IKUMI: A Groupware Development Support System with Visual Environment

Hiroyuki Tarumi, Atsushi Tabuchi, Kenji Yoshifu *

Kansai C&C Research Labs., NEC Corporation

Abstract. IKUMI is a groupware development support system, which is based on MEGUMI, the e-mail platform, developed by the authors. IKUMI provides the workflow feature, including body-less mail for synchronizing activities (beacon messages), visually defining of branching conditions, and cooperation between e-mail and realtime groupware. With IKUMI, end-users can easily and dynamically define relatively simple cooperative work with workflow, and system engineers are able to embed groupware functions into applications at low cost.

1 Introduction

Recent CSCW technologies have produced many kinds of groupware systems — *general communication tools*, including teleconference systems like MERMAID[1], e-mail, and bulletin board systems, and *task-oriented applications*, including co-authoring systems like Quilt[2] and decision support systems like gIBIS[3].

Task-oriented groupware applications are specifically designed only for particular tasks, e.g. co-authoring or decision making. Application developers, however, may want to build groupware applications for other unsupported group tasks by using general communication tools. If a groupware toolkit and a groupware development support system (GwDvSS) are provided to the application developers, these applications would be developed at lower cost.

GwDvSS should be able to be used not only for developing groupware applications, but also for customizing groupware configuration. In case of cooperative writing systems, for example, comments and corrections are transmitted with e-mail. If the system user wants to change the number of commentators or co-authors, e-mail routing must be customized. Such customization are sometimes prepared as built-in features of the cooperative writing system. If such customizations can be implemented without changing any application code, groupware applications would be more flexible.

For the above reasons, GwDvSS is very important and promising. GwDvSS should provide the following features:

1. Workflow design:
   (a) Information structure design, including multimedia data.
   (b) Information processing program design.

* {tarumi,tabuchi,yoshifu}@obp.cl.nec.co.jp

   (c) Information routing design among people, designed programs, and existing tools.
   (d) Hierarchical workflow structure design.
   (e) Teleconferencing system usage design in the groupwork context.
   (f) Bulletin board usage design in the groupwork context.
   (g) Organizational structure design for the groupwork participants.
2. Generating executable rules and/or codes from the workflow.
3. Allowing ad-hoc design and customization of the groupwork by end users.

For this purpose, many *workflow* management systems[4] were developed and are in the marketplace now. The authors appreciate them as innovative systems, but they are still insufficient as GwDvSS. First, their application area is mostly focused on business applications. To support other areas, e.g. software development management or co-authoring, GwDvSS should support ad-hoc workflow definition, coordination between realtime and non-realtime groupware tools, multimedia data handling, etc. Second, application development assistance should be enhanced, e.g. by means of workflow libraries.

The GwDvSS described in this paper, IKUMI, generates rules from visually defined workflow, which consists of information structure, processing, and flow design. IKUMI also supports a realtime and non-realtime groupware connection model and parameterized workflow library model. The generated rules are passed to an e-mail platform "MEGUMI," which can handle multimedia data, developed by the authors.

Thus, IKUMI is designed to support all above requirements, while existing workflow systems do not or weakly support 1(a), 1(e), 1(f), 1(g), 2, and 3.

After describing MEGUMI in Section 2, IKUMI profile is given in Section 3. Special IKUMI workflow features will be described in Section 4. Comparison and discussion will be given in Section 5.

2 MEGUMI: the E-mail platform

2.1 Features

MEGUMI is an e-mail system developed by the authors. It has the following pertinent features:

- Semi-formal message structure, like OVAL[5], with form oriented user interface, developed on Canae[6] and OSF/Motif UI toolkit.
- Multimedia data: figures, images including handwritten image and fax image.
- Form definition based on an expansion of the MIME[7].
- Flow management features: circular mail, express mail, and deadline management.
- File attachment on mail, as an external-body defined in MIME standard.
- Mail tracking.

- Mail processing rules in the HyperScheme language [8]. [2]
- Application Program Interface (API) as C and HyperScheme functions.

2.2 Configuration

Each MEGUMI user has one's own daemon process, megumi-d (Fig. 1). It receives all mail for the user, applies rules every time mail is received, opened, accepted, or sent, reminds the user of a deadline, invokes application programs for the user and passes them the mail, and manages the mail traffic history. Because a megumi-d can ask other megumi-d about the traffic history, mail tracking is possible.

