To: MTB Distribution MTB - 401

From: Gary C. Dixon

Date: September 5, 1978

Subject: Processing of Trouble Reports

OUTLINE

---

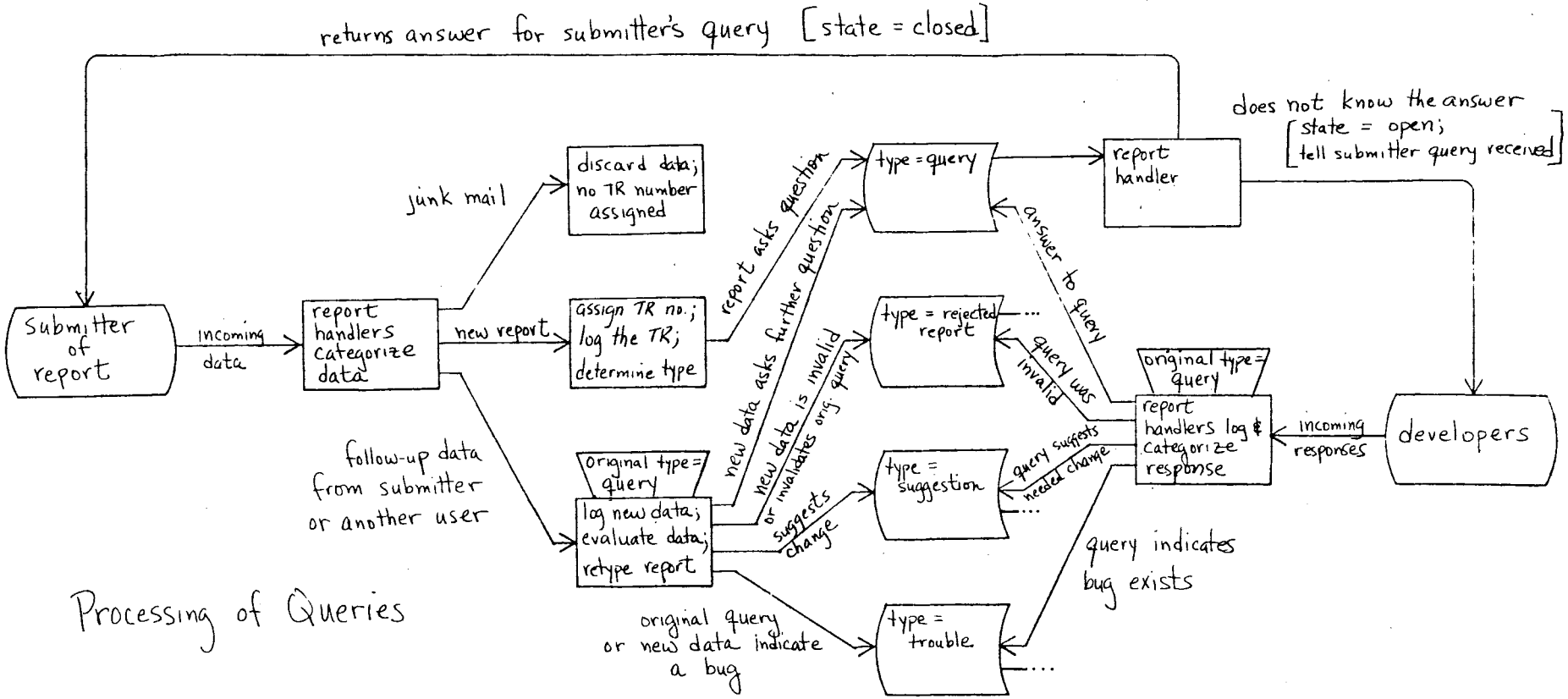
1. TROUBLE REPORTING PROCESS

The steps involved in processing a trouble report are described in the diagrams on the next four pages.

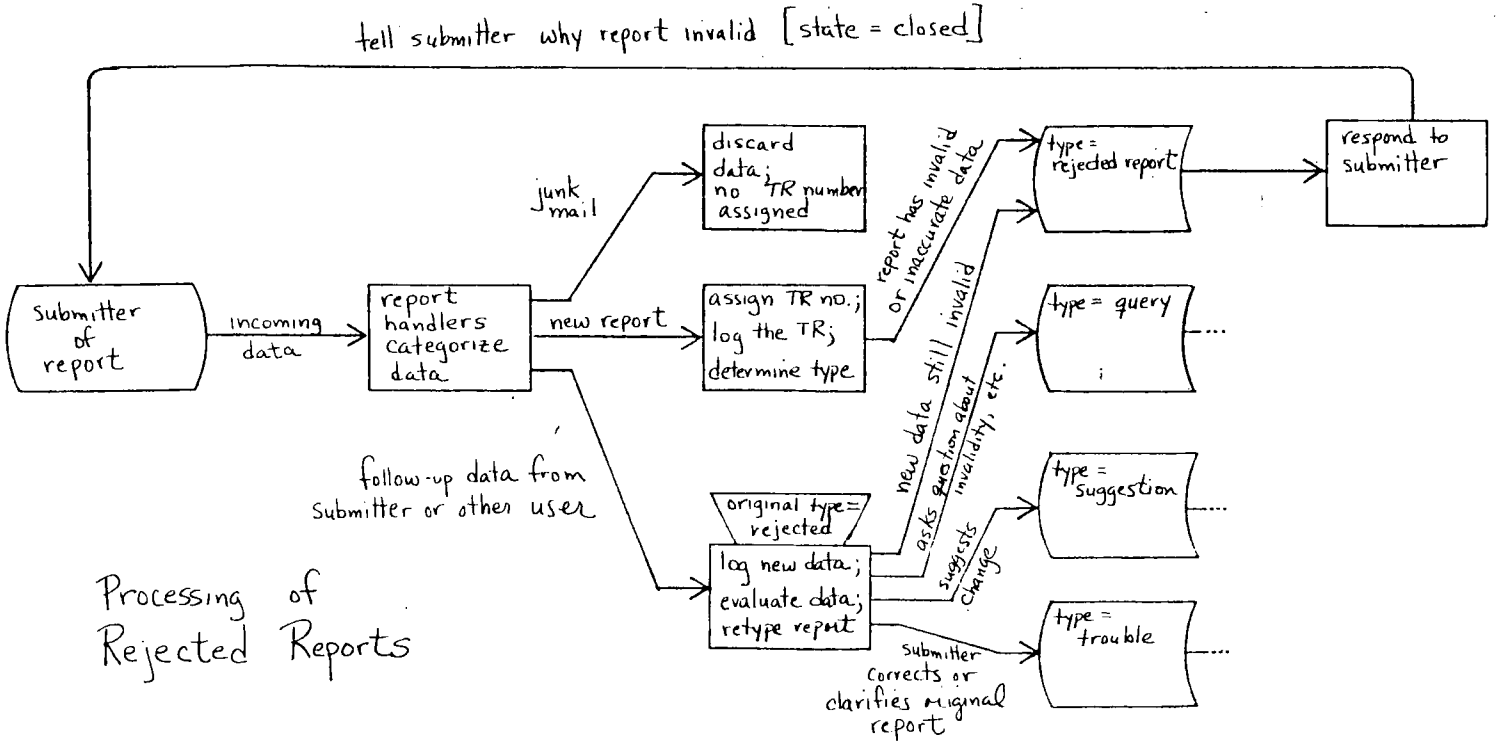
As the diagrams show, reports can be classed as one of four distinct types: a trouble report; a suggestion report; a query; a rejected report. All four types are referred to informally as trouble reports, but distinctions exist between the various types and each is processed in a different way.

