GroupIT - a prototypical Group Interaction Tool

Report about the current state of
a Smalltalk 80 groupware application

Institute for Telematics,
Dept. of Informatics,
University of Karlsruhe

Dipl.-Inform. Thomas Stingl

September 1992

Supervision: Prof. Dr. Gerhard Krüger
Dipl.-Inform. Tom Rüdebusch

Contents

List of Figures

Chapter 1

Abstract

GROUPIT is a prototypical Group Interaction Tool and provides groups of collaborating users with integrated support for various interaction types, amalgamating capabilities of well-known collaborative applications like multi-user editors, conferencing systems and electronic mail within one generic interaction model.

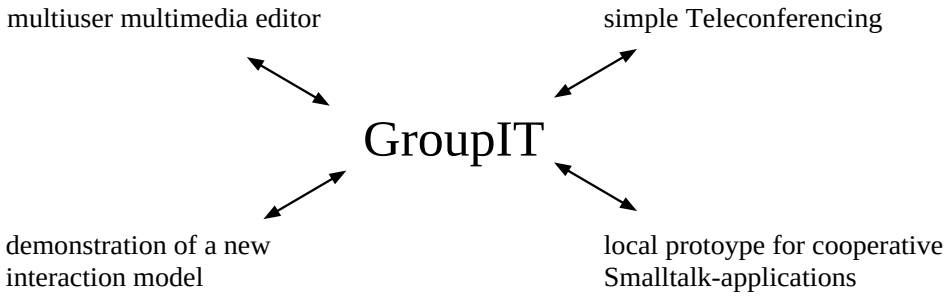

Figure 1.1: GROUPIT - general description

 Comments, requests for latest versions or related publications should be directed to Tom Ruedebusch, Institute for Telematics, Department of Computer Science, University of Karlsruhe, Zirkel 2, 7500 Karlsruhe, Germany (Email: tom@ira.uka.de).

Implementation by Thomas Stingl (Email: stingl@telematik.informatik.uni-karlsruhe.de or stingl@fzi.de).

In this report about 13 months of programming work as a part time scientific assistant programmer after gaining the master degree at the Institute for

Telematics of the University of Karlsruhe we show the current state of the Group Interaction Tool GROUPIT from September 1992. Furtheron we want to explain some details of the implementation.

All the work is based on the ParcPlace Smalltalk Release 4.0 implementation of GROUPIT developed during my master thesis [Stingl 91], which belongs to GROUPIE (Group Interaction Environment). Meanwhile the current development platform is the Release 4.1.

For people who have not heard of GROUPIT yet please refer for a quick description to illustration 1.1 and see the published articles about GROUPIE [Rüdebusch 91a, Rüdebusch 91b].