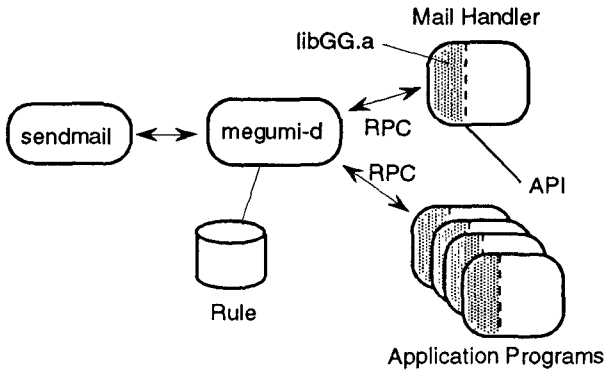

Fig. 1. MEGUMI Configuration

Application programs and mail handler (i.e. mail reader/composer program) are connected to megumi-d, via RPC protocol. Precisely, the mail handler is an application, too. The RPC protocol routines for applications are prepared as a library: libGG.a. The functional specification for libGG.a is the API for MEGUMI.

A megumi-d can communicate with two or more applications with RPC. According to the rule, megumi-d dispatches mail to their corresponding application programs. There are two groupware application architectures with this mechanism. On one architecture, all mail traffic belonging to the application is exclusively handled by an application-oriented tool. On another architecture, application mail traffic is put into one's inbox and handled with the standard mail handler, with special rules pertinent to the application.

[2] HyperScheme is a Scheme language expansion, providing object oriented features and an easy way to accomplish mutual calling with the C language.

3 IKUMI: the GwDvSS

3.1 Configuration

The IKUMI configuration is shown in Fig. 2.

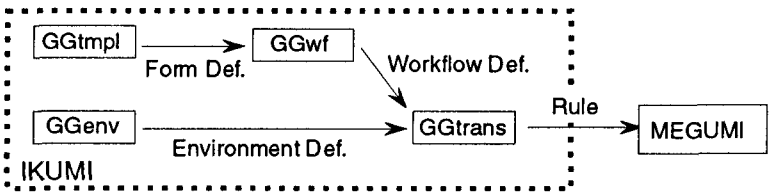

Fig. 2. IKUMI configuration

IKUMI is a set of tools, whose names all begin with the prefix GG[3]. The final tool for IKUMI, GGtrans, generates rules for the MEGUMI mail system. Other tools are all prepared for the definition of workflow or groupwork environment, with graphical user interfaces. Their main tool is GGwf (Fig. 3)[4], for defining the workflow based on the visual workflow chart.

In this subsection, we will describe each tool.

GGtmpl — Form Definition Tool With GGtmpl, the user can define the e-mail format with a visual user interface. Fig. 4 is a hardcopy showing GGtmpl's user interface.

The upper window is a palette of form components. They are form sheet, label, choice field, number field, text field, figure field, and image field, from left to right. The lower big window is the form editor. The user can pick up a form component from the palette, put it on the form editor, and resize individual components to fill the sheet.

Individual component attribute can be specified by double-clicking the component to open the attribute editor, as shown at top of the form editor window in Fig. 4. In this example, the attributes for a choice field — candidate choices, allowed choice number, etc. — are being edited.

After creating a form definition (in the authors' special language), it is passed to GGwf. Since the definition can also be directly referred to by MEGUMI mail to specify the mail's format, GGtmpl can be used separately from other IKUMI tools.

[3] GG stands for Groupware Generator

[4] All screen dumps are taken from IKUMI English version, which is converted from the original Japanese version.

