Vol. 35 No. 5 Transactions of Information Processing Society of Japan May 1994

Regular Paper

Design and Implementation of User-centered Application Software Systems in a Distributed Computing Environment

TETSUO KINOSHITA,[†] KENJI SUGAWARA,[††] MASAHIRO UKIGAI,[††] NORIO SHIRATORI[†††] and NOBUYOSHI MIYAZAKI[†]

This paper proposes a new method for designing and implementing application software systems in a distributed computing environment. In the proposed method, a model for the design of distributed application software systems, called a user-centered application software model, is proposed as a means of designing and implementing heterogeneous software modules for user-centered application software systems. According to the design model, heterogeneous software modules are realized as pluggable software modules according to the users' requirements. Combination of these modules makes it possible to create a user-centered application software system in a distributed computing environment. In addition, this paper gives an example of the design of a user-centered application software system in a local-area-network-based distributed computing environment, and discusses the advantages of the proposed method.

1. Introduction

Distributed computing environments provide users and designers with various facilities for computation and communication. Through the use of these facilities, many kinds of application software system have been developed and utilized in various domains such as information retrieval, scientific/engineering computation, on-line transaction processing, and multi-media communications.[1] Owing to the advances in both users' needs and the technologies of distributed processing, efficient development of various application software systems for a distributed computing environment according to users' requirements has become an important and difficult problem. In general, an application software system consists of many software modules that are executed by various computer systems in a distributed computing environment. Thus, the designers have to develop various software modules by using different programming languages on different computing platforms, combine the heterogeneous software modules, and realize an application software system for users.[13] Moreover, they have to consider the reuse and sharing of scarce or valuable resources in a distributed computing environment. As a result, it is necessary not only to utilize the design knowledge of skilled designers, but also to reduce the development costs of application software systems.[2)-4)]

Recently, several kinds of methods for designing software systems have been proposed with respect to dedicated computer systems[5)-7)] and the object-oriented method has also been introduced and utilized as a useful technique for designing and implementing application systems.[8)-11)] However, a systematic and effective framework for developing distributed application software systems that satisfy the various requirements of users and customers in a target distributed computing environment has not yet been proposed. This paper therefore proposes a new method for designing a distributed application software system according to various users' requirements. Hereafter, a distributed application software system realized in accordance with the proposed method is called a user-centered application software system (UAS).

As elements of the proposed method, we propose a design object model of a UAS, called a user-centered application software model (UASM), and a framework for using and shar-

[†] Systems Laboratories, Oki Electric Industry Co., Ltd.
[††] Department of Computer Science, Chiba Institute of Technology
[†††] Research Institute of Electrical Communication, Tohoku University

ing the distributed resources of a target distributed computing environment, based on a concept of pluggable software modules.[16] According to the users' requirements and design conditions, a design specification of the UAS is defined and represented by using the UASM. In accordance with the design specification, various heterogeneous software modules of the UAS are designed and implemented as pluggable software modules. Finally, the UAS of the target distributed computing environment is realized by harmoniously combining the heterogeneous software modules. Section 2 introduces the UASM and a model for the process of designing a UAS, and proposes a method for designing a UAS. Section 3 gives an example of the design of a UAS in a local-area-network-based (LAN-based) distributed computing environment, using the proposed method. Section 4 gives details of the implementation of the UAS and evaluates the proposed method.

2. Method for the Design of a User-centered Application Software System in a Distributed Computing Environment

2.1 Model for the Design of a Distributed Application Software System

A distributed application software system consists of various heterogeneous software modules running on distributed platforms in a distributed computing environment. A software module has a service function (Fi) consisting of an application task and a plug-in function (PFi), which combines several service functions to deal with a user's request, as shown in **Fig. 1** (a). Since these software modules may be implemented by using different programming languages on different platforms, many expert designers are sometimes needed to develop software modules for various design tasks (Di).

One promising approach may be to utilize software modules or products as components of a distributed application software system, in

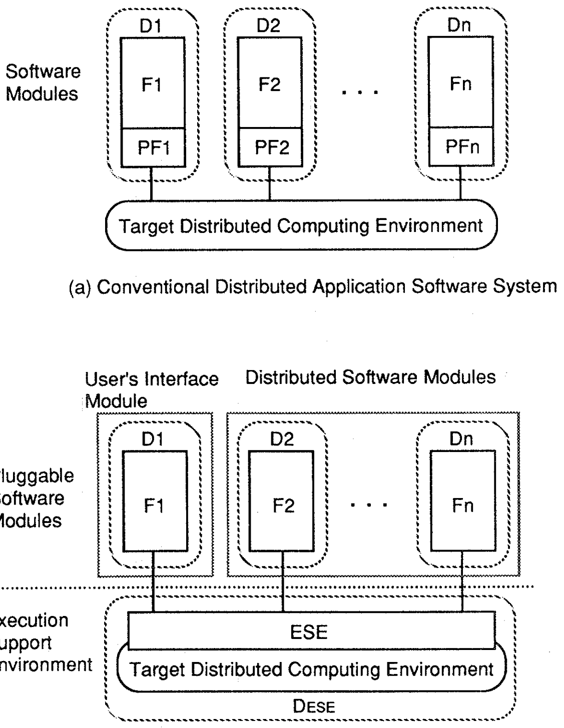

Software Modules
D1
F1
PF1
D2
F2
PF2
. . .
Dn
Fn
PFn
Target Distributed Computing Environment
(a) Conventional Distributed Application Software System
User's Interface Module
Distributed Software Modules
Pluggable Software Modules
D1
F1
D2
F2
. . .
Dn
Fn
Execution Support Environment
ESE
Target Distributed Computing Environment
DESE
(b) User-centered Application Software (UAS) System

Fig. 1 Design model of an application software system in a distributed computing environment.

order to reduce the burden on designers and minimize the development costs of distributed application software systems. However, most existing software modules have been developed in accordance with both a particular application software system, and a particular distributed computing environment. It is therefore difficult to plug existing software modules into a distributed application software system to be designed. Moreover, it is still difficult to design and implement pluggable and reusable heterogeneous software modules for distributed application software systems.[12],[13]

In order to overcome the difficulties of designing and organizing pluggable heterogeneous software modules according to the users' requirements, we propose a model for the design of distributed application software systems, called a user-centered application software system model (UASM), and a process for designing user-centered application software systems on the basis of the UASM.