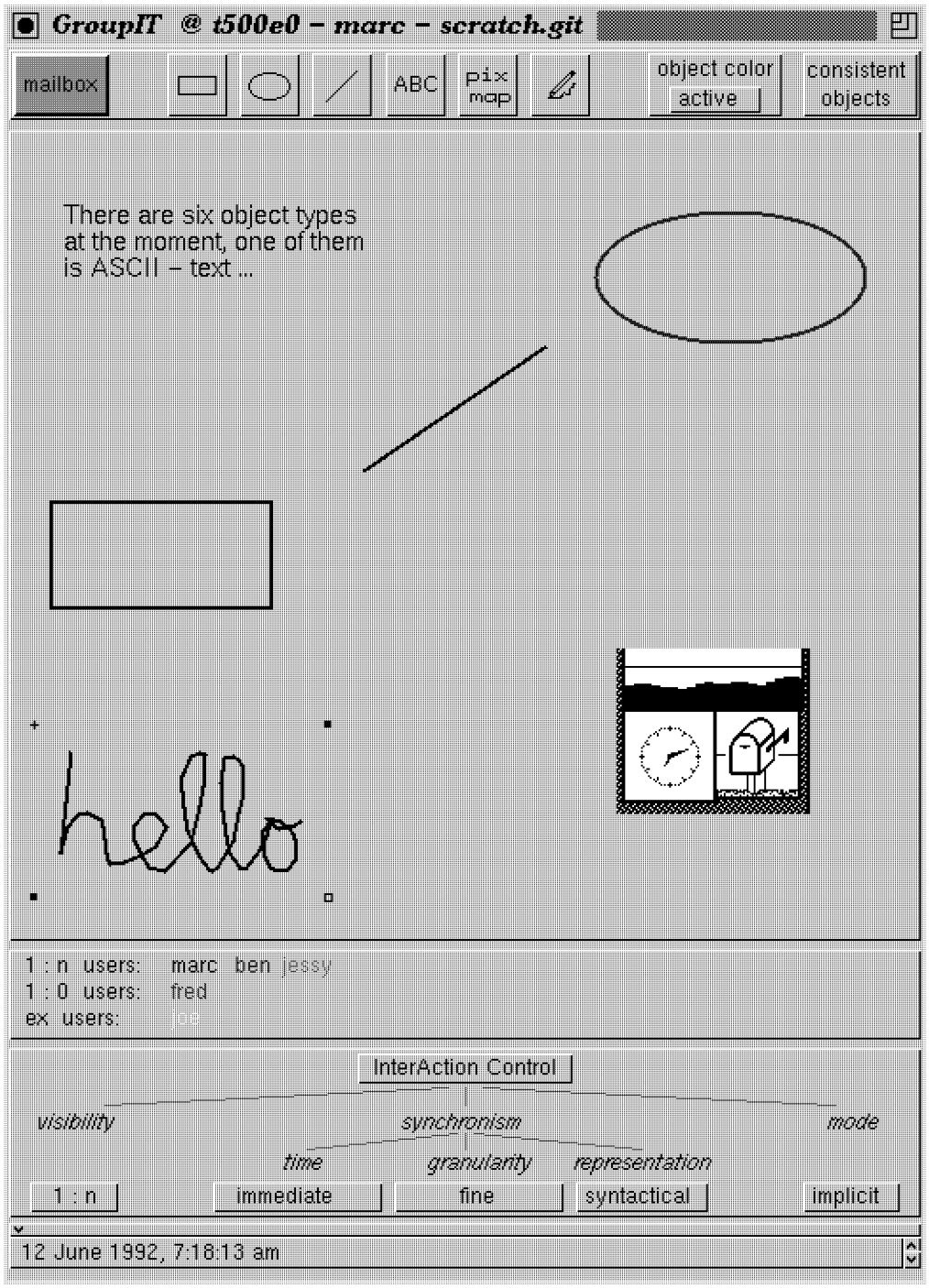

Figure 1.2: GROUPIT - User Interface

Chapter 2

User Interface and Functionality

The actual appearance of the GROUPIT - window can be seen in picture 1.2 on page 6.

Additionally to the known objecttypes listed in [Stingl 91, p. 43, ch. 5.2.2] there are now the classes `Pixmap` and `PaintItem` implemented. Pixmaps contain a part of the screen surface, PaintItems are used for user drawn items.

The interaction features listed in [Stingl 91, p. 47, ch. 5.2.3] and shown in picture [Stingl 91, pict. 5.7] are completed by *1:0 - explicit* - private mailing. The current document is converted to the 'mail' - Format.

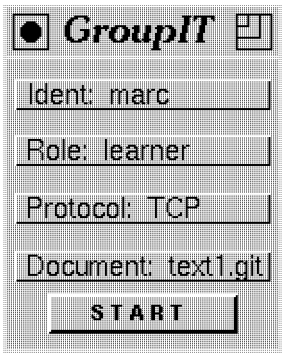

Figure 2.1: GROUPIT - Startwindow

Under the partial aspect *operation granularity* now smoother degrees are offered, additional to 'fine' and 'coarse' there exist the medium values called 'mediumfine', 'medium' and 'mediumcoarse'. These medium values are correlated to a sampling algorithm.

The role choose described in [Stingl 91, ch. 5.3.2] is now always active, that means that if there is a choice among the users to be done, always first there is a selection of the roles and after that there is a confirmation with the user names (idents) itself.

The use of most of the buttons - in particular those for interaction control and object coloring - has changed: instead of having a button for each possible value there is now only one button per paramater which offers a popup-menu when selected.

We don't use a special exit button any more; like many other smalltalk applications you can close the window (and thereby exit the GroupIT - session) using the right button menue of the mouse.

