DOCUMENT RESUME

ED 448 703 IR 020 466

AUTHOR Charlton, Colin; Little, Janet; Morris, Simon; Neilson, Irene

TITLE Beyond Chat--New Generation Software for Real-Time Discussion.

PUB DATE 1999-10-00

NOTE 7p.; In: WebNet 99 World Conference on the WWW and Internet Proceedings (Honolulu, Hawaii, October 24-30, 1999); see IR 020 454. Some figures contain illegible font.

PUB TYPE Reports - Descriptive (141) -- Speeches/Meeting Papers (150)

EDRS PRICE MF01/PC01 Plus Postage.

DESCRIPTORS *Computer Mediated Communication; *Computer Software Development; Computer System Design; Design Preferences; *Group Discussion; Programming; *World Wide Web

IDENTIFIERS *Chat Rooms; Client Server Computing Systems; Java Programming Language

ABSTRACT

This paper discusses the need for and the design requirements of generic chatroom software that is readily configurable to particular domains and that is also extensible. The architecture of a client/server chatroom generation system, ChatterBox, implemented in Java, is presented. The following components of ChatterBox are described: (1) the basic client/server system; (2) interface features; (3) configurability, including messaging format and use of profiles; (4) extensibility, including use of plug-in technology, implementation, and examples of use. Further developments are also considered. (MES)

Beyond Chat - New Generation Software for Real-time Discussion

Colin Charlton, Janet Little, Simon Morris and Irene Neilson,
Connect, Department of Computer Science, Foresight Centre,
University of Liverpool, Liverpool. L69 3GL, UK.
Tel:+44 151 794 8276, Fax:+44 151 794 8270
Email: {ccc, jml, fish, ien}@csc.liv.ac.uk

Abstract: The use of chatrooms as a forum for discussion is increasing in popularity. Consequently there is a need for generic software to be designed which is readily configurable to the requirements of particular domains and which is also extensible. This paper discusses the design requirements of such a system and presents a novel chatroom generation system, ChatterBox, implemented in Java.

1. Introduction

Since the invention of the WWW there has been an explosion of interest in the use of technology, in education. Much of this interest has focussed on the WWW as a vehicle through which to present information to students. Increasingly however, it has been recognised that in order to maximise the value of the infrastructure provided by the WWW in the service of education, *greater interactivity* is needed. On the one hand software is required that facilitates the engagement of the student with the education material and can be readily integrated into the infrastructure provided by the Web. An example of such software would be on-line simulations [Neilson et al 1996]. On the other hand, appropriate communications technology is required to support discussion about the educational material. Educational content requires to be reflected upon and debated if deep learning is to occur [Laurillard, 1993]. The focus of this paper is on communications technology.

The main forms of communications technology readily available on the Internet are email, newsgroups and chatrooms. Both email discussion lists and newsgroup technology such as Hypernews are widely recognised as educationally useful tools [McKendree and Mayes, 1997]. Chatroom technology by contrast has generally been regarded as a somewhat frivolous application. Traditionally, chatroom use has been viewed as a purely recreational activity, something that is done for fun, an opportunity to discuss a hobby with like-minded individuals. Increasingly, however, the potential educational value of chatroom technology is being recognised. Chatrooms may be used to promote learning by providing a forum for critical dialogue on a topic between peers. Chatrooms have the advantage over email of allowing multi-user discussion in real-time. They more realistically support the dynamics of traditional classroom based educational debate in the on-line environment. As societies increasingly promote distance education opportunities and the need to acquire new skills throughout a person's working life, the use of chatrooms as forums for discussion will undoubtedly increase. Large businesses might be expected to provide such facilities as part of the training provision made for a workforce to update their skills or indeed as part of an on-line customer support environment. Chatrooms can thus potentially serve a wide variety of useful commercial and educational as well as recreational purposes.