Within the framework of the proposed design method, the service functions and plug-in functions of distributed software modules are specified uniformly on the basis of the UASM and developed independently as shown in Fig. 1 (b). By decomposing the UASM, the service functions (Fi) are then realized as distributed software modules for various design tasks (Di). One of the software modules provides a user interface environment for a computer terminal, and the others provide service functions plugged into the user interface environment.

To improve the pluggability and reusability of distributed software modules, the design of plug-in functions is separated from the design of distributed software modules, and the plug-in functions are realized as an execution support environment (ESE) for dedicated design tasks (DESE).

Finally, a user-centered application software system (UAS) is constructed in the target distributed computing environment by combining the distributed software modules and the execution support environment.

2.2 User-centered Application Software Model: UASM

From a user's viewpoint, a distributed application software system can be modeled as a set of software modules of two types, namely, internal software modules and external software modules. An internal software module runs on a user terminal and provides user-oriented facilities such as user interface control, small-scale computation, and local file management. On the other hand, an external software module runs on a distributed platform and provides service functions such as application-oriented functions, large-scale computation, and data base facilities.

According to the above modeling, a user-centered application software model (UASM) is defined as a design object model that represents a design specification of a UAS in a target distributed computing environment. By means of the UASM,

1. the external software modules of the target UAS are designed and implemented as pluggable software modules, and
2. existing software modules and products can be selected and assigned as external software modules of the target UAS.

Formally, the UASM is defined as follows:

UASM=⟨IPG, EPG⟩.

IPG={ipg|ipg=⟨IPN, IPA⟩}, where
IPN={ipn|ipn∈IFN∪VFN},
IPA={ipa|ipa=⟨ipa-id, ipn_i, ipn_j⟩;ipn_i, ipn_j∈IPN},
IFN={ifn|ifn=⟨ifn-id, if-c, if-in, if-out, if-cond⟩},
VFN={vfn|vfn=⟨vfn-id, cfn-id, vf-in, vf-out, vf-cond⟩; cfn-id∈CFN∈EPG}.

EPG={epg|epg=⟨EPN, EPA⟩}, where
EPN={epn|epn∈EFN∪CFN},
EPA={epa|epa=⟨epa-id, c-typ, cfn, efn⟩; cfn∈CFN, efn∈EFN, c-typ∈TYPE},
EFN={efn|efn=⟨efn-id, ef-c, ef-in, ef-out, ef-cond, ef-ipg⟩},
CFN={cfn|cfn=⟨cfn-id, vfn-id, {efn-id}, vf-in, vf-out, {ef-in}, {ef-out}, cfn-cond⟩; ef-id, ef-in, ef-out∈efn∈EFN}.

In the UASM, the logical processing structure of both internal software modules and external software modules are represented by sets of process graphs of two kinds: a set of internal process graphs (IPG) and a set of external process graphs (EPG).

An internal process graph (ipg) consists of nodes of two kinds: a set of internal function nodes (IFN) and a set of virtual function nodes (VFN), and an external process graph (epg) also consists of nodes of two kinds: a set of communication function nodes (CFN) and a set of external function nodes (EFN), as shown in **Fig. 2**. The ipg and epg are constructed by using a set of internal process arcs (IPA) and a set of external process arcs (EPA), respectively.

An internal function node (ifn) represents a function or task of an internal software module. The ifn is specified by an identifier (ifn-id), a function class (if-c), input (if-in), output (if-out), and conditions (if-cond).

An if-c specifies a class selected from the tree-structured function classes that provide the background knowledge for describing the design specifications of internal software modules. The background knowledge is organized as a set of standardized functions called generic functions.[18]

An if-in and an if-out are specified by the attributes of data objects of the ifn, such as the name (identifier), type, format, and volume, while, an if-cond specifies the processing conditions, such as the processing time margin, location of computing platforms, and auxiliary resources. According to the specifications of if-in, if-out, and if-cond, a detailed if-c is selected or defined in a design process of the target UAS.

On the other hand, an external function node

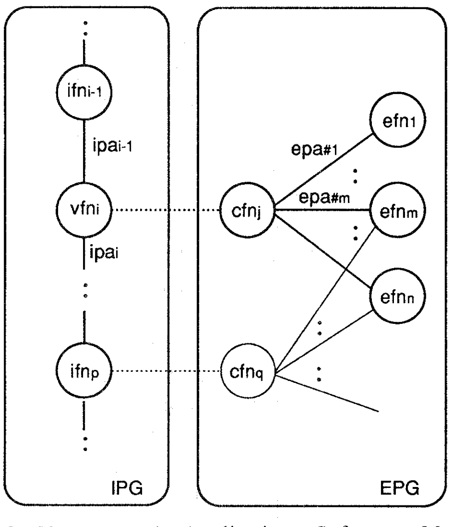


Fig. 2 User-centered Application Software Model (UASM).

(efn) represents an external software module running on a distributed computing resource. The following elements are defined in the same way as for the inf: identifier (efn-id), function class (ef-c), input (ef-in), output (ef-out), and conditions (ef-cond). Note that an ef-c is selected from a set of predefined function classes that provide the background knowledge for specifying the service functions of external software modules. Moreover, if an external function consists of a set of external software modules and the design specification of the external function is represented recursively by a process graph, then a pointer to the process graph is specified by an ef-ipg.

To allow an external software module to be designed as a pluggable software module of an internal software module, a virtual function node (vfn∈VFN∈IPG) and a communication function node (cfn ∈ CFN ∈ EPG) are introduced into the UASM. According to the design specifications of both CFN and VFN, an execution support environment of the target UAS is implemented.

A virtual function node (vfn) specifies a function that accesses a service function provided by an external software module. Although a vfn is defined in the same way as an ifn of an ipg, its function is specified by the identifier of the corresponding cfn (cfn-id) of an epg.