A trouble report reports a problem with the operation of documentation of one or more parts of the system. A suggestion report suggests changes to the operation or documentation of one or more areas of the system. Queries ask questions about the operation or documentation of the system. A rejected report incorrectly states a problem or in some other way indicates a misunderstanding by the submitter of system operation or documentation. The diagrams which follow show how each of the four types of reports is processed.

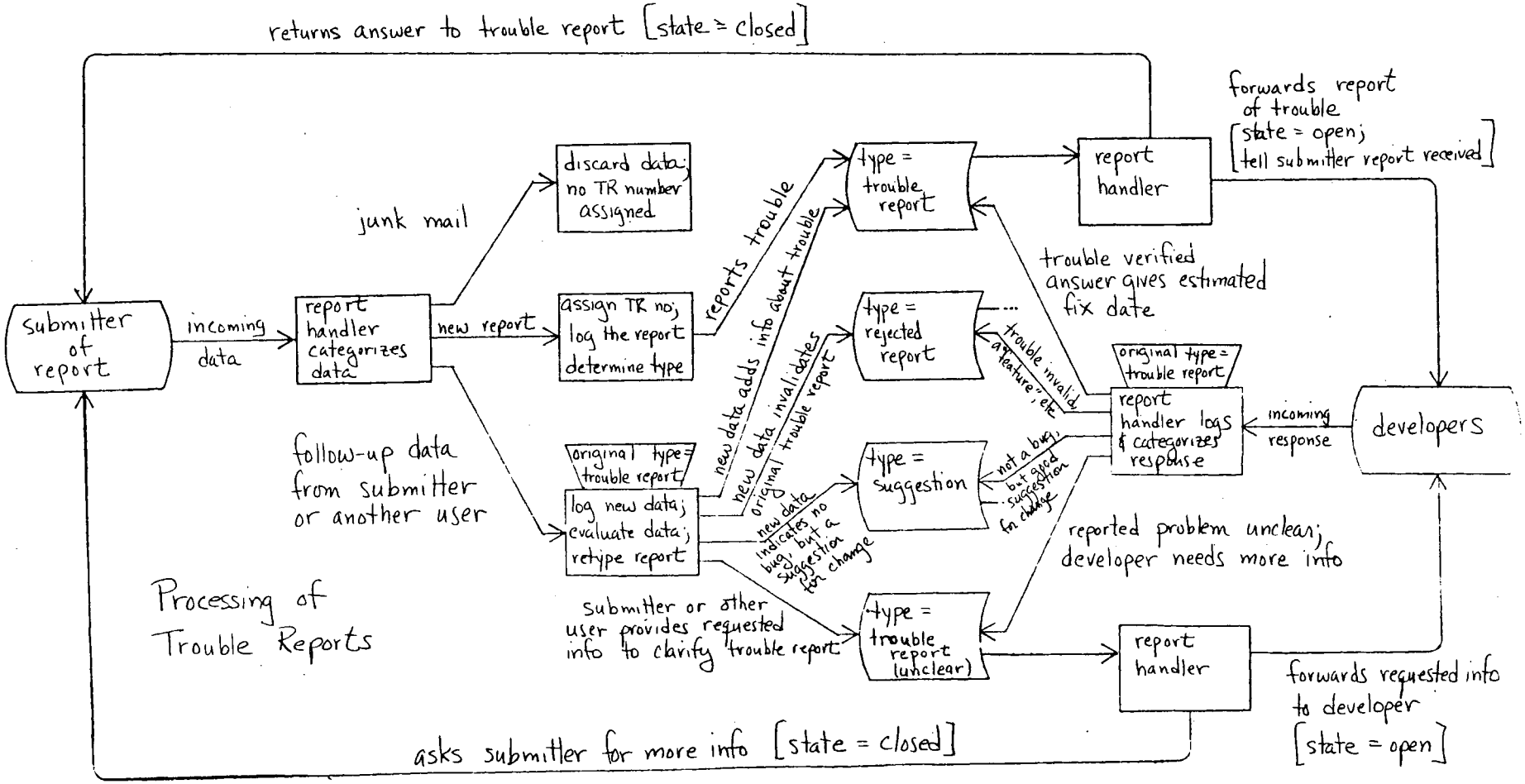
A given report may migrate from one type to another as interactions occur between the submitter, the processor of reports and the developers. The diagrams which follow illustrate how this migration can occur.