The disparate nature of the contexts in which chatrooms can be employed, requires that the technology used for chatroom generation be highly configurable to the requirements of a particular context of application. Chatrooms employed in a distance learning context can require a *differentiation of user privileges* between a instructor and a learner that does not apply in the chatroom of a football supporters' WWW site. Often one wishes to *restrict access* to the former but not to the latter type of chatroom. Where chatrooms are open to anyone there must however by a system for *regulating the number of users* who are on-line simultaneously in order to prevent server overload[1]. In some chatrooms, it may be desirous for regular users of a chatroom, such as loyal customers or senior students, *to be afforded special privileges* or for certain users, such as an instructors, to be able to *reserve a username* for their use only. In many, though not all, chatroom disruptive users can be a problem. It is thus often desirable for selected individuals to have *policing rights* in particular chatroom contexts

[1] Controlling the load on the server computer is particularly important where Java applets are involved. Security restrictions on applets forbid connection to servers other than that from which the Java classes were served. The chat server thus needs to live on the same machine as the web server.

in addition to the system being capable of *automatically monitoring* and *controlling* anti-social behaviour such as abusive language. In a learning context an individual may wish to hide their ignorance about a topic by communicating only with a trusted friend or with the tutor. There is thus a need of an *option for one to one conversation*, (referred to in this paper by the term *whispering*), that does not perhaps apply so much in a recreational context. There may also be a need for a chatroom to have *filtering capabilities* so that an individual can selectively chose not to receive messages from other users who are found to be irritating- for example fans from a rival football team in a sports'chatroom.

Generic software for chatroom generation also requires to be readily extensible. Most chatroom generation software is based on the client/server model. Increasingly it is implemented in Java with the server residing on the same machine as the WWW server and the client taking the form of an applet embedded in an HTML document[2]. Applets are however also increasingly used to extend the functionality of a WWW site. There is thus a need for chat client applets to be able to communicate with other applets on a WWW site in order to increase the 'interactivity' of the host WWW site. This point is best illustrated by an example. We developed a football commentator system, http://anfield.merseyworld.com/commentary/, which allowed details of live football matches of a the Liverpool football team to be broadcast around the world over the WWW along with text transcripts of a game 'as it happened'. This commentator system employed Java applets client side. Our webmaster wanted the transcripts of the match to be fed into the Liverpool football club's supporters' chat room, http://anfield.merseyworld.com/chat/detach.html, in order that supporters could see the commentary of a current match as part of the chat. A special interface had to be built to allow the commentary system applet to pipe its output into the chat system applet. A more generic solution than this is desirable.

Finally, many generic chat systems fail to adequately support the use of colour in their interface. Support for coloured text in the chat area would not only brighten the interface but would also be functionally useful. Colour coding can provide a visible means of filtering fast moving discussion and distinguishing between different types of message. Utilisation of icon buttons (instead of AWT text only buttons) would also help to customise the look-n-feel of the chat applet to a particular application context. Finally as Web Sites are frequently quoted in on-line discussions, any URLs within a user's message text should be clickable and operate in a similar fashion to web page hyperlinks.

This paper presents a Java based client/server system for chatroom generation, ChatterBox, which is both configurable and extensible and includes the aforementioned interface features.

2. The architecture of ChatterBox

2.1 The basic client/server system

ChatterBox is a generic system for chatroom generation written in Java. It employs a client/server architecture. However, while in many chat systems server side processing dominates this is not the case with ChatterBox. In ChatterBox, only essential filtering such as that required for security is conducted server side. Conducting filtering client side, as for example through the plug-in system we describe, has the advantage of allowing client systems with vastly different filtering requirements to co-exist on the one server without fear of incompatibility. The robustness of the Java server was tested using a simulated client which waits a random time before starting, then repeatedly connects/disconnects at random intervals, sending a random number of messages each time of a random length at random time intervals. This test client was designed to try to approximate the workload the chat server might have to service and to evaluate the number of simultaneous connections it could reasonably be expected to handle.

2.2 Interface features

The interface is designed to support the use of colour. 24 bit colour values are specified as a decimal integer.. Chat is in black (zero) by default but this can be altered to make a particular user's text stand out. For example the football commentary system which is broadcast into a supporters chatroom uses red as its text colour to make its messages distinctive. A lecturer's contributions to an educational discussion could be similarly flagged. Currently, however, the ChatterBox client does not allow an individual user to change the colour his/her messages will appear in on each client. This facility was originally made available but has been switched off as feedback from users when it was available was negative. Users did not want multi-coloured chat unless the change of colour had functional significance in a particular application context.

[2] An example of such chat generation software is Parachat, http://www.parachat.com/

Colour is also used to cue *whispered* messages. A user selects those names with whom s/he wishes to communicate and then clicks on the SET WHISPER button. This restricts the display of the message to only the named individuals. Whispered messages are prefaced by a line "[Whispered to: ...]" followed by the names of the recipients of the message and is shown in blue. To reply to this sub-group of individuals a receiver has to set their own whisper function to the appropriate names. The system does not do this automatically. The WHISPER function can be toggled on and off. It is a feature that is frequently employed by users.

