other business logic to gain complicated business logic.

(3) The workflow of application is determined by framework assembly architecture. The main framework calls framework component and business component. The higher framework calls lower framework and business component. The system function is realized by iterative. The calling flow can be reversed.

(4) The converter, mapping and glue for no matching interfaces are done by framework component.

III. THE RAPID DEVELOPING MODEL OF HCIS

The human-computer interactive system can be developed rapidly recurring to an assistant developing tool which is similar to the software of drawing and designed self-owned. Each component in the component library has its style which can be used directly to build the interface style of the target system. For the components nonexistent in component library, the interface style can be designed by the assistant developing tool.

When designing a human-computer interactive system, the style of human-computer interface is gained recurring to the assistant developing tool, as well as the designing result file. The target system can be build based on the file. The rapid developing model is shown as Figure 3 and the developing steps are shown as follows.

Figure 3. the rapid developing model of HCIS

(1) Determining components: abstracting the requirement and determining system framework model to gain the components which build the target system based on the requirement analysis.

(2) Determining the flow of interface switching: analyzing application workflow to gain the flow of interface switching.

(3) Gain designing result: generating interface designing result file which includes the ID of framework component, the ID of business component, the ID of the component needing developing and interface switching information based on the step one and step two.

(4) Component developing: developing the new components whose ID is generated in step three and putting them into the component library.

(5) Component assembly: assembling the various components to gain target system according to the designing result file which is generated in step three.

(6) Assembly testing: testing the target system, if the target system satisfies the need, the designing process is accomplished. If not, the process will be iterative.

IV. A FAULT-TOLERANT SCHEDULING MODEL

Several definitions used in the fault-tolerant scheduling model and scheduling theorem are introduced.

Definition 1: An Interface Scheduling Task (IST) refers to a kind of prearrange scheme to the ICs
scheduling. An Interface Scheduling Request (ISR) refers to an applying command which makes the IST begin to carry out. IST is a static concept, and ISR is a dynamic concept.

Definition 2: The Interface Scheduling Task Set (ISTS) is a 2-tuple, ISTS = {ISTnft, ISTft}. Among them, ISTnft means the tasks set without FT requirements. If the scheduling of ISR of ISTnft fails, nothing will be done. ISTft means the tasks set with FT requirements. If the scheduling of ISR of ISTft fails, the ISR will be executed repeatedly by the system.

Definition 3: The Set ISTnft is a 4-tuple, ISTnft = <SN, DN, CS, P>. Among them, SN refers the node which sends ISR. DN refers the node which ICs scheduling happens in. CS is the lists of ICs which are needed showing. P refers the priority of ISR.

Definition 4: The Set ISTft is a 5-tuple, ISTft = <SN, DN, DI, CS, P>. Among them, SN refers the node which sends ISR. DN refers the node which ICs scheduling happens in. DL is the deadline of ISR finished. CS is the lists of ICs which are needed showing. P refers the priority of ISR.

Definition 5: The SN and DN of an IST is the same node, the task is called Local Scheduling Task (LST), and the corresponding ISR is called Local Scheduling Request (LSR). Otherwise the task is called Uniform Scheduling Task (UST), and the corresponding ISR is called Uniform Scheduling Request (USR).

Definition 6: CS is a list, CS = {Componentk|0<k<=N, Componentk: ICs}. There, N is the maximal number of ICs which a display can show synchronously.

Theorem 1: The IST in set ISTnft can all be succeeded in carrying out, if and only if for each ISR of each IST, the interval from beginning that the ISR executes successfully to DL is greater than the interval that the scheduling course needs. This is described as (1).

\[ ∀\text{IST} ∈ \text{IST}_{nft}, \text{IST} \rightarrow \text{success} \iff ∀\text{IST} ∈ \text{IST}_{f}, \text{VISIR} ∈ \text{IST} \land \text{DL} − \text{ISRST} ≥ \text{ET} \ (1) \]

Among them, \( \rightarrow \) refers executing, success refers executing successful, ISRST refers the beginning time of the ISR executing successful, ET refers the time that executing scheduling needs.