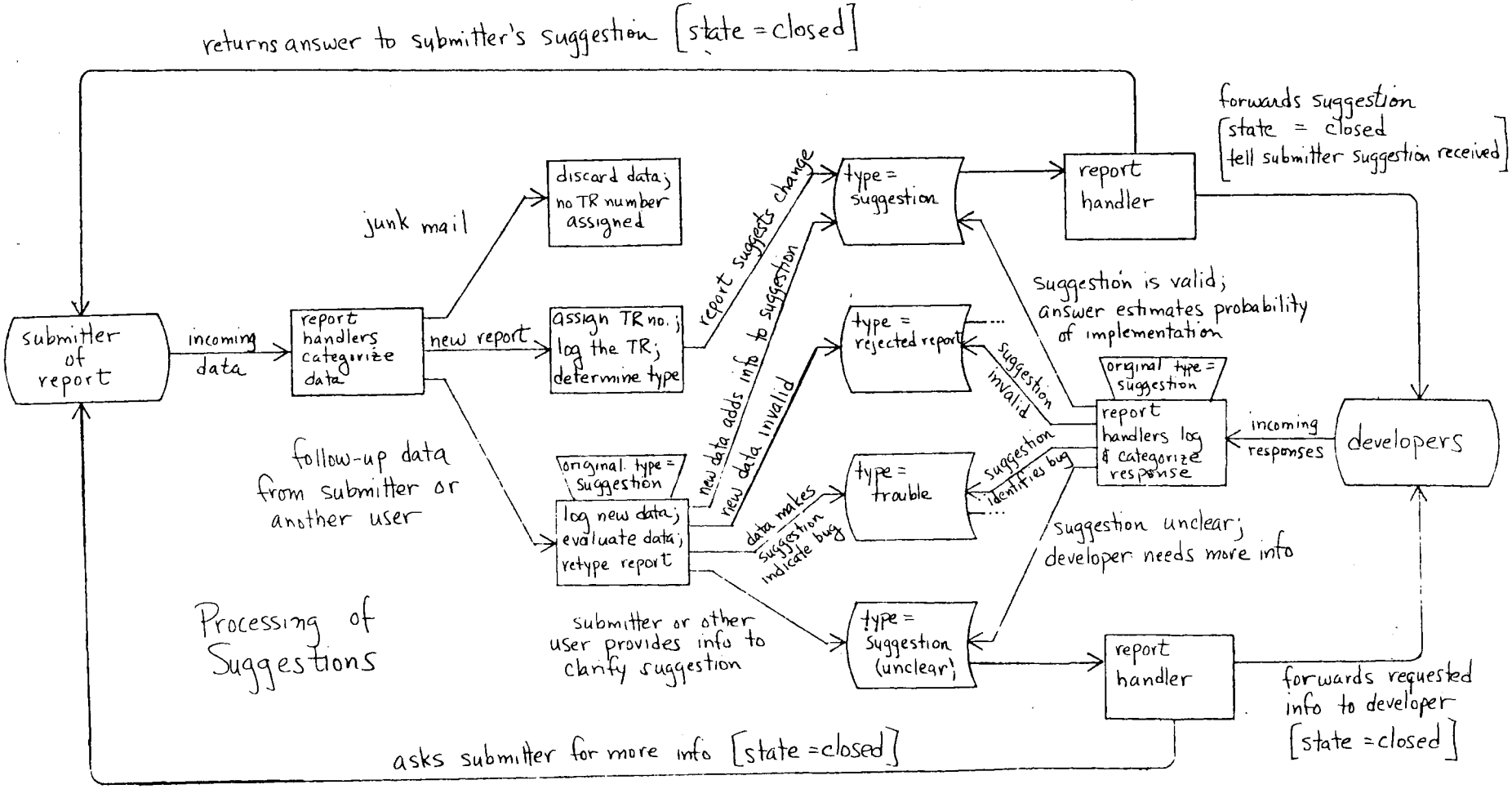


Processing of Queries



Processing of Rejected Reports


returns answer to trouble report [state = closed]
submitter of report
incoming data
report handler categorizes data
junk mail
discard data; no TR number assigned
new report
assign TR no; log the report determine type
reports trouble
type = trouble report
report handler
forwards report of trouble [state = open; tell submitter report received]
trouble verified answer gives estimated fix date
new data adds info about trouble
type = rejected report
trouble invalid, a feature, etc
original type = trouble report
report handler logs & categorizes response
incoming response
developers
new data invalidates original trouble report
follow-up data from submitter or another user
original type = trouble report
log new data; evaluate data; retype report
new data indicates no bug, but a suggestion for change
type = suggestion
not a bug, but a good suggestion for change
reported problem unclear; developer needs more info
Processing of Trouble Reports
submitter or other user provides requested info to clarify trouble report
type = trouble report (unclear)
report handler
forwards requested info to developer [state = open]
asks submitter for more info [state = closed]


returns answer to submitter's suggestion [state = closed]
submitter of report
incoming data
report handlers categorize data
junk mail
discard data; no TR number assigned
new report
assign TR no.; log the TR; determine type
follow-up data from submitter or another user
original type = suggestion
log new data; evaluate data; retype report
report suggests change
new data adds info to suggestion
new data invalid
data makes suggestion indicate bug
type = suggestion
type = rejected report
type = trouble
type = suggestion (unclear)
report handler
forwards suggestion [state = closed; tell submitter suggestion received]
suggestion is valid; answer estimates probability of implementation
suggestion invalid
suggestion identifies bug
original type = suggestion
report handlers log & categorize response
incoming responses
developers
suggestion unclear; developer needs more info
report handler
forwards requested info to developer [state = closed]
asks submitter for more info [state = closed]
submitter or other user provides info to clarify suggestion
Processing of Suggestions

2. CURRENT PROCESSING METHODS

The current trouble report processing system uses a series of ad hoc tools and data bases to maintain information about the various reports. As reports are received, the text of the report is stored as an archive component and a line for the report is entered in a status file. When an answer or additional information is received for a TR, this information is appended to the original problem statement in the archive component and an additional line is added to the status file.