The icons used by the chat applet can be made appropriate to a particular context by specifying the relevant image files through the ICONFILE param of the chat client applet Any URL quoted in the text of a message is automatically converted to a hyperlink. The normal user interface to the chat system is illustrated in Figure 1. If a user has superuser status the admin interface would be active and accessible through the admin user button. The admin interface allows a superuser to bar, ban or 'kill' users and to view chatroom statistics such as the load on the server.

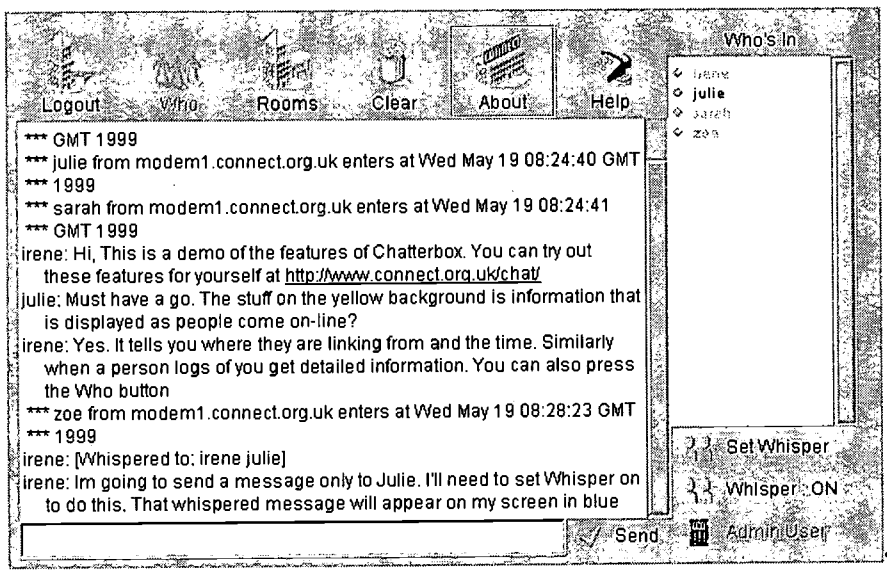

Figure 1: The ChatterBox Client Interface

2.3 Configurability

Messaging format:- To achieve a highly configurable system, a messaging format was required which would enable messages to be constructed from 'tagged' data sections (in much the same fashion as image formats like IFF and TIFF). Program code could then pick and choose which sections to respond to, ignoring those that were not understood. This would also allow multiple types of information to be encapsulated inside one message. For example the 'remove client' message sent to all clients when someone leaves a chat room could be accompanied by an announcement to that effect suitable for displaying on screen. The format chosen is sown below. All messages are sent in plain ASCII text format.

```
<tag> <data>|<tag> <data>| . . . . . |<tag> <data>
```

A pipe character separates each section. For this reason pipe is not allowed within any section - it was picked because it is a symbol that does not appear in English text and as ChatterBox is based around text communication its absence is unlikely to be noticed. A section begins with a tag name followed by a single space, then the data for that section. A real life example of this format taken from a session in a chatroom is shown below. Note that Laura is communicating with two people, one publicly (D.J.) and the other via whispering (Faisal). CHATMESG is the message itself, CHATWHISPER is a list of users to whom the message will appear if the whisper feature is active.CHATADDUSER is transmitted to all clients to introduce a new user. Lines without a USER section are assumed to be system messages. Other examples can be inspected at http://www.connect.org.uk/chat/

```
    CHATMESG fish from thirl.csc.liv.ac.uk enters at Tue Feb 23 17:22:18 GMT 1999|CHATADDUSER
fish thirl.csc.liv.ac.uk 919790536594 t
    USER D.J.|CHATCOLOUR 0|CHATMESG I can do it now, but they never asked me to in the test
    USER Laura|CHATCOLOUR 0|CHATMESG I have trouble reverse parking as well
    USER Laura|CHATCOLOUR 0|CHATMESG have u sent it yet|CHATWHISPER Laura Faisal
```

```
USER Faisal|CHATCOLOUR 0|CHATMESG I'll try to send it|CHATWHISPER Faisal Laura
CHATDELUSER Laura|CHATMESG Laura from usr217-kno.cableinet.co.uk leaves at Tue Feb 23
17:33:27 GMT 1999
```

Use of Profiles:- A capacity to tailor the functionality of a chatroom to the requirements of particular users could be achieved though the use of profiles. Profiles would also allow some users to be granted 'superuser status' giving them the authority, which itself could be varied, to censor disruptive behaviour via an interface built into the client. This would reduce the need to run a special administration programme server side to manually ban people. Policing by users would also complement the automatic censor system[3].