On the other hand, a communication function node (cfn) specifies a function that plugs in external software modules to an internal software module specified by a vfn. A cfn is defined by an identifier (cfn-id), a vfn's identifier (vfn-id), a set of efns' identifiers ({efn-id}), the input and output of a vfn (vf-in and vf-out), the efns' input and output ({ef-in} and {ef-out}), and processing conditions (cfn-cond). The cfn may be connected to one or more external function nodes by using external process arcs (epa∈ EPA). Hence, each epa has a control type (c-typ) selected from a set of control types (TYPE) that specify the control scheme of external software modules. For example, the parallel execution control of external software modules is specified by a c-typ named "PAR."

By means of the UASM, (1) heterogeneous software modules can be specified homogeneously, (2) harmonic combinations of heterogeneous software modules can be specified explicitly, (3) pluggable software modules can be designed systematically, and (4) cooperative or collaborative development of distributed software modules can easily be carried out by expert designers using various programming languages on distributed platforms.

2.3 Process of the Design of User-centered Application Software Systems

The proposed method for the design of user-centered application software systems (UASs) is given by a design process model defined in accordance with a generic design process model of the knowledge-based design methodology (KDM).[2] As shown in **Fig. 3**, the proposed design process model of the UAS consists of the following four design phases: (1) a requirement definition phase, (2) a conceptual design phase, (3) a detailed design phase, and (4) a prototyping phase. Designers define and use a dedicated process for the design of the target UAS by applying the design process model to the design of a UAS in a target distributed environment.

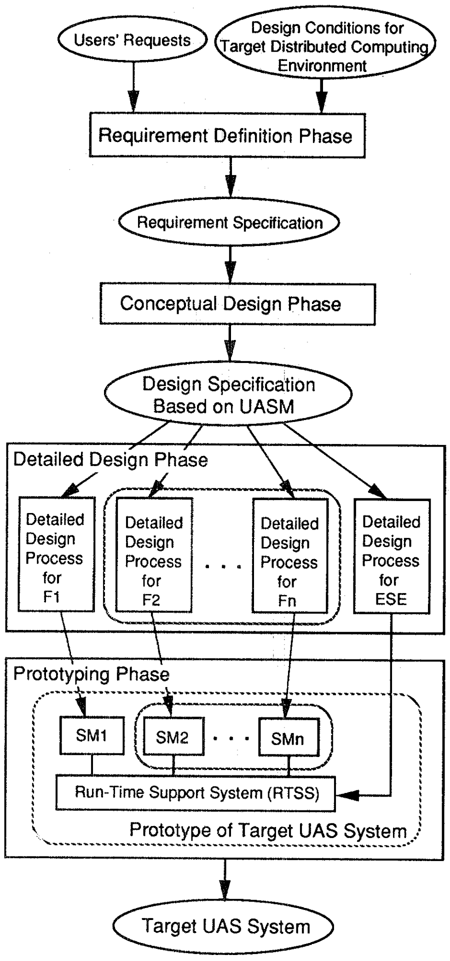

Users' Requests
Design Conditions for Target Distributed Computing Environment
Requirement Definition Phase
Requirement Specification
Conceptual Design Phase
Design Specification Based on UASM
Detailed Design Phase
Detailed Design Process for F1
Detailed Design Process for F2
. . .
Detailed Design Process for Fn
Detailed Design Process for ESE
Prototyping Phase
SM1
SM2
. . .
SMn
Run-Time Support System (RTSS)
Prototype of Target UAS System
Target UAS System

Fig. 3 Design process for user-centered application software systems.

In the requirement definition phase, a requirement specification of the target UAS is defined in accordance with users' requests and the design conditions of the target distributed computing environment. The requirement specification can be represented by using the existing models and methods for requirement specification, such as a requirement graph model, object-oriented specification models and methods, knowledge-based requirement definition methods.[20]

In the conceptual design phase, the requirement specification of the target UAS is transformed into a design specification. According to the UASM, the design specification is defined and represented by the designers. Within the framework of the KDM, the designers can also access and utilize two kinds of knowledge, namely, a set of design rules called mappings and background knowledge. The mappings provide knowledge for manipulating and transforming the requirement specification, and the background knowledge also provides knowledge referred by the mappings. The derived description of the design specification is called an instantiated UASM of the target UAS.

In the detailed design phase, the design specification, that is, the instantiated UASM, is decomposed into several parts consisting of the design specifications for the service functions and an execution support environment of the target UAS. Heterogeneous software modules are realized by respective design processes corresponding to design specifications. The decomposition of the design specification and the detailed design of heterogeneous software modules are done by designers using the design knowledge given by a set of mappings and the background knowledge, as in the conceptual design phase.

Finally, in the prototyping phase, designers code and implement heterogeneous software modules and realize a prototype of the target UAS. In the prototype system, the service functions are realized as a set of pluggable software modules (SMi), and the execution support environment is also realized as a run-time support system (RTSS), as shown in Fig. 3. Since the logical combinations of heterogeneous software modules are given by the instantiated UASM, the target UAS can be constructed systematically. The validation and testing are also done by using this prototype system.

3. Design of User-centered Application Software in a LAN-based Distributed Computing Environment

3.1 Overview of the Target Distributed Computing Environment

A LAN-based distributed environment is selected as the target environment for application of the proposed design method. A brief overview of the target environment is now given.

The target distributed environment consists of several platforms such as user workstations, and distributed computing resources such as a transputer system, a file server, and a print server. A user workstation provides a user interface environment and the distributed platforms also provide various facilities such as parallel computation and file management. These platforms are connected by an Ethernet-LAN, as shown in **Fig. 4**. In the target environment, many programming languages are provided for users and designers. For instance, a user workstation provides general purpose programming languages such as C and Smalltalk-80,[14] and a transputer system also provides a parallel programming language, OCCAM2.[15]

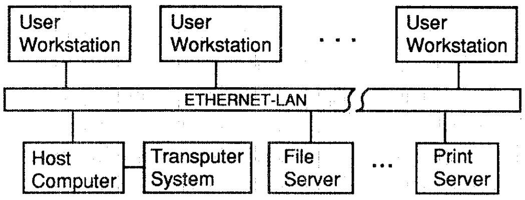


Fig. 4 Target Distributed Computing Environment (DCE).