2.1 Trouble Report Data Bases

The major data base of the current TR processing system is the TR archives. All information about each TR is placed in an archive component. This component serves as a complete log of all processing performed on the TR, except that test cases and written materials such as terminal logs and listings are kept separate from the component. Thus the TR archives provide a complete history of each TR.

A TR is placed in one of five archives depending upon how old it is, and what its current status is. Three status values are defined: open, indicating a trouble report for which no fix date has been given, or a query for which no answer has been given; closed, indicating a trouble report which has a fix date assigned or which is unclear and needs clarification, an answered query, a suggestion or a rejected report; under investigation, indicating that the TR has not had initial processing.

Open TRs and those under investigation are placed in TR_log.archive. Closed TRs between TR1501 and the latest closed TR are placed in TR_log_closed.archive. Closed TRs between TR0001 and TR0500 are placed in TR_log_old_1.archive; between TR0501 and TR1000 are placed in TR_log_old_2.archive; between TR1001 and TR1500 are placed in TR_log_old_3.archive.

The trouble reports are split into several archives primarily to reduce the time required to search the TR_log and TR_log_closed archives, containing the most recent open and closed TRs.

The status file is an ASCII segment containing one line for each TR transaction. Transactions include: initial processing of a new TR (originate); response from the developer to a TR (answer); request for additional information from the submitter (more info); additional information supplied by the submitter or another user (info). Each time a TR is processed for any reason, a transaction line is added to the status file. This file is used to track the status and current holder of a TR.

Test cases and data files associated with TRs are kept in a separate test case directory, and are identified by the number of their associated TR.

2.2 A Typical Report

A typical trouble report is shown on the next page, followed by its lines from the status file. Note that the report includes date of submission, date of initial processing (date originated), and date of each transaction. It includes the name of the developer holding the report at any given point. It includes the name, address and telephone number of the submitter. It may include small test cases, and scripts of test case execution.

The status lines include the TR number, date of transaction, type of transaction, current status and hold of the TR.

A Typical Trouble Report

Subject: TR1633
3) From: DJordan.SiteSA 05/26/78 1041.8 mst Fri (23 lines)

Dave Jordan...(303) 234-4810...US Geological Survey

MR 6.1:

We have encountered a problem in MR 6.1 which I haven't reported previously because I haven't had detailed information to report. I still have not obtained such information, but thought I would report the data I do know in case someone already knows about the problem.

We have seen instances where the FNP seems to forget about a port. Any input directed to the system through that port seems to be ignored until the user hits quit, at which point everything seems to return to normal. As I said, I haven't really got sufficient information on these problems, but, if I do get that information, I will pass it along. Urgency: normal

|---| |---| |---| |---| |---| |---| |---|

07/07/78: We have not seen your problem here on System M. I will forward you report to the developers for consideration. Gary Dixon

To: Coren (CISL)

07/26/78: << Mail from Coren.Multics 07/17/78 0737.8 mst Mon >>

This is PROBABLY the result of "exhaust" status (flooding the FNP with more input or status changes than it can handle) in which case the line drops out of receiver mode. In MR7.0 the handling of this condition is somewhat more elegant -- after a short time (~ 10 sec.) the system generates a QUIT for you. I think the condition is also somewhat less likely to arise in 7.0.

2.3 Current Tools

Tools used currently to process reports consist of a set of exec_coms, ted_coms, ed_coms (ed is my personal editor), a file searching command and many system commands. The next two diagrams illustrate how these tools interact with the trouble report data bases.

The cycle of processing reports begins with reports being formatted by the trouble_report command. This command queries the submitters for a description of the problem and then mails the formatted information to the Consultant's mailbox.

Every weekday afternoon, an absentee job copies the contents of the Consultant's mailbox into a file (trouble.[date]) which is dprinted for distribution at CISL. This file contains the unprocessed reports.

Periodically (once or twice a week during the current backlog, daily when there is no backlog of reports), the existing trouble segments are aggregated together by make_tr.ec into a single segment which the TR Administrators use to preview the incoming reports. These preprocessing steps are completed almost automatically, with little interaction of the TR Administrators.

Actual processing of reports begins when the TR Administrators (people who handle the reports) preview the aggregated reports to decide how to handle each report. Then do_tr.ec selects reports from the aggregate file for processing in sequence. It assigns each TR a number (using get_tr_no.ec, which is not shown in the diagram). Then it invokes ed (my editor) to edit the text of the report. Editing consists of correcting typos, changing information to clarify the problem statement, adding an initial response to the report (the originate transaction). The initial response may answer a query, verify that a problem does exist, or merely state that a problem is under investigation.

Then an ed_com (\b(rf)) is used to invoke runoff to format the report, limiting lines to 79 characters and filling lines when appropriate to reduce the line count of the report.

The report with initial response is then mailed back to the submitter to indicate its receipt and initial processing. The \bm (for mail) ed_com is used for this.

If the report is to be forwarded to a developer for processing, the name of the developer is appended to the end of the originate transaction, and \bm is used to mail the TR to the developer (on System M). If the developer does not regularly log in to System M, a copy of the TR is dprinted (by \bh for hardcopy) for the developer at CISL.