ChatterBox features an extensive configuration mechanism based upon a main configuration file. Within this the default user and room profiles are defined. A room profile defines variables such as the maximum number of users per room. The default user profile defines variables such as the period of inactivity tolerated, whether swearing, block capital typing, spamming etc are allowed, whether whispering (the selective sending of messages to particular individuals is allowed). This profile will be assigned to any user who logs onto ChatterBox without a specific profile of their own set up in the configuration file. Individual user profiles are created using the COPY attribute from the default profile (a sort of subclassing of the other profile) and are password protected. The following code illustrates the profile of a user who has 'superuser' status on the Anfield Football Chatroom.

```
<PROFILE=steve_d> COPY=!def_anfield PASSWORD=obscured KILLUSER=y
BARUSER=y BANUSER=n SEEWHISPER=y SUPER=y </PROFILE>
```

The user has the name 'steve_d' and a password to secure his username/profile from being used by anyone else. He has been given permission to 'kill' user connections and bar users, but not ban them permanently. He can see all whisper messages regardless of whether he is part of the target subset of users. The final attribute 'SUPER' affords him a red S symbol next to his name in the WHO'S IN list on the chat client applet and access to the ADMIN USER interface.

2.4 Extensibility

Use of Plug-in technology:-The key to making the system extensible was identified as creating the ability for plug-ins to interact with the input and output data streams of the applet. . Plug-ins could be regular classes loaded from the server or other applets resident on the same web page as the client applet . Plug-ins had to be capable of forming could form input-output chains, intercepting outgoing and incoming messages and altering or extending them if necessary. For example, a plug-in could be devised which scrambled the contents of any CHATMESG section on output and unscrambled them on input - ensuring a degree of privacy for dialog sent from client to client. Plug-ins could also be used to control the input/output to other plug-ins.

Use of plug-ins to extend the capabilities of a chat system has the major advantage of leaving the core code of the chat client/server application intact. When a specific feature is needed for a particular chat context, a plug-in is simply used to provide the feature. Customised filtering can also be provided through use of the plug-in system.

Implementation:-Within the ChatterBox system, the procedure for linking in plug-ins is reasonably straightforward. Upon start-up of the chat client applet, a MessageHandler class instance is created which scans the applet's PARAMs for occurrences of 'PLUG_INx' or 'PLUG_OUTx' (where x is an integer number beginning at zero and incrementing with each successive in/out parameter line). The value of each parameter which is found is then parsed. The data is split on the first colon into two fields. The first field is assumed to be the plug-in name, the second is assumed to be a parameter string.

If the plug-in's name begins with an asterisk the MessageHandler looks for an applet using that name. If it does not then the MessageHandler attempts to load and build a class instance of the named type. If the name is simply an asterisk on its own the chat client applet itself is used. All plug-in applet/classes must implement the *Pluginable interface* designed specifically for the ChatterBox plug-in system.

The plug-in class is initialised by passing it environment details such as whether it is being connected to the input or output chain and whether it is being used on a client or server. (Currently however only the ChatterBox client implements the plug-in system.) The parameter string, which formed the second part of the param value, is also passed. Each plug-in can use this string as they wish - ChatterBox makes no assumptions about its contents

[3] Automatic censors are common in chat systems. These monitor all communications and detect system abuses –such as foul language, the sending of messages in ALL CAPS and people vertically writing a message, one character per line. Persistent abusers of the system are generally banned on the basis of their IP addresses or hostnames. In some cases, where IP addresses are dynamically assigned, this might necessitate the banning of a range of addresses using a wildcard option

or purpose. This string is intended to allow individual configuration details for each plug-in to be specified from within the HTML document.