3.2 Process for the Design of a Target User-centered Application Software System

We now explain the process for the design of a target UAS, focussing on the detailed design phase and the prototyping phase.

A dedicated design process is defined by applying the design process model in Fig. 3 to the design of the target UAS, as shown in **Fig. 5**. Design knowledge and background knowledge can be acquired, represented, and stored in the knowledge bases, namely, the ISM-Design-KB, ESM-Design-KB, and CSM-Design-KB, respectively. Since the management facilities of the knowledge bases are now being developed, the design knowledge for our experimental development of the target UAS has been used without the knowledge bases.

As explained in section 2.3, the dedicated

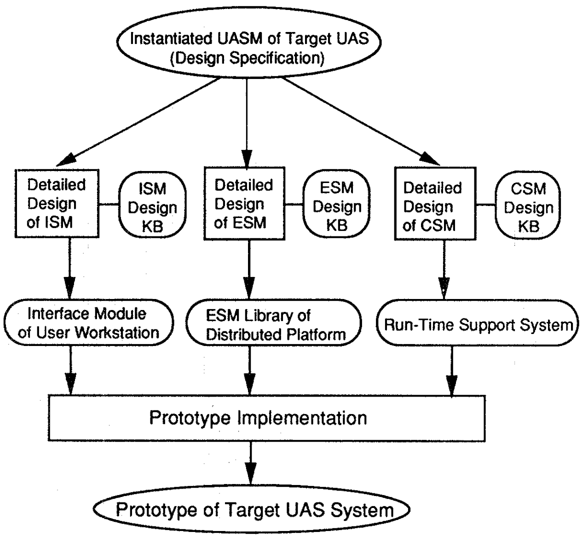

Instantiated UASM of Target UAS (Design Specification)
Detailed Design of ISM
ISM Design KB
Detailed Design of ESM
ESM Design KB
Detailed Design of CSM
CSM Design KB
Interface Module of User Workstation
ESM Library of Distributed Platform
Run-Time Support System
Prototype Implementation
Prototype of Target UAS System

Fig. 5 Process for the dedicated design of the target UAS in a target DCE.

design process of the target UAS has been formalized as a set of separate design processes in which an upstream design object, that is, an instantiated UASM, is decomposed and transformed into downstream design objects, namely, internal software modules (ISMs), external software modules (ESMs), and communication software modules (CSMs). The ISMs correspond to the software modules of a user interface environment on a user workstation. The ESMs correspond to the application software modules on distributed platforms such as transputer systems. These ESMs are organized and managed as a data base called the ESM Library. The CSMs correspond to the software modules of a run-time support system of the target UAS.

In the detailed design of various software modules, designers can select and use suitable design methods and tools, such as object-oriented design methods,[8] and various CASE tools.[22] Knowledge-based design methods based on KDM can also be applied; for instance, a knowledge-based interface design method (KDM/UI)[18] can be used to develop a user interface system, and a knowledge-based protocol design method (KDM/P)[19] can be used to design a special purpose communication protocol for communication software modules.

In the prototyping of the target UAS, the user interface modules, application software modules, and communication software modules are cooperatively implemented as heterogeneous software modules on distributed platforms by their respective designers. A prototype system of the target UAS is then generated by combining these software modules in accordance with the instantiated UASM.

3.3 Design Specification of the Target UAS

As an example, a UAS that generates and displays computer graphics of fractals[21] has been designed and implemented in the target distributed environment. A user requests a transputer system to compute a fractal set and display it graphically in a window of a user workstation. Therefore, the target UAS has to provide the following functions: (1) setting up parameters for computing a fractal set, (2) generating fractal graphics by using a transputer system, and (3) displaying the graphics on a user workstation.

A design specification of these functions has been defined and represented as an instantiated UASM. A part of the descriptions of internal process graphs and external process graphs of the instantiated UASM is shown in **Fig. 6**. Although a specification description language is now being developed, the instantiated UASM has been formally represented by using a frame-based knowledge representation language,[17] as shown in **Fig. 7**.

Given the instantiated UASM, the functions of both setting up parameters and displaying fractal graphics, are specified as internal soft-