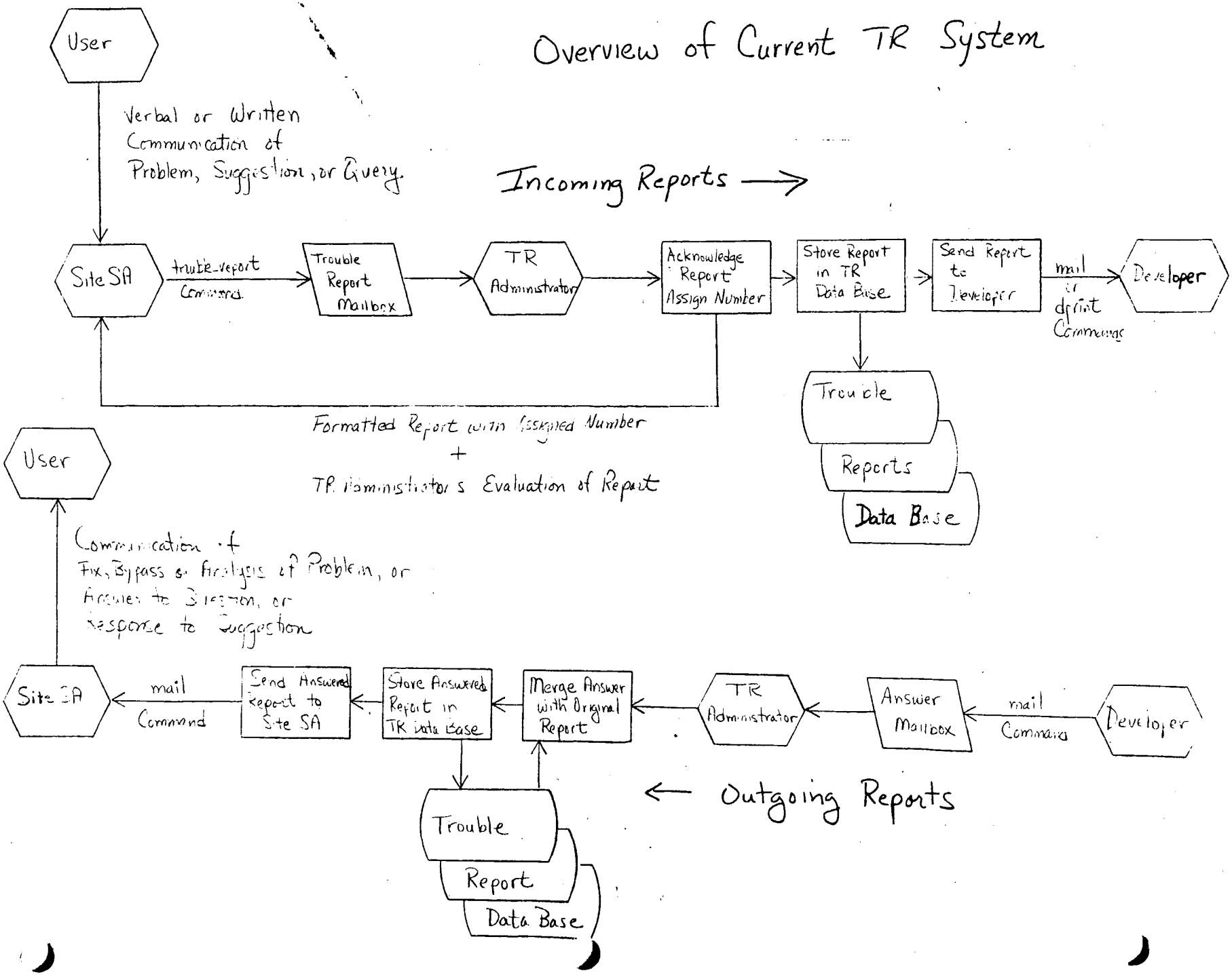
The report is then added to either the open or closed TR archive, depending upon its status. This is done by either the \bo (for open) or the \bc (for closed) ed_coms, which also: copy the report into a special directory containing reports processed during the day; delete the report when archived (unless \bh is used to dprint it, in which case the I/O Daemon deletes it after dprinting). A daily absentee job (not shown in diagram) uses library_print to combine all reports processed during the day. The job then dprints this for the TR Administrators to review, for Multics management to review, and for MIT's information.

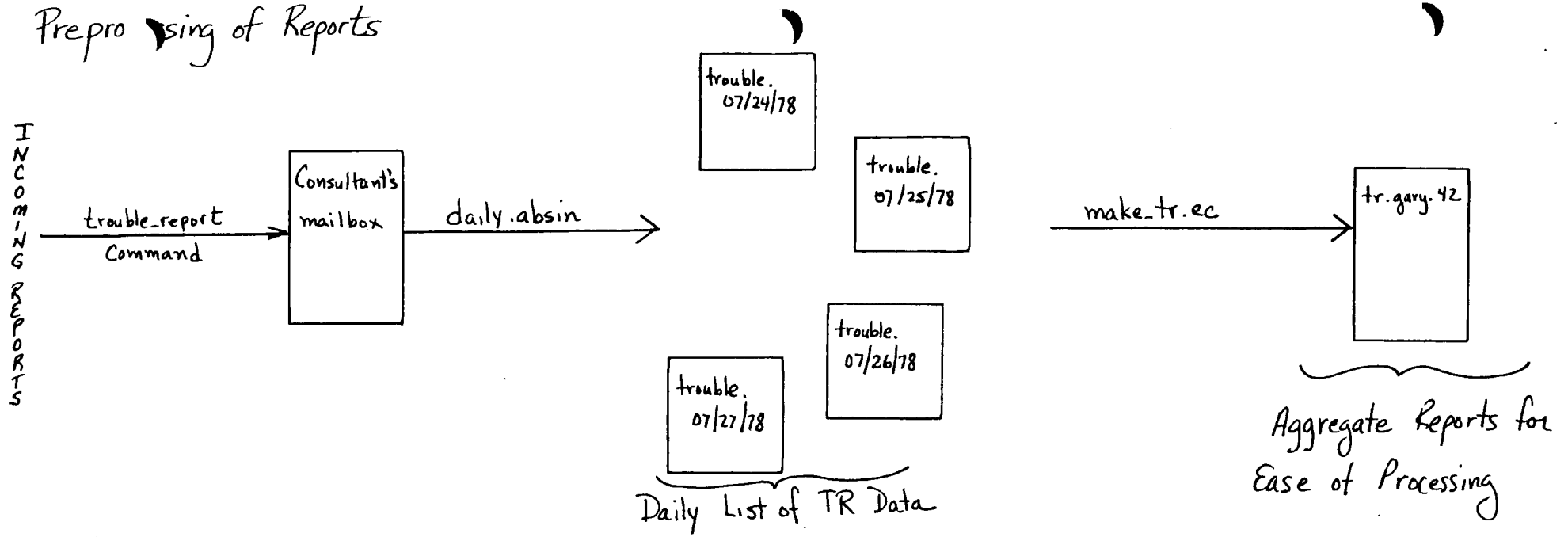
The final processing step is to place a line in the status file. status_tr.ec does this by querying the TR Administrator for transaction type, status and holder of the report. This completes processing of the initial report.