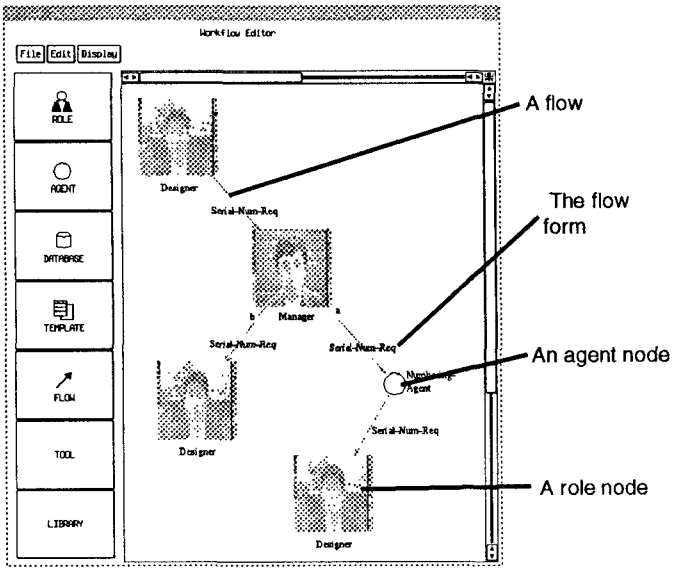

Fig. 3. GGwf

GGwf — Workflow Definition Tool The following explains the GGwf outline with rather simple workflow examples. Enhanced workflow features will be described in Section 4.

As already shown in Fig. 3, the user defines a workflow as a visual workflow chart — a directed graph — in GGwf. Each node may be a *role* or an *agent*. Each arc of the graph is an e-mail transfer. In this paper, individual arc is called a *flow*.

Role Role is a task unit in the workflow, which is performed by a single person. The role definition is given as the one responsible for receiving a form as an e-mail, filling or editing particular fields of a form, invoking a program and putting a form to the program, creating a new form, or sending a form; or a combination of these responsibilities.

These responsibilities are defined in the detailed definition window shown in Fig. 5, which appears when the user double-clicks the role node. In this window, the workflow designer can specify the program that processes the incoming mail, fields that should be filled by the role, conditions for branch, and the processing deadline at this node.

To specify fields that should be filled, the user pops up another window, like Fig. 6, and clicks the target field on the window. This visual definition method is simple and acceptable by end users.

More description regarding condition specifications are given in Section 4.2.