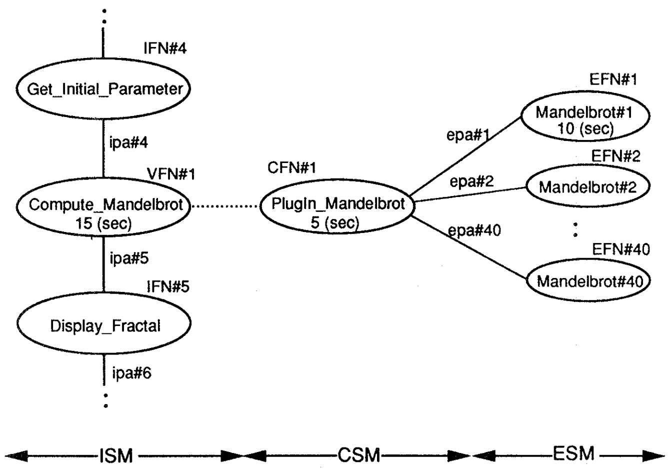


Fig. 6 Part of the design specification of the target UAS (Instantiated UASM).

```
IFN10004     F
    SUP      spo    IFN
    IF-ID    v      IFN0004
    PRE-N    f      IFN10003
    PST-N    f      VFN10001
    IF-C     f      Get-Data
    IF-IN    ps     ("Prompt-Txt",
                     Character,
                     Text,
                     36)
    IF-OUT   f      D10011
    TAT      ps     (nil, s)
    P-LOC    f      USER-330
    AUX-F    f      nil


VFN10001     F
    SUP      spo    VFN
    VF-ID    v      VFN0001
    PRE-N    f      IFN10004
    PST-N    f      IFN10005
    CFN-ID   f      CFN10001
    F-CLS    v      "Fractal-CG"
    VF-IN    f      D10011
    VF-OUT   f      D10012
    TAT      ps     (15, s)
    SEF      f      ERR-Demon-1
    P-LOC    f      USER-330
    AUX-F    f      nil


CFN10001     F
    SUP      spo    CFN
    CF-ID    v      CFN0001
    VF-ID    f      VFN10001
    EF-ID    ps     (EFN11001, PAR) &
                    (EFN11002, PAR) & ...
                    (EFN11040, PAR)
    VF-IN    f      D10011
    VF-OUT   f      D10012
    EF-IN    ps     (D10011, EF-ID, P-S)
                    : ncc (Broadcast-D)
    EF-OUT   ps     (D20001, EF-ID, P-AS)
                    : ncc (Make-Array)
    TAT      ps     (5, s)
    CNT-T    v      1-N
    SEF      f      ERR-Demon-101
    P-LOC    f      USER-330
    AUX-F    f      nil
```

```
D10011       F
    SUP      spo    D-FORM
    D-NAM    v      "Parameter-list"
    D-TYP    f      Real-Num
    D-FORM   f      list
    D-VOL    v      10


D10012       F
    SUP      spo    D-FORM
    D-NAM    v      "F-CG-Data"
    D-TYP    f      Real-Num
    D-FORM   f      Array
    D-VOL    v      16384


D20001       F
    SUP      spo    D-FORM
    D-NAM    v      "Part-CG-Data"
    D-TYP    f      Real-Num
    D-FORM   f      list
    D-VOL    v      200


EFN11001     F
    SUP      spo    EFN
    EF-ID    v      EFN1001
    EF-C     f      Mandelbrot-F
    PRE      f      nil
    PST      f      nil
    CFN-ID   f      CFN10001
    EF-IN    f      D10011
    EF-OUT   f      D20001
    EF-IPG   f      DEFN11001
    TAT      ps     (10, s)
    SEF      f      ERR-Demon-21
    P-LOC    f      CIT-TP
    AUX-F    f      nil
```

Fig. 7 Example of frame descriptions of the design specification of the target UAS.

ware modules (ISMs). For example, a virtual function node named "Compute_Mandelbrot" specifies that the input (vf-in) is a set of numerical parameters of a fractal function, the output (vf-out) is a set of numerical data of fractal graphics, and the processing time margin in a vf-cond is set to 15 seconds. Furthermore, the function for computing fractal set is specified as a set of pluggable external software modules (ESMs) running on a transputer system. For instance, an efn named "Mandelbrot # 1" specifies that its function class (ef-c) is a fractal function class named "Mandelbrot," the input (ef-in) is a set of parameters given by a vfn named "Compute_Mandelbrot," an output (ef-out) is a partial data set of fractal graphics, and the processing time margin in ef-cond is set to 10 seconds.

A communication function node (cfn) is also specified, to connect the ESMs on a transputer system with the ISMs on a user workstation. For example, a cfn named "PlugIn_Mandelbrot" specifies that the input data of "Compute_Mandelbrot" are broadcast as the input data of the ESMs, that is, "Mandelbrot # 1" and so on, and that the output of the ESMs are unified in the form of the output of "Compute_Mandelbrot." The processing time margin (cfn-cond) is set to 5 seconds. Moreover, each external processing arc (epa) has a control type (c-typ) named "PAR" to activate and execute respective ESMs in parallel.

3.4 Design of ISMs for User Workstations

According to the instantiated UASM, an internal software module (ISM) is developed by a designer of a user interface environment. In the target UAS, two kinds of ISM, that is, functional ISMs (F-ISMs) and virtual ISMs (V-ISMs), are designed and implemented as a set of class objects of Smalltalk-80, as follows.

An F-ISM is realized as a class object with respect to an internal function node (ifn). For instance, if a function class (if-c) of "Get_Initial_Parameter" in Fig. 6 is specified by "get_value," which receives data input by a user on a keyboard, then an F-ISM is defined as a subclass named "Get_initial_parameter" as a class named "FillInTheBlank" of Smalltalk-80. Because of the design specification of input (if-in), output (if-out), and condition (if-cond),

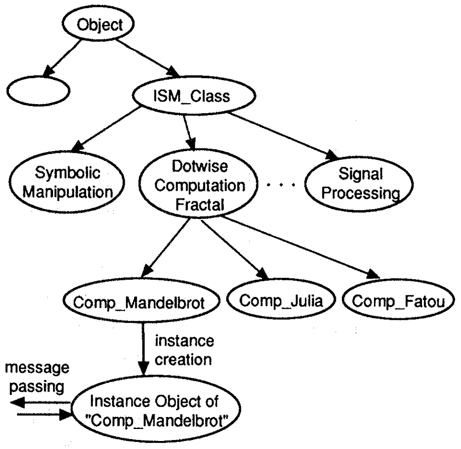


Fig. 8 ISM_Class of the target UAS.

the class methods of "Get_initial_parameter" are then defined to accept an initial parameter of a fractal function from the keyboard. By using Smalltalk-80's existing classes, various F-ISMs can be incrementally defined and attached to provide a user-specific interface environment.

On the other hand, a V-ISM is realized as a class object in a special class called ISM_Class. According to the user's requests, many kinds of V-ISM such as "DotwiseComputationFractal," "SignalProcessing," and "SymbolicManipulation" can be defined as shown in **Fig. 8**. Moreover, in the target UAS, several fractal computation functions such as "Mandelbrot," "JuliaFractal," and "FatouFractal" are implemented as ESMs. Hence, for instance, by means of a vfn named "Compute_Mandelbrot" in Fig. 6, a class object named "Comp_Mandelbrot" is defined as a sub-class of "DotwiseComputationFractal," as shown in Fig. 8, in order to utilize the fractal computing function "Mandelbrot." Once the instance objects of V-ISMs have been created during UAS run-time, the V-ISMs can be accessed in the same way as the F-ISMs.