Two chains are formed, one for input and one for output. The plug-ins are linked in numerical order based upon the PLUG_INx / PLUG_OUTx value, from low to high. Once these chains are created they are passed any outgoing message from the client chat applet prior to it being transmitted and any incoming message prior to being processed by the client software itself. Each plug-in in turn is passed the message in a Hashtable format - with each section of the message stored hashed by its tag name.

This is where the sectioned/tagged format of ChatterBox's message system is particularly beneficial. Each plug-in can then append, remove or replace sections of each message before allowing it to be passed down the chain. They can deal specifically with those sections of the message that they understand and ignore all other content - including those added by other plug-ins previously on the chain. Once the end of the chain is reached the message is transmitted (for output) or delivered to the client (for input).

Examples of use: A series of demonstrations of the plug-in system and the capabilities of ChatterBox in general is provided at http://www.connect.org.uk/chat/. Two examples are included within this paper for illustration. The first example is of the Debug Plug-in. The Debug Plug-in simply displays to a TextArea AWT widget the entire contents of messages being passed in and out of the client chat applet. It is a useful plug-in in that it allows the actions of other plug-ins to be monitored. The Debug Plug-in is inserted into the input and output chain of the client chat applet simply by including the HTML fragment below.

```
<PARAM NAME="PLUG_IN0" VALUE="*Debug:In> ">
<PARAM NAME="PLUG_OUT0" VALUE="*Debug:Out> ">
```

Figure 2 is a screen dump of the client applet with the Debug plug-in running. The output generated by the Debug plug-in is displayed in the TextArea below the client chat applet display. Thus Debug OUT displays all chat messages transmitted between client and server in the ChatterBox system while Debug IN displays all messages sent by the server to the client, many of which will contain text for display within the chatroom.

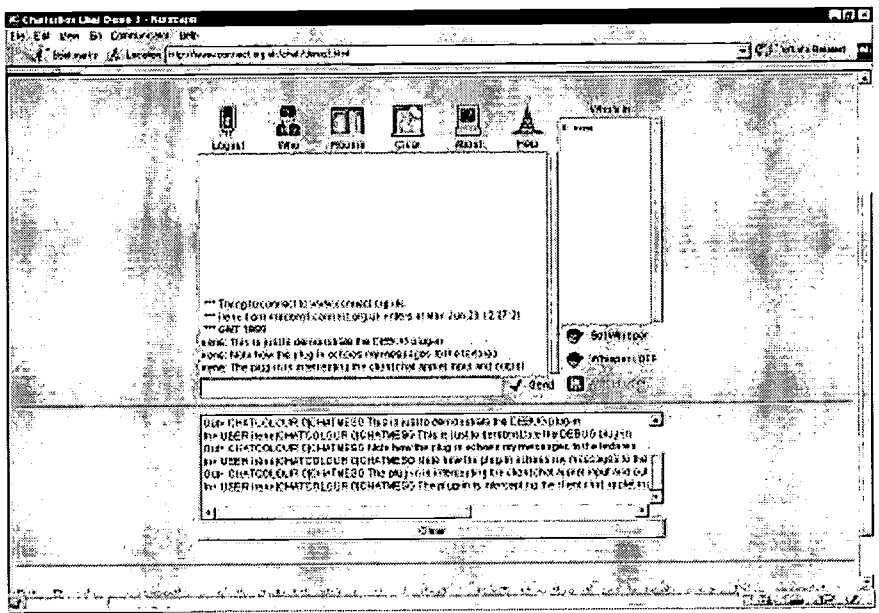

Figure 2: The Debug Plug-in displays messages passing out of and into the chat applet

In our second example, illustrated in Figure 3, a different plug-in, called 'Test' has been inserted in both the input and output chain of the applet by inclusion of the HTML fragment shown below.

```
<PARAM NAME="PLUG_IN0" VALUE="*Test:-">
<PARAM NAME="PLUG_OUT0" VALUE="*Test:-">
```

This plug-in displays the total number of characters etc sent in the CHATMESG section of the message for input and output. It also allows the user to modify the value of the CHATCOLOUR section to change the text colour used to render a transmitted message in all chat client. Different types of messages can thus be colour coded.