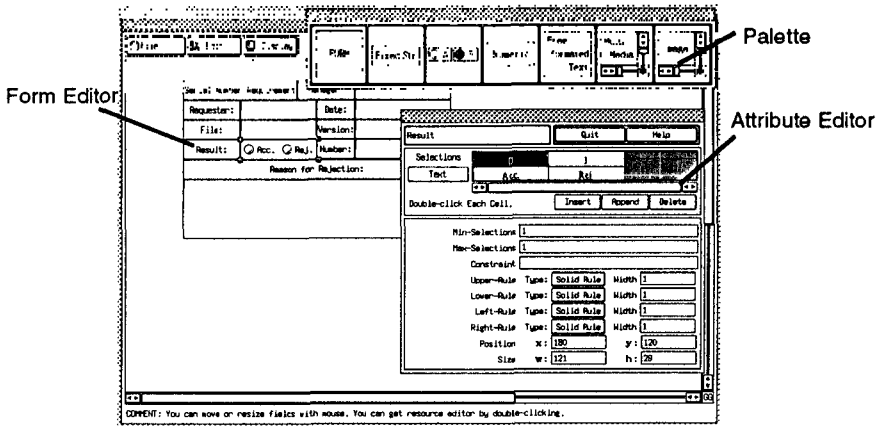

Fig. 4. GGtmpl

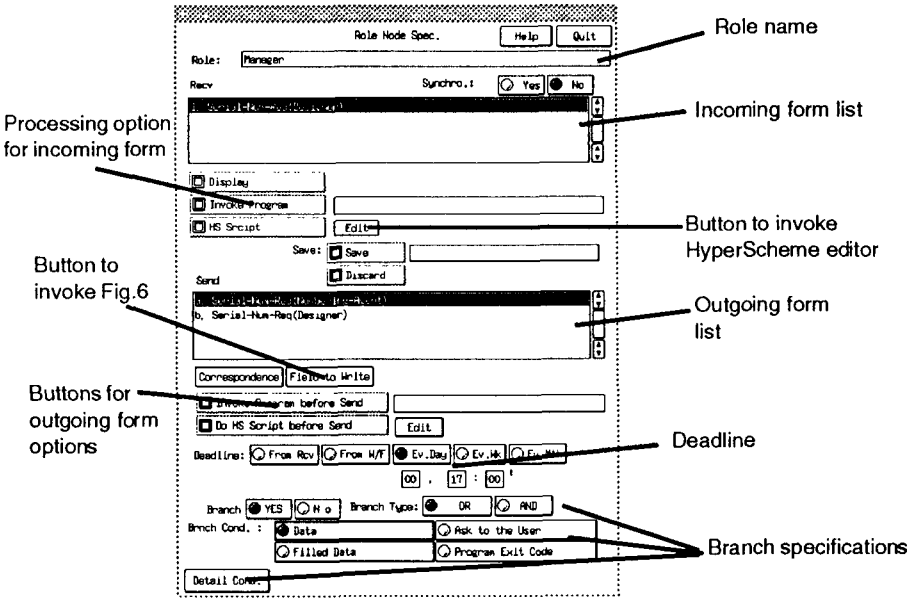

Fig. 5. Node Specification in GGwf

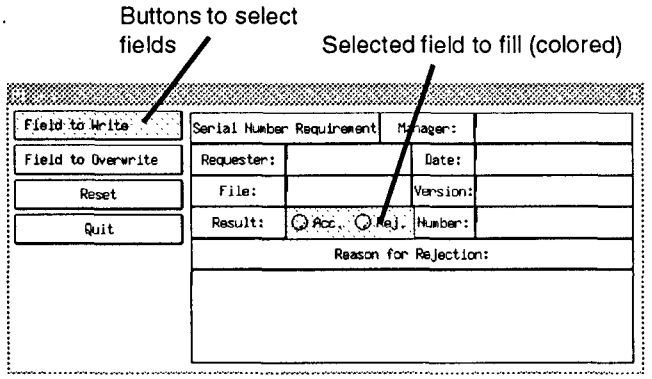

Fig. 6. Field Selection in Node Specification

The user can also define the name for each role, the icon for each role, and constraints among roles. An constraint example is a rule to prohibit a person from concurrently engaging in two particular roles.

Agent In IKUMI, agent means a computer program which receives and sends e-mail traffic, without interacting with any user. For example, a program which automatically fills in the serial number field for a form can be an agent, if the form is given from and sent to another node as e-mail. Each agent is represented as a circle icon in the visual workflow chart.

When the GGwf user double-clicks an agent node, the detail definition window similar to Fig. 5 appears. In this window, the user can specify the program name that processes the incoming mail, or directly write a program in HyperScheme language. If the agent function is simple, e.g. copying fields or filling fields with constant values, the user can specify this visually in almost the same way as in Fig. 6. Hence, even end users can define agents without writing codes.

Flow A flow basically represents an e-mail transfer from one node to another node. The user can specify the mail form by attaching a form name to the flow. More discussion on a special flow type (beacon message) will be presented later in this paper.

Total GGwf Usage Description The GGwf usage outline is as follows:

1. Click the ROLE button and select the role subwindow; create new roles on the subwindow and insert them on the workflow editor.
2. In the same way, click the AGENT button and select the agent subwindow; create agents on the subwindow and insert them on the workflow editor.

3. Connect nodes by flows.
4. Click the FORM button and select a list of forms defined in GGtmpl. For each information flow, a form name must be selected and attached.
5. Double-click each node, and select a detail definition window for the role or agent, and give the detailed definition.
6. Save the workflow and quit.

The GGwf output is given as a set of S-expressions.

GGenv — Environment Definition Tool With GGenv, the user defines the environment for executing the groupware application. For example, the name of person who fulfills each role, machine type which each person uses, and people's organizational structure are environmental parameters. These definition will be reflected onto the rule generation phase at GGtrans.

Currently, GGenv only supports the definition of the name of the person who fulfills each role. To achieve workflow portability, it is necessary to separate this definition from the workflow.

GGtrans — Rule Generation Tool GGtrans is the only tool which has no graphical interaction with the user. It receives the definitions from other IKUMI tools, and generates rules for each user's megumi-d.

For each person engaging in any role in the workflow, GGtrans generates rules, i.e. compile workflow and environment definition into rules, that will be invoked at the event of creating, receiving, and sending a form, and originating a workflow. It also generates metarules, which are used to select the rule file at each event. These rules and metarules are distributed by control mail to the megumi-d for each groupwork member.

If a person engages in two or more roles in a workflow, rules for all these roles are merged for the user with metarules. The user's megumi-d selects the proper role rule, when it receives or sends e-mail traffic, by evaluating the metarules.

An metarule example is:

```
( gg-eval-rule "receive"
  ( "ex1" 1.0  1.0  "FL001" ) ( "FormA" 1.0 1.0 ) "ex1-1.0-C-1.scm" )
```

This metarule means that, if the megumi-d receives mail, whose workflow-name is "ex1," whose workflow-version is 1.0, whose flow-id in the workflow is "FL001," whose form is named "FormA," and whose form-version is 1.0, then it executes a rule file "ex1-1.0-C-1.scm." Flow-id is a unique ID in a workflow, which is appended on every e-mail, by the sender node's rule at the sending event.