The **Startwindow** (Pict. 2.1) makes it easy to start the tool with its varieties concerning the actual user id, -role, communication protocol and document.

Following values are set as default: for *ident* the unix-account, under which the smalltalkimage is running; for *role* the standardrole 'learner'; for *protocol* the protocoltype 'TCP' and for *document* the default name 'scratch.git'. The chosen role and communication protocol values are stored as new default values to be used for new instances of the start window.

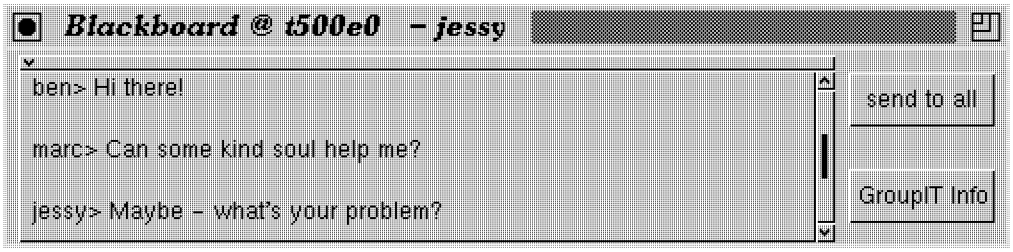

Figure 2.2: `Blackboard` - Window

Is the protocol type '-none-' chosen, then GROUPIT will run without any intratool communication and therefore can be started for demonstrations purposes on arbitrary smalltalk - platforms (until now the implemented lower communicationslayers require a Unix - OS).

The simple teleconferencing system called **Backboard** has changed in the user interface too, as can be seen in picture 2.2. The feature of *strong WYSIWIS* is obtained though.

The important middle-button menue used for editing is showed in pict. 2.3.

To be complete we show the log window in pict. 2.4, which however is primarly intended for programmers only. It can be opened using the middle-

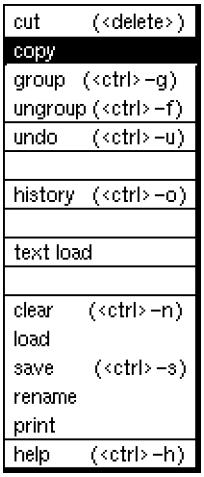

Figure 2.3: Menue within `MMView`

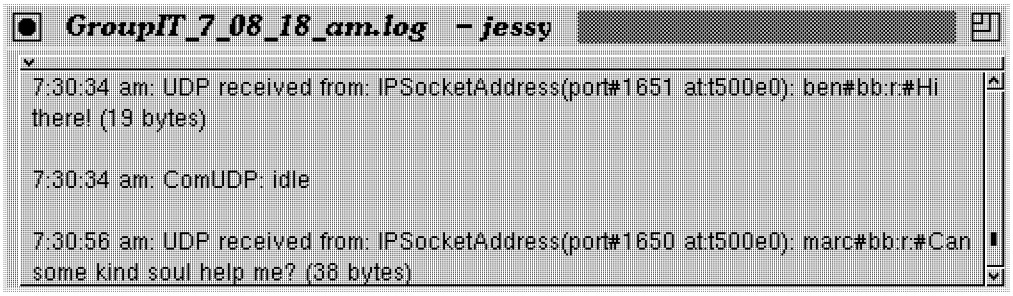

Figure 2.4: `Log` - Window

button menue of the 'state view' right at the bottom of the GROUPIT window. The menue is shown in pict. 2.5.

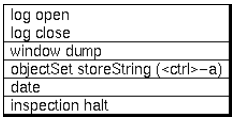

Figure 2.5: Menue within stateview

Chapter 3

Implementation

An overview about the class categories can be found in the illustrations 3.1 and 3.2. The once within one class category 'GroupIT-General' combined classes are now spread over four class categories *GroupIE-Support*, *GroupIE-Unix-Support*, *GroupIT-General* and *GroupIT-MultiMedia*; furtheron there exist corresponding installation files named *InstallGroupIE-Support.st* and *InstallGroupIT.st*.