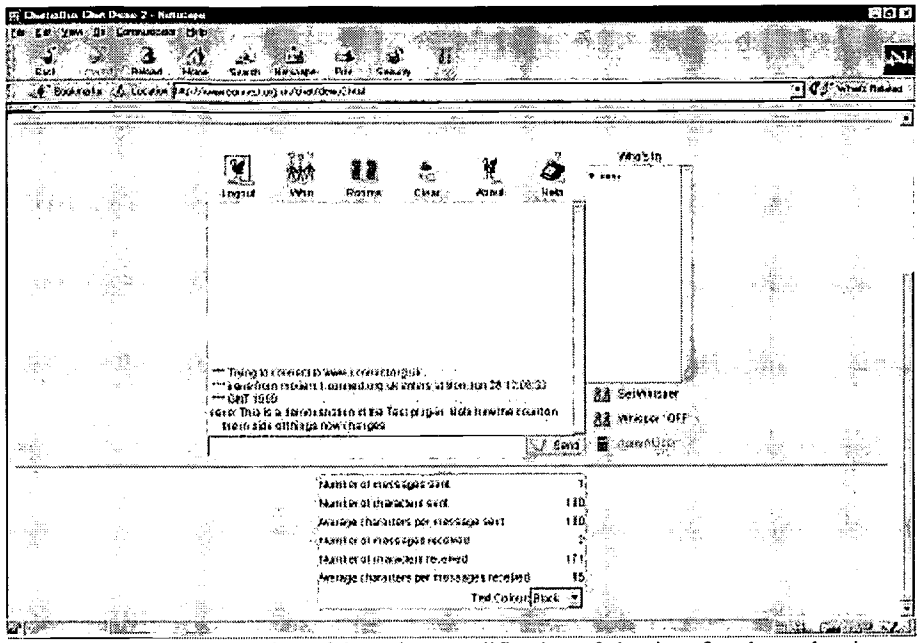

Figure 3: Demonstration of the colour modification of chat interface by a plug-in

3. Further developments

The plug-in system allows the client chat applet to be extended without the need to alter the client code itself. While the potential applications of the technology are considerable - effectively it offers outside programs complete control over the ChatterBox's chat client's input and output - there is also scope for improving the system.

Firstly, the fact that plug-ins are tied only to the input and output of the client is a major limitation in that they can only act when the client sends or receives messages. A system where by plug-ins can initiate their own outgoing messages without having to wait for the chat client to send would be more desirable. Originally the plug-in system was seen as serving only the client itself - but with the ability of applets to be used as plug-ins it became clear that this was unduly limiting. Because applets can be entire mini-applications they might need better network support than that provided by the chat client messaging system alone. Secondly, if a plug-in requires a user interface it must utilise a separate applet on the same web page as the client chat applet. It would be advantageous if plug-ins could supply their own UI components within the client's interface, supplementing or perhaps replacing the standard chat area component. Finally, the plug-in configuration is limited to a single string in the PLUG_INx / PLUG_OUTx param line - unless the plug-in reads it's own configuration file from the network. It would be useful if the configuration files associated with plug-ins could be stored as part of the main ChatterBox user/room configuration file.

Other unique features of the system could be further exploited. For example, in many chat system users are effectively anonymous - in that they had no permanent presence on the system. The introduction of user profiles users however effectively gives users an 'account' on the chat system. The personal information by the profiling system could be used in a variety of ways. For example if ChatterBox was deployed in a distance learning context, the ability to look up a lecturer's specialist interests, office and phone details during an on-line conference could prove useful. Highly configurable and extensible systems such as ChatterBox are needed to provide the flexible environments required by the growth of interest in distributed discussion groups.

4. References

Laurillard, D. (1993. *Rethinking University Teaching*. Routledge

McKendree, J. and Mayes, T. (1997). The Vicarious Learner: investing the benefits of observing peer dialogue. In *Proceedings of CAL'97*, Exeter, UK.

Neilson, I., Thomas, R., Smeaton, C., Slater, A., and G. Chand (1996). Education 2000: Implications of W3 Technology, *Computers and Education*, 26 (1) 113-122

U.S. Department of Education
Office of Educational Research and Improvement (OERI)
National Library of Education (NLE)
Educational Resources Information Center (ERIC)

NOTICE

REPRODUCTION BASIS

☒ This document is covered by a signed "Reproduction Release (Blanket) form (on file within the ERIC system), encompassing all or classes of documents from its source organization and, therefore, does not require a "Specific Document" Release form.

☐ This document is Federally-funded, or carries its own permission to reproduce, or is otherwise in the public domain and, therefore, may be reproduced by ERIC without a signed Reproduction Release form (either "Specific Document" or "Blanket").