For each agent, rules are given to the agent program. The agent program is a UNIX program, invoked from `/usr/lib/sendmail` on receiving mail traffic.

3.2 Target Users

IKUMI's target users are groupware application developers and groupware end users. This subsection describes the applications for these target users.

Groupware Application Developers Application developers develop groupware application, in the following way:

1. Designs the workflow, and defines it on IKUMI tools.
2. Installs the generated rules and agents.
3. If necessary, develops programs which are called from the rules or agents, using MEGUMI's API (libGG.a).
4. Installs the programs.

Groupware End-Users End users can use IKUMI in almost the same way as the developer's. However, because they cannot develop programs in C or HyperScheme language, function of agents and roles are restricted.

Even with this restriction, end users can develop form handling groupware applications. They can define forms, each role's responsibility for filling form fields, and flows. They can even define agents for copying or filling form fields. MEGUMI's standard mail handling program is used for such an application. It supports sufficient features for such simple groupware applications, i.e. form-like user interface and interface for reminding the user of a deadline for making a response.

Thus, the end-user can create and install ad-hoc workflow definitions. This is important for supporting real office work.

4 Workflow Features

This section describes special IKUMI workflow features. These features are given by expansion or additional attributes for flows (Section 4.1) or nodes (other). These features enhance the workflow expressive power in the visual environment.

4.1 Beacon Message

A beacon message is a special flow category, which is implemented as e-mail with no body and is invisible to the end users. A beacon message is represented as a broken line arrow in GGwf.

It is useful to synchronize group members' activities. For example, assume such a workflow as that shown in Fig. 7. Role A, the originator of this workflow, creates a progress report and sends it to Role C. At the same time, Role C requires the newest financial report, which is made by Role B, in reading the progress report. In this case, Role A sends a beacon message to Role B, when Role A sends the progress report to Role C. Role B's megumi-d prepares (i.e. invokes or creates a GUI icon for) an editor for the financial report, when it receives the beacon message. The beacon message is, of course, automatically sent by the rule. With this mechanism, Role A and Role C do not need to write a message to Role B, like "Please prepare the newest financial report." Role B does not need to read such routine-work mail, either.

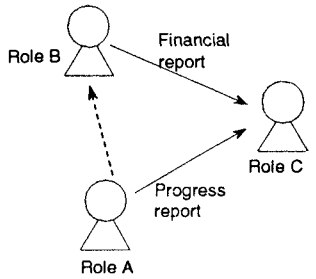

Fig. 7. A Workflow with Beacon Message

4.2 Branch in Workflow

Some workflows have flow branches. IKUMI provides two branch categories: *AND-branches* and *OR-branches.*

The OR-branch specification is visually defined in the detailed definition for the branching node, with a window such as shown in Fig. 8. The user defines the specification by clicking the focused field on the form, specifies the value and operator for comparison, and gives the selection for each comparison result.

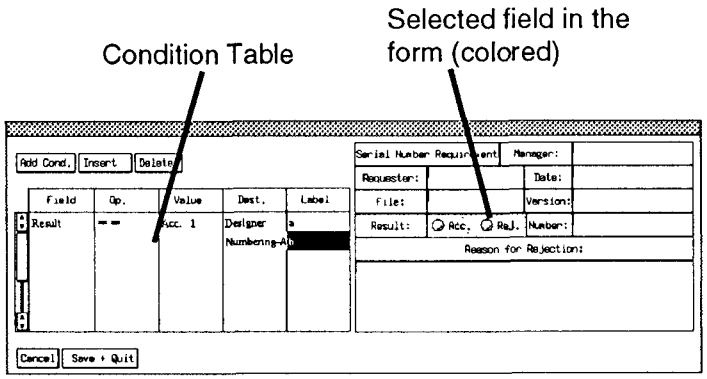

Fig. 8. Window for branch specification

4.3 Realtime Conference Node

Realtime communication and non-realtime communication are both necessary for groupwork. Most existing groupware tools support only one of them. The

authors support seamless connection between non-realtime communication and realtime communication by *conference nodes* in groupwork workflow.