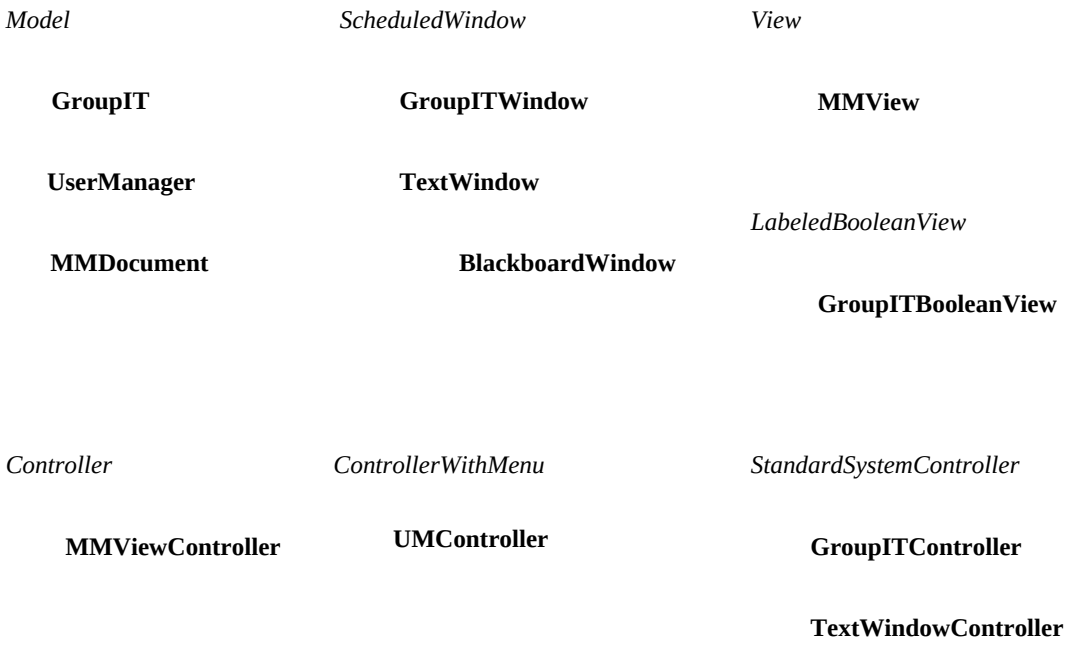

Figure 3.1: Class Overview (part A)

A diagram of important relations between the most important instances can

be seen in 3.3.

3.1 Class Category GroupIT-General

3.1.1 BlackboardWindow

`BlackboardWindow` is a subclass of `TextWindow` with additional methods for text input and output.

An instance of `BlackboardWindow` is shown in ill. 2.2.

3.1.2 GroupIT

When using GROUPIT for demonstration purposes we faced the problem, that the remote Smalltalk image stopped updating the open windows after some seconds of no user input. The solution was to send the message 'checkForEvents' to the `ScheduledWindow`. This is done e.g. within the method `GroupIT>>oVM:r:`.

3.1.3 GroupITBooleanView

GroupITBooleanView adds the possibility of easy access to the 'offImage' of a LabeledBooleanView. We need this feature for changing the color of some buttons at run time.

3.1.4 GroupITController

A slightly modified subclass of `StandardSystemController`, used for the blue button 'close' feature.

3.1.5 GroupITWindow

A slightly modified subclass of `ScheduledWindow`, responsible for the display of the interaction tree within the GROUPIT - window. It has an own version of the superclasses method *displayDamageEvent*, which sends the remaining two new messages *displayTreeGray* and *displayTreeRed*.

3.1.6 InteractionControl

InteractionControl implements the interaction aspects developed for GroupIE. The parameters for interaction settings are implemented according to the model for interaction as described in [Rüdebusch 91b].

3.1.7 UserManagerController

UMController is the controller for UserManager and a MMView. It offers a menue for displaying remote user data.

3.1.8 UserManager

The heart of the UserManager is the dictionary of the user descriptions which in fact are dictionaries too. At the moment every user description has thirteen key - value pairs.

The sampling algorithm implemented in the method `UserManager>>listFromSample:` gets from `MMViewController` as parameter the amount of pixels, about which the local cursor has been moved. Additionally the current value of the local granularity is required; if it has the values 'coarse' or 'extracoarse' the method breaks off and an empty user list is returned.

3.2 Class Category GroupIT - Multimedia

When implementing the MultiMedia - features [Stingl 91, ch. 5.3.2] we strove for good modularity, so the introduction of the classes `MMPixmap` and `MMPaintItem` to the `MMObject` - hierarchy was straightforward.