3.5 Design of ESMs for Transputer Systems

A parallel computing program in OCCAM2 for a transputer system is a typical example of the design of external software modules for a target UAS.

In accordance with the design specification regarding the computation of fractals, the external software modules (ESMs) are designed and implemented as parallel software modules of the transputer system by a skilled OCCAM2 programmer. Since the transputer system in the target environment consists of a transputer array with 40 transputers, the OCCAM2 programmer also has to design the structure for assignment of the ESMs over the transputer array.

The ESMs are designed and implemented as load modules and stored in the ESM Library, which is allocated to a host computer of the transputer system. For instance, forty parallel software modules are realized as load modules of ESMs for the parallel computation of a Mandelbrot-set.

According to a request issued by an ISM, a set of ESMs is extracted from the ESM Library and loaded onto the transputer array by a root transputer. The plug-in function that connects these ESMs with the ISM is realized as a set of communication software modules (CSMs) of the target UAS, as explained in section 3.6.

Various parallel software modules can be developed and stored in the ESM Library in advance. As explained in section 3.4, in our target UAS, several fractal functions for Mandelbrot, JuliaFractal, and so on, have been developed and stored into the ESM Library. Thus, if a set of parallel software modules which satisfies the instantiated UASM is stored in the ESM Library, then the designer only selects and assigns these modules as ESMs of the target UAS.

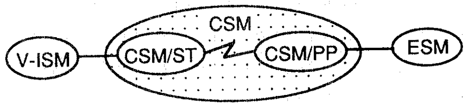


Fig. 9 Detailed structure of a Communication Software Module (CSM).

3.6 Design of CSMs

In this section, an example of the design of a communication software module (CSM) that manages the communication between a virtual software module (V-ISM) of a user workstation and a set of external software modules (ESMs) of a transputer system is explained.

By means of a communication function node (cfn) of the instantiated UASM, the following functions of the CSM can be designed: (f 1) setting up the pluggable ESMs, (f 2) activating the ESMs, (f 3) controlling the execution of ESMs, (f 4) exchanging data with the V-ISM, and (f 5) exchanging data with the ESMs.

In the target UAS, the CSM is realized by two kinds of software module: (1) CSM/STs, which

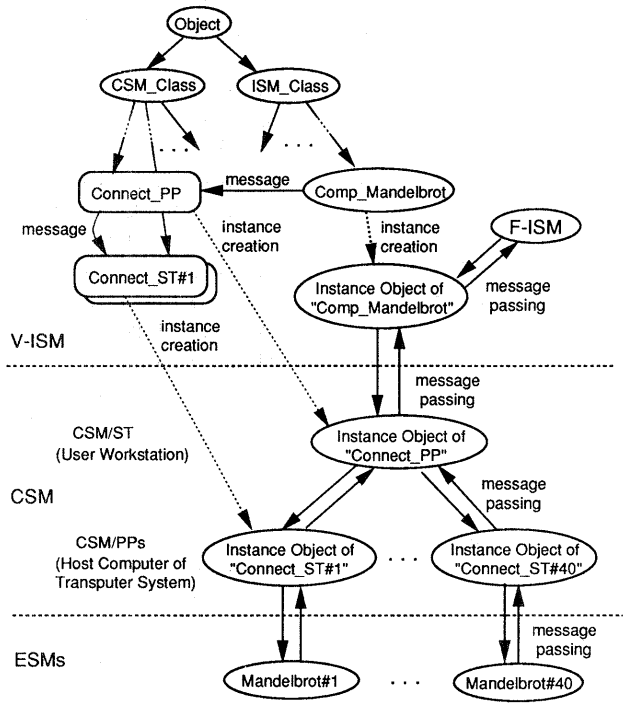


Fig. 10 Detailed design of a CSM/ST and a CSM/PP.

provide functions f 1, f 3 and f 4, and (2) CSM/PPs, which provide functions f 2 and f 5, as shown in **Fig. 9**.

For instance, a cfn named "PlugIn_Mandelbrot" in Fig. 6 is realized by a CMS/ST running on the user workstation and a set of CSM/PPs running on a host computer of the transputer system, for the parallel computation of a Mandelbrot-set. As shown in **Fig. 10**, the CSM/ST is realized as a class object named "Connect_PP," and forty CSM/PPs are also realized as a set of class objects such as "Connect_ST # 1." On the other hand, the functions for exchanging data and messages between a V-ISM ("Comp_Mandelbrot") and the ESMs ("Mandelbrot # 1" and so on) are realized by the class methods of both CSM/ST and CSM/PPs.

4. Implementation and Evaluation of a User-centered Application Software System

4.1 Prototype of a User-centered Application Software System

The configuration of a prototype of the target UAS is shown in **Fig. 11**. In order to support the design and implementation of the target UAS, a design support facility called UAS-Browser was also developed and provided to the designers. By using the UAS-Browser, the designers can efficiently retrieve, inspect, select, and modify the design specifications of the ISMs, ESMs, and CSMs.

The CSMs have been organized as a run-time support system (RTSS) of the target distributed environment. As explained in section 3.6, the RTSS consists of dedicated class objects of both CSM/STs and CSM/PPs that run on a user workstation and a host computer of the transputer system, respectively.

During a run-time of the target UAS, the ESMs are plugged into the ISM by using the RTSS. For instance, as shown in Figs. 10 and 11, first, when a V-ISM named "Comp_Mandelbrot" receives a create message sent from an F-ISM, an instance of "Comp_Mandelbrot" is created. Next, an instance object of a CSM/ST named "Connect_PP" and forty instance objects of the CSM/PPs, namely, "Connect_ST # 1" and so on, are also created as an RTSS of the target UAS. Then, the RTSS sends a set-up message to a host computer to extract a set of ESMs from the ESM Library. After that, the load modules of ESMs such as "Mandelbrot # 1" are loaded onto the transputer array and the transputer system is initialized. Finally, the connections between the V-ISM and the ESMs are established and computation of a Mandelbrot-set will start.

By means of the above mechanisms in our target UAS, the user can freely and quickly generate and display various graphics of fractals in the windows of a workstation, using the parallel computation capability of the transputer system.