Fig. 9 is a conference node example in a workflow definition. A conference node has attributes such as the name of the chairperson, names of participants, and the name of minutes writer. All incoming flows to the conference node represent the required documents for the conference. The sources for these flows are the roles responsible for the documents. The outgoing flows from the conference node represent the minutes, and the destination nodes of these flows are the receivers of minutes other than participants.

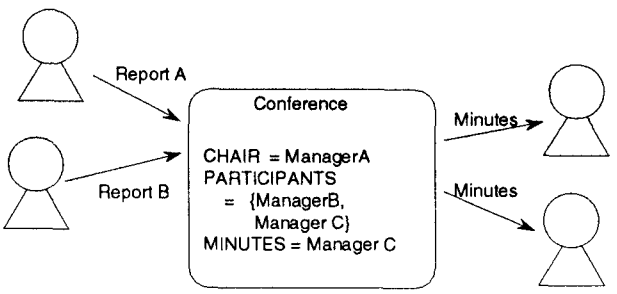

Fig. 9. A Conference Node Example

From such a workflow, GGtrans will generate the following rules for the chairperson's megumi-d .

1. Rules to wait for all incoming documents.
2. Rules to prepare a GUI icon for the conference, which can invoke a conference program with specified parameter file.

GGtrans also generates a parameter file for the conference. The parameter includes the names of required documents, the conference participants, the chairperson's and minutes writer names. When the conference is started, an editor for the minutes appears on the minutes writer's display. The editor sends the minutes to all destinations by e-mail, when the conference is finished and minutes are completed.

This type of workflow is still effective for a non-electronic, traditional style meeting. In such a case, GGtrans can generate rules for the chairperson, like:

1. Rules to wait for all incoming documents
2. Rules to send these documents to all participants by e-mail
3. Rules to show a message to the chairperson indicating that the chairperson can call the meeting.
4. Rules to send a beacon message to the minutes writer, which let the minutes writer's megumi-d prepare the editor for minutes.

4.4 Workflow Library

Frequently used workflow patterns should be collected and prepared as a workflow library set. For example, a *comment-and-rewrite* pattern should be prepared as a parameterized skeleton for the workflow in Fig. 10(a). It will be used as a node in a concrete workflow in the same way as macro expansion, as shown in Fig. 10(b).

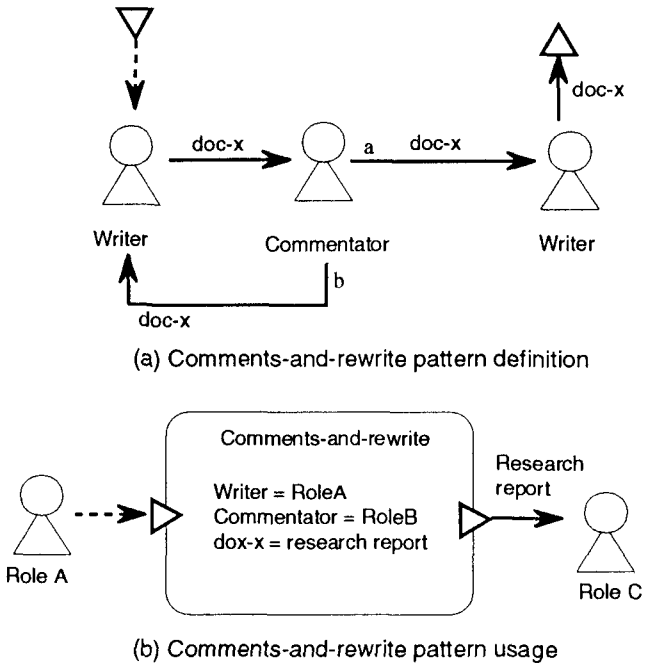

Fig. 10. An Workflow Skeleton Example

Note that the role "writer" in Fig. 10(a) is replaced by "Role A" in (b). This causes the source and destination nodes for the beacon message to be identical. In such cases, no beacon message is made when this workflow is executed, but a state transition from the previous mode to the comment-and-rewrite mode is caused in Role A's megumi-d .

5 Comparison and Discussion

5.1 Comparison with Other Workflow Systems

The closest systems to IKUMI would be WorkMAN [9][10] and Regatta[11]. WorkMAN provides Forms Designer and Forms Router tools, corresponding to

GGtmpl and GGwf. Regatta[11] also provides a visual environment for groupwork design. IKUMI's most notable differences from them are visual programming environment with direct manipulation on forms (Fig. 6 and Fig. 8) and connection to realtime conference.