3.2.1 MMDocument

MMDocument is prepared to be model of a MMView and the corresponding MMViewController.

For reasons of quickness the MMDocument holds the information about remote user cursors too.

There is no remote cursor visible, when

- the corresponding user is not within the MMView (MMView controller controlloop)
- the corresponding user in 1:0 mode or leaves GROUPIT
- granularity is 'extracoarse'

3.2.2 `MMDocument`

A slightly modified subclass of ScheduledWindow, adapted for the display of the contents of a MMDocument. It is used for the printing feature of GroupIT.

3.2.3 `MMObject`

MMObject is used by GroupIT. It's on top of a hierarchy of multimedia objects.

The classes **`MMRectangle`**, **`MMEllipse`**, **`MMLine`**, **`MMText`** and **`MMPaintItem`** are obvious derivations from **`MMObject`**.

3.2.4 `MMPixmap`

The image contents is stored in black/white always; the color - if necessary - is created when displaying. Reason: there are converting losses if we change the pixmaps in different colors (threshold problem).

3.2.5 `MMView`

The default controller is MMViewController, model might be aMMDocument or aUserManager.

3.2.6 `MMViewController`

This is a versatile controller for instances of MMView; a mixture between WidgetController and ParagraphEditor with special features for the MMView. The model should be a MMDocument.

The **`MMViewController`** accepts the following special keyboard input:

ctrl-a: display of all objects in storage format (string)

ctrl-h: help

ctrl-n: new dokument

ctrl-o: object data

ctrl-u: undo

ctrl-s: save document

Blank: like pressing the left mouse button (e.g. selection).

When granularity is 'extracoarse' a new object is hold 'invisible', until the interaction insert/size is terminated.

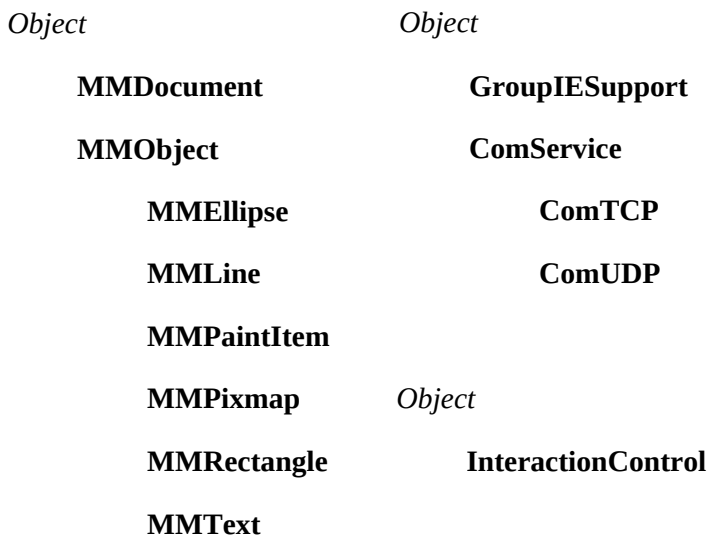

Figure 3.2: Class Overview (part B)

3.3 Class Category GroupIE-Support

We thought of a pool for smalltalk hacks to be used within the GROUPIE project.

3.3.1 BufferedWrapper

The class BufferedWrapper, found in USENET news (thanks to the unknown author of this goodie), is mandatory for flicker-free displaying of graphic

objects.

BufferedWrapper buffers its component to avoid flashing. It caches a Pixmap in pixmap so one doesn't need to be created every time it displays.

3.3.2 `GroupIEFileBrowser`

GroupIEFileBrowser returns the selected filename. Therefore you must provide a reference to the client and the methodname to be sent with the selected filename. The third parameter is an optional pattern string.

3.3.3 `GroupIESupport`

The class GroupIESupport contains stuff useful for all implementations within GroupIE, methods which should be available from the system.

Here we collect methods which seem to be missing within Smalltalk yet but which are useful for the implementation of GroupIE. We have to use class methods to allow access without creating new instances.

For usage of the described Smalltalk TCP/IP communication modules apart from GROUPIT see e.g. [Arendt 92].

3.3.4 `TextViewNoMarker`

This is just the same as TextView except that the marker is invisible.