This model is made up of Center Scheduling Component and many human-computer interactive nodes. Each human-computer interactive node includes a Local Scheduling Processing Component, an Interface Generating Component and some human-computer interactive interface components.

The architecture of the fault-tolerant scheduling model is shown as Figure 4. This model is made up by a Center Scheduling Component (CSC) and many human-computer interactive nodes. Each human-computer interactive node includes a Local Scheduling Component (LSC), an Interface Generating Component (IGC) and some interface components (ICs). When the system is structured, a LSC, an IGC, and some ICs are disposed on each human-computer interactive node based on the logical function of the node (finished work of the node self) and the fault-tolerant requirement for other nodes (substituting other nodes breaking down).

The scheduling process of ICs is illustrated using example of node 1 as follows. According to requirements, the operator of node 1 sends an ISR to LSC of node 1. If the ISR is LSR and belongs to set ISTnft, the LSC deals with the ISR, and sends the result Interface Forming Information (IFI) to IGC of node 1. The IGC carries out ICs addressing, framework generating and ICs loading. Finally, the display is shown. Otherwise, the ISR is sent to CSC by the LSC. The CSC deals with the ISR, and sends the IFI to the IGC of the node which the ISR specifies. The IGC finishes the work as the IGC of node 1.

In addition, CSC detects the running state of each node. Suppose there is a fault in node 1. According to the load of other nodes, reliability requirement and priority of the running ICs of node 1 at that time, CSC produces one or more scheduling request, which are carried to make other nodes substitute the entire or partial functions of node 1. At the same time, a message is sent to show that there is a fault in node 1, which needs to be maintained. This course is called systemic FT. Moreover, if the operator thinks the scheduling result of above description is unsuitable, he can send ISR to reschedule. This course is called factitious FT.

V. SCHEDULING ALGORITHM

The main algorithms used in the model are introduced in this section.

A. The Algorithm of LSC

The main function of LSC is to receive ISR. If the ISR is a LSR without FT requirements, it was processed by the LSC. Otherwise, it was sent to CSC. The algorithm of LSC is described as algorithm 1.

Algorithm 1

Step 1: receive ISR; /* ISR ∈ IST, IST ∈ ISTS */
Step 2: if (IST ∈ ISTnft) goto Step 3; else goto Step 4;
Step 3: if (IST.DN == IST.SN) goto Step 5; else goto Step 4;
Step 4: send ISR to CSC; goto Step 7;
Step 5: process ISR, and get the IFI;
Step 6: send the IFI to IGC;
Step 7: end.
B. The Algorithm of CSC

The CSC has three functions. First, receiving ISR sent by LSC and pushing it in the appropriate position of ISR queue according to the priority of ISR. Second, processing the request of the front of the queue and sending the IFI to the IGC of DN of ISR. Third, detecting the state for each node and realizing systemic fault-tolerant. If there is a fault in a certain node, according to the fault-tolerant requirements of the node and systemic load, the entire or partial functions of the node are substituted by other nodes. The algorithms of CSC are described as algorithm 2 and algorithm 3.

Algorithm 2
Step1: get the front request, processing to get the IFI;
Step2: if $(IST \in IST_{nf})$ goto Step4;
Step3: get startTime; /*startTime refers the time of the ISR starting to execute*/
\[ \text{if} \ (DL - \text{startTime} \geq ET) \]
\[ \text{send the IFI to IGC of IST.DN}; \]
\[ \text{else goto Step7}; \]
Step5: send the IFI to IGC of IST.DN;
Step6: receive the return (value)of IGC of IST.DN;
\[ \text{if} \ (!\text{value}) \ goto \ Step3; \]
Step7: scheduling success; goto Step8;
Step8: end.