5.2 Mail vs. Shared File

IKUMI's workflow model represents every flow as an e-mail transfer. Data transfer among groupwork members can also be realized by shared files. However, IKUMI also supports a file sharing type of data transfer. As described in Section 2, the user can attach a shared file as an external-body component to MEGUMI mail. In this case, the mail is a control transfer among groupwork members, and it is logically equal to the workflow system based on shared files.

Using this method, a simple and uniform control transfer mechanism can be obtained. Regardless of the data transfer style — mail, shared file, or hybrid —, megumi-d can get all control transfers from any person or agent by mail, so that megumi-d can manage all to-do items for all workflow definitions uniformly.

5.3 Distributed vs. Centered Control

Some workflow systems have centered control mechanisms [9]. In this category, there is logically one control and data center. (Physically, the data repository may be duplicated or distributed.) IKUMI and MEGUMI do not adopt it. In the author's system, megumi-d units have distributed control, and data stores are personally managed or given as project repositories in each workflow. Megumi-d units exchange implicit control messages to track mail, to inform about dynamic alternations to mail route, etc. Such distributed architecture can be applied to cross-organizational working group.

6 Conclusion

This paper has described a Groupware Development Support System IKUMI, especially its function and the workflow model. IKUMI supports multimedia mail (requirement 1(a) in Section 1), agents(1(b)), mail routing (1(c)), workflow libraries (1(d)), conference node (1(e)), rule generation (2), and visual environment which gives easy interfaces even to end users (3).

Groupwork support is expected in not only business applications, but also in CASE and CAD applications. In these areas, questions and answers on product specifications, bug reports, progress reports, etc. are e-mail items. Tracking these documents is helpful for manager's work, and also an effective way to satisfy ISO-9000 standards. Of course MEGUMI's multimedia mail feature is indispensable to these application areas.

The authors believe that the workflow system is an important unit in the groupware toolkit, but it is not the entire thing. Other important factors are co-producing and sharing. As for co-producing, multimedia document review

and decision support would be important functions in a groupware toolkit. For sharing, bulletin boards must be in the toolkit. Co-producing and sharing toolkit are now being developed.

IKUMI and MEGUMI are implemented on SVR4 UNIX system. Client program library, libGG.a, is also implemented on personal computers running MS-Windows 3.1 operating system. IKUMI and MEGUMI are applied to CASE and CAD products.

Acknowledgements

The authors would like to thank Masao Managaki, Hitoshi Miyai, and other laboratory members for helpful comments on this research, and thank Hiroyuki Yagyu for the implemention of IKUMI and MEGUMI. Finally the authors wish to acknowledge the support of several divisions in NEC.

References

1. Watabe, K., et al. : Distributed Multiparty Desktop Conferencing System: MERMAID. Proc. of ACM 1990 Conf. on CSCW (1990) 27–38
2. Leland, M. D. P., et al. : Collaborative Document Production Using Quilt. Proc. of ACM 1988 Conf. on CSCW (1988) 206–215
3. Conklin, J., and Begeman, M. L. : gIBIS: A Hypertext Tool for Exploratory Policy Discussion. Proc. of ACM 1988 Conf. on CSCW (1988) 140–152
4. Marshak, R.T.: Requirements for Workflow Products. Groupware '92 Proceedings, Coleman, D.D. Ed. (1992) 281–285
5. Malone, T.W. and Fry, Ch. : Experiments with Oval: A Radically Tailorable Tool for Cooperative Work. Proc. of ACM 1992 Conf. on CSCW (1992) 298–205
6. Tarumi, H., Rekimoto, J., Sugai, M., et al.: Canae — A User Interface Construction Environment with Editors as Software Parts. NEC Research and Development. **98** (1990) 89–98
7. Borenstein, N. and Freed, N.: MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies. RFC1521 (1993)
8. Saji, K. and Kageyama, T.: HyperStation: Concept of Distributed Object-Oriented Dynamic Language HyperScheme. Proc. 45th Annual Conf. of IPSJ. (1992) 2Q-1 (in Japanese)
9. Reinhardt, A.: Smarter E-mail is Coming. Byte. **18-3** (1993) 90–108
10. Udell, J.: Workman Needs Work. Byte. **18-9** (1993) 167–170
11. Swenson, K.D.: A Visual Language to Describe Collaborative Work. Proc. of 1993 IEEE Symp. on Visual Languages. (1993) 298–303