3.3.5 `TextWindow`

For the display of text within a separate window we developed the class `TextWindow`. GROUPIT uses such windows for the *Blackboard* and for the *log*. In the past these windows were implemented like the *System Transcript*, but due to the necessary class variables this was not the best way. The current solution is much less complicated and has shorter code size.

3.3.6 `TextWindowController`

The only difference to the `StandardSystemController` is the removal of the blue button menu option 'close'; the instances of `TextWindow` are closed by

the creating program only.

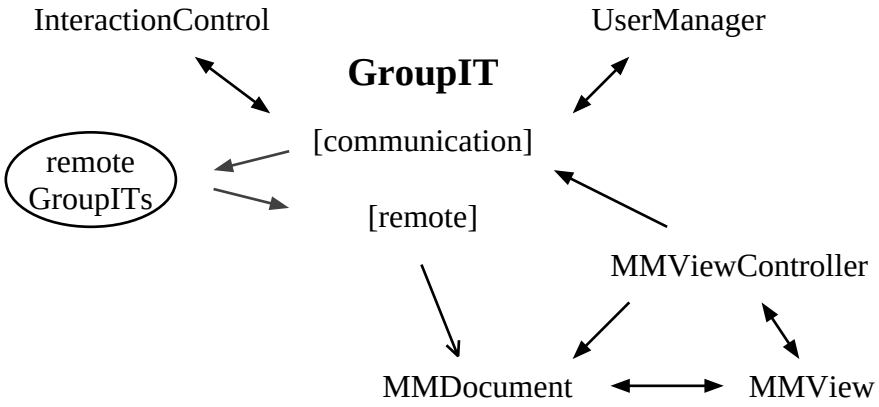

Figure 3.3: Internal class / instance relations

3.4 Class Category GroupIE-Unix-Support

As kind of framework for the intratool communication classes, which are implemented independent from GroupIT the class category *GroupIE-Support* was created.

The class `UDPService` explained in [Stingl 91] was developed further into the class hierarchy `ComService`, `ComTCP` and `ComUDP`.

The interface methods are: for sending *broadcast:*, *multicast:*, *send:ToUser:* and *send:to:*; received messages are passed to the method *eval:from:protocol:* provided by the actual 'client' of the communication instance.

3.4.1 `ComService`

ComService provides ASCII telecommunication using either the TCP or the UDP protocol. It maintains the general communication variables for its subclasses. You get an instance of ComUDP or ComTCP by calling following class method of COMService:

COMService for: aClient protocol: aProtocol portNames: pnames

The client must provide the following methods:

eval: aBuffer gotFrom: anAddress via: aProtocol (is called when a new message has arrived)

log: aString (called with log messages form communication)

See method category 'sending' to for send methods. A method called randomDelay is offered for simulation purposes.

The method `ComService>>startReceiving` starts the main receiving process *(mainProcess)*.

3.4.2 ComTCP

You should check the method connectTCP:, there is a timeout variable which you can tune for your own needs. See class method category 'documentation' for further explanations of the implemented TCP communication.

3.4.3 ComUDP

ComUDP is the subclass of ComService using the UDP Protocol for communication.

3.4.4 ComUnixSocketAccessor

Class ComUnixSocketAccessor is used by the `ComService` hierarchy communication instances, it is a little extension for the system class UnixSocketAccessor.

The only difference to the class `UnixSocketAccessor` is the timeout feature when connecting remote TCP ports.

The timeout is done within the loop of the method *waitForConnectionWithTimeout:*; we allow max. number iterations. Within each iteration, the connection is tested and a delay is executed. The length of the delay is determined by the method `IOAccessor>>readPauseInterval`.

The other methods of `ComUnixSocketAccessor` only pass on the timeout value to the mentioned method.

Chapter 4

Miscellaneous

4.1 Tuning

In the following we describe some tuning- and adaption possibilities of the internal GROUPIT - software, which can't be done using only the user interface.

For simple simulation purposes there exists a random delay feature in `ComService`, which is activated when selecting the interaction parameter 'time - no constraints'. Every transmission is delayed for a time within 0 and the value of the class variable *RandomDelayMax*.

We tested the fork of the eval method out of the receiving process both in `ComTCP` and `ComUDP`, but the process scheduler seems not to be optimized ...