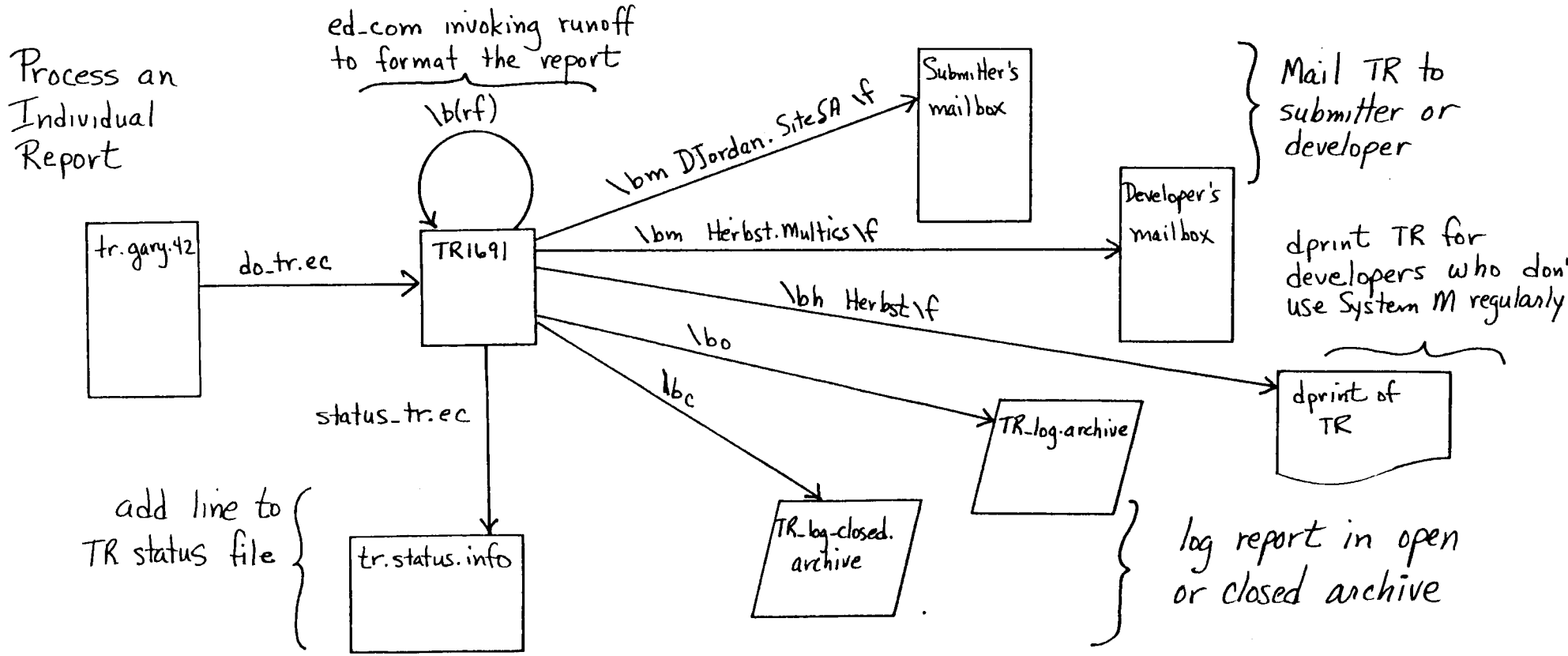
Answers for reports generally come to the TR Administrator's mailbox. The Author Maintained Library read_mail command is used to write each answer or additional piece of info for a report into a separate file. The complete report is extracted from the appropriate TR archive. Then ans_tr.ec combines the answer or added info with the complete TR, and invokes ed to allow editing, etc. The same ed_coms used in initial processing are used here to runoff the report, mail to submitter/developer, dprint for developer, archive the report in the appropriate archive. Two additional ed_coms are used: \bf to finish processing a report (move it from open to closed archive); \br to reopen a closed report.

TR reporting tools include: weeks_trs.ec, which generates a status list for TRs processed during the week; months_trs.ec, which does the same for those processed during the month; the search_file command, which lists all lines of the status file containing one or more keywords; a ted_com (archive.ted) which performs a keyword search of reports in the TR archives. The weekly and monthly status lists provide input data for TR Project status reports. The search_file command reports the transaction history of a given TR, or the list of TRs held by a given person, etc. The ted_com is used to find TRs reporting a given problem to see if a problem has been previously reported, or to print a TR without extracting it from the TR archives.