4.2 Evaluation of the User-centered Application Software System

Use of a prototype, the target UAS has confirmed the following advantages of the proposed design method:

1. Heterogeneous software modules can eas-

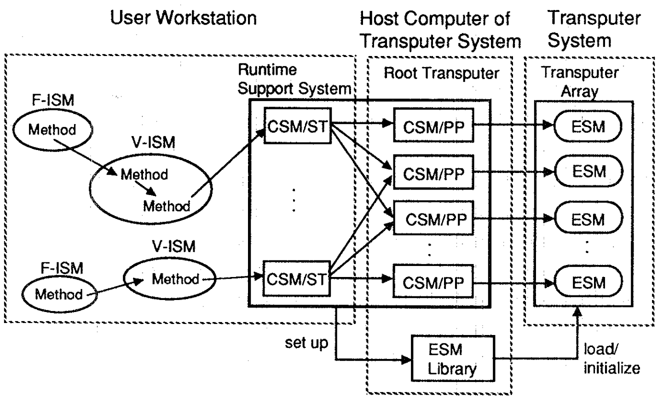


Fig. 11 Configuration of the target UAS.

ily be designed and implemented. According to a design specification given by an instantiated UASM, the design and implementation of heterogeneous software modules can be done separately and cooperatively by designers and programmers of distributed platforms. In our experiments, a Smalltalk-80 programmer can concentrate on the design of user interface environments for displaying fractal graphics. An OCCAM2 programmer can also concentrate on the design of parallel software modules for computing fractals. As a result, rapid prototyping was done on the basis of cooperative work by a Smalltalk-80 programmer and an OCCAM2 programmer, and the productivity of the UAS was raised two or three times with respect to the development time, as compared with a design and development of the UAS done by an application designer.

2. Heterogeneous software modules can be harmoniously combined. By designing V-ISMs and CSMs based on the instantiated UASM, a harmonious combination of the ESMs in OCCAM2 and the ISM in Smalltalk-80 can easily be attained in the target UAS. Moreover, by using a run-time support system, dynamic linkage of ISMs and ESMs can be done systematically. As a result, the usability of distributed computing resources such as a transputer system can be increased, and the development costs of distributed application software systems can also be reduced.

3. An efficient distributed application software system can be developed. Through the use of dedicated heterogeneous software modules running on distributed computing resources, a high-performance UAS has been realized. Usu ally, the problem in generating computer graphics of fractals is the large amount of processing time required. For instance, an application program coded only in Smalltalk-80 takes about 60 minutes to compute a Mandelbrot-set and display the fractal graphics. On the other hand, by using a prototype of the target UAS with a transputer system, the execution terminated in 18 seconds, of which the ESMs used about 7 seconds to calculate a Mandelbrot-set, the ISM used about 7 seconds to display the graphics of the Mandelbrot-set, and the CSMs used about 4 seconds to exchange data among ISMs and ESMs. As a result, the design specification of the target UAS, which required a maximum of 20 seconds for processing this task, has fully been satisfied, and the turnaround time for obtaining fractal graphics has been remarkably improved.

Through the experimental design of a UAS in a target distributed computing environment, it has been confirmed that the proposed design method is useful for systematic development of efficient user-centered application software systems. However, many problems remain to be solved, such as refinement of the UASM, design of a requirements specification description language based on the UASM, acquisition of a variety of knowledge to improve the design of the UAS, and development of a knowledge-based design support system for the UAS.

5. Conclusion

A new method for the design of distributed application software systems, using the framework of the knowledge-based design methodology, has been proposed and used to design and implement a distributed application software system with various software components in a distributed computing environment. In the proposed method, a design object model, called a user-centered application software model, is proposed to allow the formalization and representation of the design specifications of heterogeneous software modules of a distributed application software system. A design process model of a user-centered application software system is also proposed to facilitate the design of various software components as pluggable software modules running on distributed platforms. On the basis of the proposed method, a design example of a user-centered application software system was demonstrated by using a LAN-based distributed computing environment. In the design example, two kinds of heterogeneous software module, namely, a user interface module realized in Smalltalk-80 on a user workstation and parallel application modules realized in OCCAM2 on a transputer system, were cooperatively designed and harmoniously combined. As a result, an efficient user-centered application software system was realized. The advantages of the proposed design method were also been confirmed.