4.2 Software requirements

As additional software to the standard smalltalk-image the following file-ins are required:

- UnixSock.st

  ... belongs to standard Smalltalk Release.

- ExtIPC.st, UnixIPC

  ... belongs to standard Smalltalk Release.

- OKSupprt.st

... in fact we use the class `SelectionSetInListView` only; the whole is distributed with ObjectKit / Advanced Programming.

4.3 Interface to the Synchronization Editor

For use of GROUPIT documents within the Synchronization Editor described in [Blakowski 92] we developed following methods:

`GroupIT(class)>>displayOnGC:document:`
`GroupIT(class)>>displayOnMMWindowDocument:`

The document is expected to be filed. Furtheron the method

MMDocument(class)>>withClient:fromFile:

can be used to create instances of `MMDocument`, which accept the messages *size* and *displayOn:*.

4.4 Sourcecode differences for PP Smalltalk Release 4.0 and Release 4.1

For adapting the GroupIT source code to Release 4.1 we had to change the following methods:

`GroupIT (class) >>startInteractive:`

replaced xButton display by xButton invalidate

`TextWindow>>goDown:`

simplified

`MMObject>>deepCopy`:

We had to write an own method ...

`MMViewController>>dispatchKeyboardEvent:`

`dispatchTable now adds an event rather than a character`

`InteractionControl>>set...Buttons:`

`replaced xButton display by xButton invalidate`

`UserManager>>userInfo:`

Here and within other methods we had to replace *DialogView show:withLabel:* with *PopUpMenu labelArray:*.

GroupIT>>openView:

window openNoterminate changed to window open

GroupIT(class)>>start:

Smalltalk garbageCollect changed to ObjectMemory garbageCollect

The installation files are modified.

4.5 Troubleshooting

Problem:

After activating the 'Start' - button the GROUPIT - window does not open, but an error message occurs complaining not allocable storage.

Cause:

The smalltalk image has not enough object memory to create and allocate a new instance of a MMView.

Solution:

GarbageCollector activation (*Smalltalk garbageCollect*)

Problem:

The method UnixSocketAccessor>>getName works only sporadic; we want to use it for the log.

Cause:

Problem within the primitive.

Solution:

Placed as comment only within the methods of ComService hierarchy.

Problem:

When trying to connect a long distant internet node we get

connection timeout.

Cause:

See chapter about the ComUnixSocketAccessor.

Solution:

You can increase the timeout value within the method ComUnixSocketAccessor>>connectTCP: .

4.6 Future work

It appears straightforward to add other edit features and other objecttypes as well, like arrows etc. or video clips.

Being the first prototype for local Smalltalk groupware and considering the results of other applications currently in the state of development, e.g. the local work done in the field of distributed object management and the steady implementation of the basic modules of the GROUPIE environment, it would be useful to integrate this work to a complete orthogonal structurized environment for collaborative work.

Bibliography

[Arendt 92] Ralph Arendt. *Erweiterung des MVC-Konzepts um Mehrbenutzer-Fähigkeit*. Studienarbeit, Institut für Telematik, Universität Karlsruhe, 1992.

[Blakowski 92] Gerold Blakowski, Jens Hübel, Ulrike Langrehr. Tools for Specifying and Executing Synchronized Multimedia Presentations. In R. G. Herrtwich (Hrsg.), *Network and Operating System Support for Digital Audio and Video, Second International Workshop, Heidelberg, November 1991, Proceedings*, S. 271--282. Springer, Berlin etc., 1992.

[Rüdebusch 91a] Tom D. Rüdebusch. Development and Runtime Support for Collaborative Applications. In *HCI International '91 --- IVth Intl. Conference on Human-Computer Interaction*, Stuttgart, Germany, Sept. 1991. To be published.

[Rüdebusch 91b] Tom D. Rüdebusch. The Interaction Concept as a Basis for Open CSCW Systems. In *ECSCW'91 Developers Workshop*, Amsterdam, Sept. 1991.

[Stingl 91] Thomas Stingl. *Problematik und Lösungsansätze verteilter CSCW-Anwendungen am Beispiel einer exemplarischen Realisierung*. Masterthesis, Institute for Telematics, University of Karlsruhe, 1991.