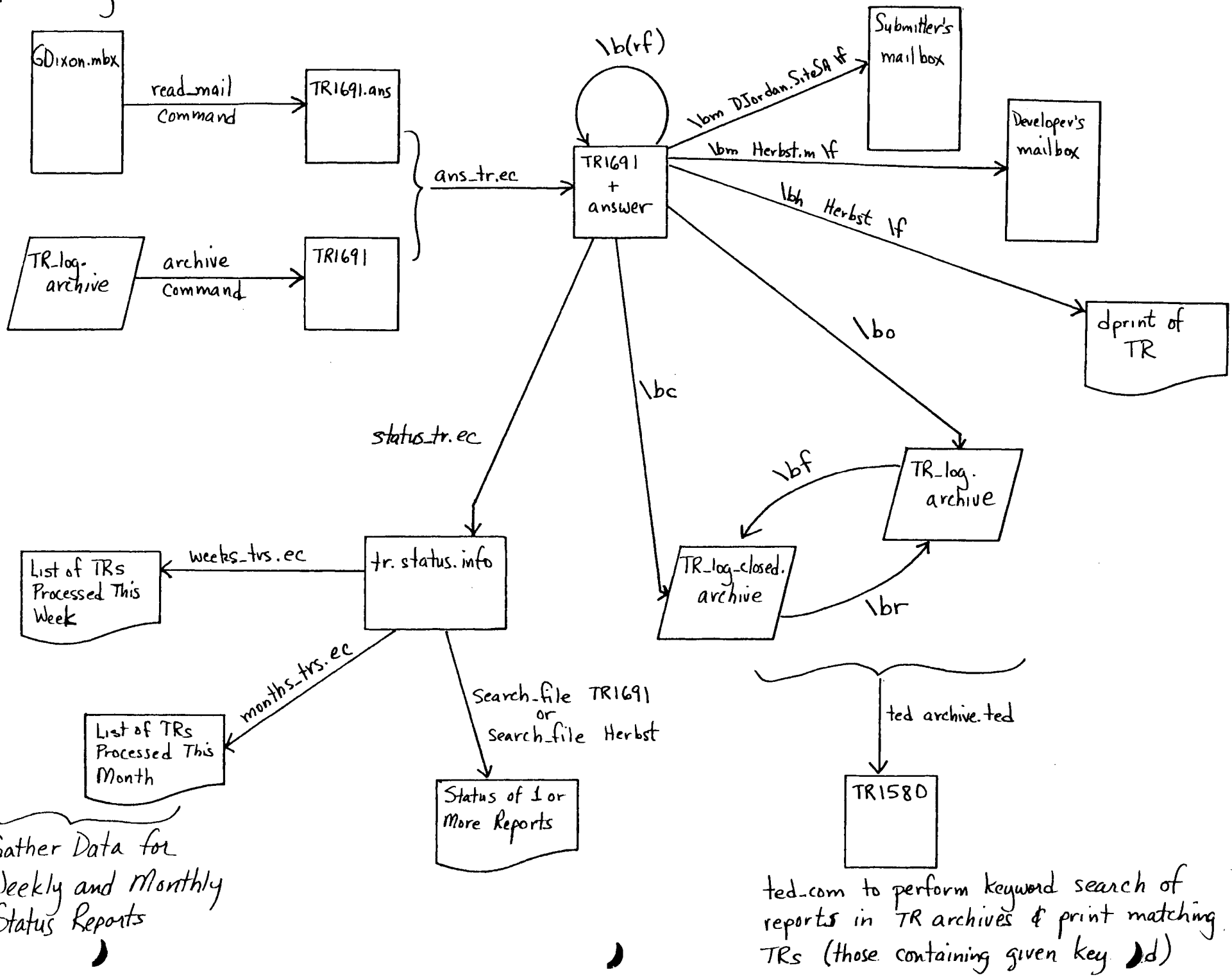
Overview of Current TR System




Prepro sing of Reports
INCOMING REPORTS
trouble_report
Command
Consultant's
mailbox
daily.absin
trouble.
07/24/78
trouble.
07/25/78
trouble.
07/26/78
trouble.
07/27/78
Daily List of TR Data
make_tr.ec
tr.gary.42
Aggregate Reports for
Ease of Processing


Process an
Individual
Report
ed_com invoking runoff
to format the report
\b(rf)
tr.gary.42
do_tr.ec
TR1691
\bm DJordan.SiteSA \f
Submitter's
mailbox
Mail TR to
submitter or
developer
\bm Herbst.Multics \f
Developer's
mailbox
\bh Herbst \f
dprint TR for
developers who don't
use System M regularly
\bo
dprint of
TR
\bc
TR_log.archive
status_tr.ec
add line to
TR status file
tr.status.info
TR_log_closed.
archive
log report in open
or closed archive

Processing TR Answers and Additional Info



2.4 Deficiencies of the Current Method

The current trouble report data bases and processing tools have a variety of deficiencies which we hope to correct in the future. The most important of these are summarized below.

Deficiencies of Report Submission

- When the submitter enters a trouble report, he should receive a number by which to reference the report in future correspondence. Currently, a number is assigned to the report when it is first processed. This may be several days or weeks after submission (depending upon the TR backlog). Meanwhile, the submitter (or other users) may have additional information to provide with the report, but will have no clear way to reference the report.

- The submitter is asked a series of questions about his name, address, phone number, and whether he can be answered by mail for each report submitted. This information should be stored in a table and merely updated when necessary.

- Some submitters like to be prompted for all information fields. Others like to enter data in their choice of order, without waiting for prompting. Typically, users who dislike prompting use the mail command to create and submit the report. It is easy for such users to omit pertinent information since they aren't being prompted. The submission command should accept keyword-identified free-format input, but should scan this input and prompt for any missing data.

- Many reports omit some necessary information, such as the name of the site at which the problem occurred, the Multics release on which the problem occurred, etc. The report submission command should prompt for such information, especially from Site SAs.

- The submission command should check for the existence and proper access settings on all test programs and data files. It should probably force access to the files, with the user's permission. The current program doesn't do this.

- The submitter usually wants to remember what reports he has submitted so that he can periodically check on the status of these reports. Currently, the submitter must maintain his own log of submitted reports. The report submission command should maintain this log, instead.

Deficiencies of Report Processing

- The report processing procedure is too complicated. There are many steps involved in processing a report, and it is easy to forget them. The report processing tools should prompt the TR Administrator for missing information and forgotten steps, based upon the kind of transaction being performed and the data provided.

- The report processing tools should be fast. The current tools are too slow and force the TR Administrator to wait for long periods while data base updating is taking place. The amount of Administrator wait time must be minimized.