Algorithm 3
Step1: detect Ni; /*Ni \in \text{the set of HCIN}*/
\[ \text{if} \ (!\text{malfunction}) \ \text{loop(Timeout)}; \]
Step2: if(all of ICs running in Ni without FT)
\[ \text{goto Step6}; \]
Step3: CSC selects one or more HCIN, recording DNi. Then, produce FT Information, process to get IFI, and send it to DNi;
Step4: get startTime; /*startTime refers the time of the ISR starting to execute*/
\[ \text{if} \ (DL - \text{startTime} \geq ET) \]
\[ \text{send the IFI to IGC of DNi}; \]
\[ \text{else goto Step7}; \]
Step5: receive the return (value)of IGC of DNi;
\[ \text{if} \ (!\text{value}) \ goto \ Step4; \]
Step6: notice Ni malfunction, and schedule the ICs with FT showing in DNi; goto Step8;
Step7: notice Ni failure, and scheduling failure;
Step8: end;

C. The Algorithm of IGC

The functions of IGC are ICs addressing, frame generating and ICs loading according to the IFI of CSC or LSC sending. After that, send the display result to CSC. The algorithm is described as algorithm 4.

Algorithm 4
Step1: receive the IFI from CSC or LSC;
Step2: addressed ICs according to the IFI;
Step3: generate frame according to the IFI;
Step4: load ICs;
Step5: send the display result to CSC.
Step6: end;

VI. EXPERIMENT ANALYSIS

The architecture of simulation experiment is made up of Information Process System, Application System, Data-Command Agency and Human-computer Interactive System. The software platform is CORBA. Among them, the Information Process System produces and processes the simulation necessary data. The Data-Command Agency manages the data and command uniformly. The architecture of simulation system is shown as Figure 5.

The fault-tolerant scheduling model of the Human-computer interactive subsystem is determined according to the designing result file which is generated recurring to the assistant developing tool. In the model, the CSC cooperates with LSC to accomplish the fault-tolerant scheduling function. The LSC receives ISR, processing it or sending it to CSC according to the fault-tolerant requirement. The CSC receives and processes ISR sent by LSC to generate interface forming information. In the mean time, it detects the states of each node to generate the fault-tolerant information. The fault-tolerant scheduling model is shown as Figure 6.

The Human-computer interactive subsystem configuration is shown as follows in simulation system.

(1) Hardware: PIII800 processor; 256M memory; 10/100M Ethernet.
(2) Software: Linux OS; CORBA;

The results of the experiment are described as follows.

A. The system running normal

The ISR is LST: The scheduling result is correct. The time from the ISR sent to the scheduling finished is 300ms.

Figure 5. The architecture of simulation system

Figure 6. The fault-tolerant scheduling model of the Human-computer interactive subsystem
The ISR is UST: The scheduling result is correct. The time from the ISR sent to the scheduling finished is 500ms.

B. One node malfunction

FT: The FT result is correct. The time from the malfunction happening to the processing finished is 2-2.5s. This time is impacted mainly by the time of the malfunction detecting.

ICs Scheduling: The scheduling result is correct. The scheduling time is the same as (A).

C. Two nodes malfunction

FT: Commonly, the FT result is correct. The time from the malfunction happening to the processing finished is 2-3s. This time is impacted mainly by the time of the malfunction detecting. When there are many ICs with FT requirements in fault nodes, and the priority of the ICs running in other two nodes is more prior, partial functions of the fault nodes are substituted possibly.

ICs Scheduling: The scheduling result is correct. The scheduling time is the same as (A).

VII. CONCLUSION

Three achievements are done in this paper. First, a fault-tolerant scheduling model is presented which can realize interface component tolerant and uniform switching control in distributed human-computer interactive system. Furthermore, it can realize dynamic reconfiguration and upgrading online of interface components. Second, a new method and technique is provided to design human-computer interactive system rapidly. Third, the feasibility of the model and its algorithms are proved by the simulation experiment. There are still some problems remained to study such as the granularity division of the interface components, the mathematic model of effect evaluating of scheduling, the dynamic link and composition of interface components, which will be researched in the future work.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China under Grant No.70671035 and the Key Scientific & Technological Project of Henan Province China under the grant No.082102210076.

REFERENCES