References

1) Coulouris, G. and Dollimore, J.: *Distributed Computing Systems: Concepts and Design*, Addison-Wesley, Reading, Mass. (1988).
2) Kinoshita, T., Sugawara, K. and Shiratori, N.: Knowledge-based Design Support System for Computer Communication System, *IEEE Journal on Selected Areas in Communications*, Vol. 6, No. 5, pp. 850-861 (1988).
3) Shiratori, N., Takahashi, K., Sugawara, K. and Kinoshita, T.: Using Artificial Intelligence in Communication System Design, *IEEE Software*, Vol. 9, No. 1, pp. 38-46 (1992).
4) Nii, N. P., Aiello, H., Bhansali, S., Guidon, R. and Peyton, L.: Knowledge Assisted Software Engineering (KASE): An Introduction and Status-June 1991, Technical Report No. KSL 91-28, Knowledge Systems Laboratory, Dept. of Computer Science, Stanford Univ. (1991).
5) Wu, M. Y. and Gajski, D. D.: Hypertool: A Programming Aid for Message-passing Systems, *IEEE Trans. on Parallel and Distributed Systems*, Vol. 1, No. 3, pp. 330-343 (1990).
6) Gabber, E.: VMMP: A Practical Tool for the Development of Portable and Efficient Programs for Multiprocessors, *IEEE Trans. on Parallel Distributed Systems*, Vol. 1, No. 3, pp. 304-317 (1990).
7) Marzullo, K., Cooper, R., Wood, M. D. and Birman, K. P.: Tools for Distributed Application Management, *IEEE Computer*, Vol. 24, No. 8, pp. 42-51 (1991).
8) Coad, P. and Yourdon, E.: *Object-Oriented Analysis*, Prentice-Hall (1988).
9) Sernadas, C. and Fiadeiro, J.: *Towards Object-Oriented Conceptual Modeling*, Chen, P. P. (ed.), Data & Knowledge Engineering 6, pp. 479-508, North-Holland (1991).
10) Lewis, J. A., Henry, A. M., Kafura, D. G. and Schulman, R. S.: An Empirical Study of the Object-oriented Paradigm and Software Reuse, *Rroc. OOPSLA '91*, pp. 184-196, ACM (1991).
11) Yau, S. S., ~~Jia, X. and~~ Bae, D-H.: Software Design Methods for Distributed Computing Environments, *Computer Communications*, Vol. 15, No. 4, pp. 213-224, Butterworth (1992).
12) Notkin, D., Hutchinson, N., Sanislo, J. and Schwartz, M.: Heterogeneous Computing Environments: Report on the ACM SIGSOPS Workshop on Accommodating Heterogeneity, *CACM*, Vol. 30, No. 2, pp. 132-140 (1987).
13) Notkin, D., Black, A. P., Lazowska, E. D., Levy, H. M., Sanislo, J. and Zahorjan, N.: Interconnecting Heterogeneous Computing Systems, *CACM*, Vol. 31, No. 3, pp. 258-273 (1988).
14) Goldberg, A. and Robson, D.: *Smalltalk-80: The Language and Its Implementation*, Addison Wesley (1983).
15) May, D. and Shepherd, R.: *Communicating Process Computers*, Inmos Tech. Note 22, Inmos Ltd. (1987).
16) Fukushima, M., Ukigai, M., Sugawara, K. and Miida, Y.: Pluggable Parallel Processing Modules in a Distributed Processing Environment, *Proc. ISMM Inter. Symp. on Computer Applications in Design, Simulation and Analysis, International Society for Mini and Micro Computers*, pp. 242-245 (1991).
17) Kinoshita, T. and Nakazawa, O.: Experimental Knowledge Base System based on the Frame Model, *Oki Tech. Rev.*, Vol. 52, OKI Elec., pp. 1-8 (1985).
18) Kinoshita, T., Iwane, N., Sugawara, K. and Shiratori, N.: Formalization of the Interface Design Method and Construction of the Design Support System based on the Knowledge-based Design Methodology, *Trans. IPSJ*, Vol. 31, No. 6, pp. 906-915 (1990).
19) Kinoshita, T., Sugawara, K. and Shiratori, N.: Knowledge-based Protocol Design for Computer Communication Systems, *IEICE Trans. Inf. & Syst.*, Vol. E75-D, No. 1, pp. 156-169 (1992).
20) Kinoshita, T., Sugawara, K. and Shiratori, N.: Knowledge-based Requirements Specification and Definition Method for Computer Communication System Design, *IEICE Trans. Fundamentals*, Vol. J76-A, No. 3, pp. 528-539 (1993).
21) Mandelbrot, B. B.: *The Fractal Geometry of Nature*, V. H. Freeman and Company (1977).
22) Norman, R. J. and Forte, G. (eds.): CASE in the '90s, *Commn. ACM*, Vol. 35, No. 4, (Special Section), Apr. (1992).

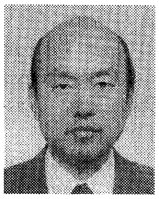
(Received November 26, 1992)
(Accepted December 9, 1993)

Tetsuo Kinoshita is a research manager of the systems laboratories, Oki Electric Industry Co., Ltd., Tokyo. He received a B.E. degree in electronic engineering from Ibaraki University, Japan, and the M.E. and Dr. Eng. degrees in information engineering from Tohoku University, Japan. His research interests include knowledge representation model, knowledge-based system, cooperative distributed processing system and human interfaces. He received the IPSJ Research Award in 1989. Dr. Kinoshita is a member of the IPSJ, IEICE, JSAI, Society for Cognitive Science of Japan, Association for Computational Linguistics and AAAI.

Kenji Sugawara received the B.E., M.E. and Dr. Eng. degrees in electrical engineering from Tohoku University, Sendai, Japan, in 1973, 1977, and 1983, respectively. He is now a professor in the department of computer science, Chiba Institute of Technology, Chiba, Japan. His research interests are in the area of computer communication networks, knowledge engineering, distributed artificial intelligence, CAI, and human interfaces. Prof. Sugawara is a member of the IEEE, IEICE, IPSJ, JSAI, Society for Cognitive Science of Japan, and Robotic Society of Japan.

Masahiro Ukigai is an associate professor in the department of computer science, Chiba Institute of Technology, Chiba, Japan. He received a B.E. in electronic engineering from Chiba Institute of Technology in 1978. He received a Dr. Eng. degree in electrical engineering from Keio University, Japan, in 1986. His research interests include advanced technology on computers in education and parallel processing applications. Prof. Ukigai is a member of the IPSJ, Japan Society for CAI, the Acoustical Society of Japan, IEICE and IEEE.

Norio Shiratori After receiving his doctorate degree at the Tohoku University, Dr. Shiratori joined the research institute of electrical communication (RIEC) of Tohoku University in 1977, and is now a professor of the RIEC. He has been engaged in research on an intelligent distributed processing system, based on a computer communications network, including software design. He is also working on human interfaces, CSCW and advanced intelligent networks. Prof. Shiratori received the 25th Anniversary IPSJ Memorial Prize-Winning Paper Award in 1985 and the 6th Telecommunications Advancement Foundation Incorporation Award in 1991. Prof. Shiratori is a member of the IEEE, IEICE, IPSJ and JSAI.



Nobuyoshi Miyazaki received the B.S. from Kyoto University in 1973, the M.S. in Computer Science from University of Illinois at Urbana-Champaign in 1979, and Dr. Eng. from Kyoto University in 1990. He has been working for Oki Electric Industry Co., Ltd. since 1973 and is currently a department manager at Media Laboratory. He was a senior researcher at Institute for New Generation Computer Technology from 1982 to 1985. His research interests include databases and knowledge information processing. He is a member of IPSJ, IEICE, JSAI, IEEE/CS and ACM.