- It is difficult for TR Administrators (especially new Administrators) to remember which developer is responsible for a given piece of software. There should be a responsibility data base which states who is responsible for modules in a given area of the system. While this data base will never be totally up to date, it will give a starting point in assigning reports to some developer, and it will handle most of the reports correctly.

- Some developers seldom log into System M, and therefore don't frequently check their System M mailboxes. Reports forwarded to these developers must be dprinted. Other developers log in frequently, and prefer receiving reports by mail rather than by dprint. Also, some developers keep a log of all trouble reports they handle. When additional information arrives for a report, they want to see only the new information. Other developers need to see all of the information each time they receive a report. It has been difficult to keep these preferences straight, so the current tools always transmit the entire report to the developer each time additional information arrives. Reports are mailed or dprinted (or both) to a developer based upon the TR Administrator's memory of that particular developer's System M usage wants and desires in this area.

- It is important for the submitter to know what problems are outstanding in a given area before reporting his problem, so that he can avoid duplicate reports and talk about his problem in terms of the existing knowledge of the problem. It is equally important for the TR Administrators to have this information so that reverification of known problems can be avoided, and so that current information about a problem is understood before trying to verify the new problem. There should be a fast way: to find all reports which deal with a given area and module of the system; to list a summary of these reports; and to select pertinent reports for perusal. The current tools provide only keyword searches of the entire text of a report, thus

confusing FORTRAN problems with those of debugging FORTRAN programs via probe. Also, since the tools search entire TR archives, they take many page faults and are very slow.

Deficiencies in Answering Reports

- When developers answer reports, they often omit pertinent information, such as the Multics release in which a problem will be fixed, the actual cause of the problem, any bypass for a problem, etc. They should be prompted for this information.

- The developers often forget to include the report number in their answer. This forces the TR Administrator to search the TR Data Base for a report which seems to match the answer. A developer should not be able to answer a report without giving the report number.

- Often the answer to a report is sent to the wrong mailbox, and is thereby lost or delayed in processing. A command should exist which will send an answer to the appropriate place without developer intervention.

The same comments apply, as well, to adding information to a report. Often the submitter or another user who has read a report wants to provide additional information with the report. There should be a command interface which supports this function, asks the appropriate questions, and sends the information to the correct mailbox.

Deficiencies in Reporting on Trouble Reports

- We need tools to report on the status of open TRs (and TRs in other categories) so that developers and management can better control the correction of problems in Multics. The current tools don't provide sufficient information to create such management reports.

- Customers have asked that a list of known problems be distributed with each Multics release, and that updates to this list be distributed between releases. We currently have no mechanism for providing such problem information organized in a reasonable way (by release, area of the system, module, etc) in a timely manner.

3. NEW TROUBLE REPORT DATA BASE

To solve the deficiencies listed above, the information in a trouble report must be categorized into individual data fields, and stored in a data base whose records are keyed on selected data fields. An MRDS data base is appropriate for several reasons.

- The first few versions of any such data base are experimental. The data is likely to be improperly organized, with some pertinent data fields omitted. An MRDS data base facilitates data base reorganization with minimal impact on the tools which access that data base. Also, an MRDS data base is easier to create, and to check out prior to writing any data base interfacing tools.

- The data base development group needs a real, live MRDS data base for testing purposes. The TR data base can be used for this.

- Customer's will think we lack confidence in MRDS if we choose an alternative implementation for an application which is an obvious MRDS candidate.

It is my feeling that the selection of an MRDS data base over a straight keyed file implementation will not take any longer to develop, will be easier to debug and maintain, and will have the advantages of a LINUS interface for data base queries. For this reason, an MRDS data base has been chosen as the implementation medium.

Solution of the deficiencies also requires new or improved tools to: submit trouble reports; answer trouble reports; process reports; scan existing reports; produce statistics on existing reports. These tools are described in Section 4.

3.1 Data Base Contents and Organization

The figure on the next page illustrates the MRDS data base planned for trouble reports. A detailed description of this data base is included in Appendix A in create_mrds_db source language. Appendix B shows PL/I include files describing the relations in the data base.

The data base include 19 kinds of relations: 13 containing data from trouble reports; 1 registering sites; 1 registering site SAs; 1 registering users; 1 registering developers; and 2 registering various system modules and their maintainers. The need to register the users of the data base, and system modules covered by reports in the data base was discussed above under "Deficiencies of the Current Method". The actual data in a trouble report is described further below.

Information is a report is divided into two categories: general information about a problem, query or suggestion; priviledged information about the problem, query or suggestion. Most information in a report falls in the general category, and can be distributed to developers, site SAs, users, prospective customers, and management. Some information is private within the Multics project, should not be released to the field, to site SAs, users or prospective customers. This information falls in the priviledged category.

```
/* ********************************************
   *                                          * LEGEND:
   * BEGIN tr_log.table                       *
   *      created: 09/19/78  1538.9 mst Tue   *          ---------------/  /--------
   *           by: create_mrds_dm_table (1.0) *          |*          || /  /  |       |
   *                                          *  relation | Attribute |        |       |
   * Data model >udd>m>gd>trouble_reports>tr_log *       | Data Type |        |       |
   *      created: 09/19/78  1536.8 mst Tue   *          |-----------|--/  /--|-------|
   *      version: 3                          *                          /  /
   *           by: GDixon.SysMaint.a          *          * = Key Attribute
   *                                          *          I = Index Attribute
   *                                          *
   ******************************************** */
```

report	* no	date_orig	type	site	release	sa	pers	proj	summary	symptom
	char(8)	char(35)	char(16)	char(32)	char(8)	char(22)	char(22)	char(9)	char(256) var	char(9600) var

	urgency	holder	status	mr_fix	date_mod	tran_no	path_no	misc_no	mod_no
	char(16) var	char(12)	char(16)	char(8)	char(35)	fixed bin	fixed bin	fixed bin	fixed bin

rpt_priv	* no	hold_pers	hold_proj	real_mr_fix	priority
	char(8)	char(22)	char(9)	char(8)	char(4)

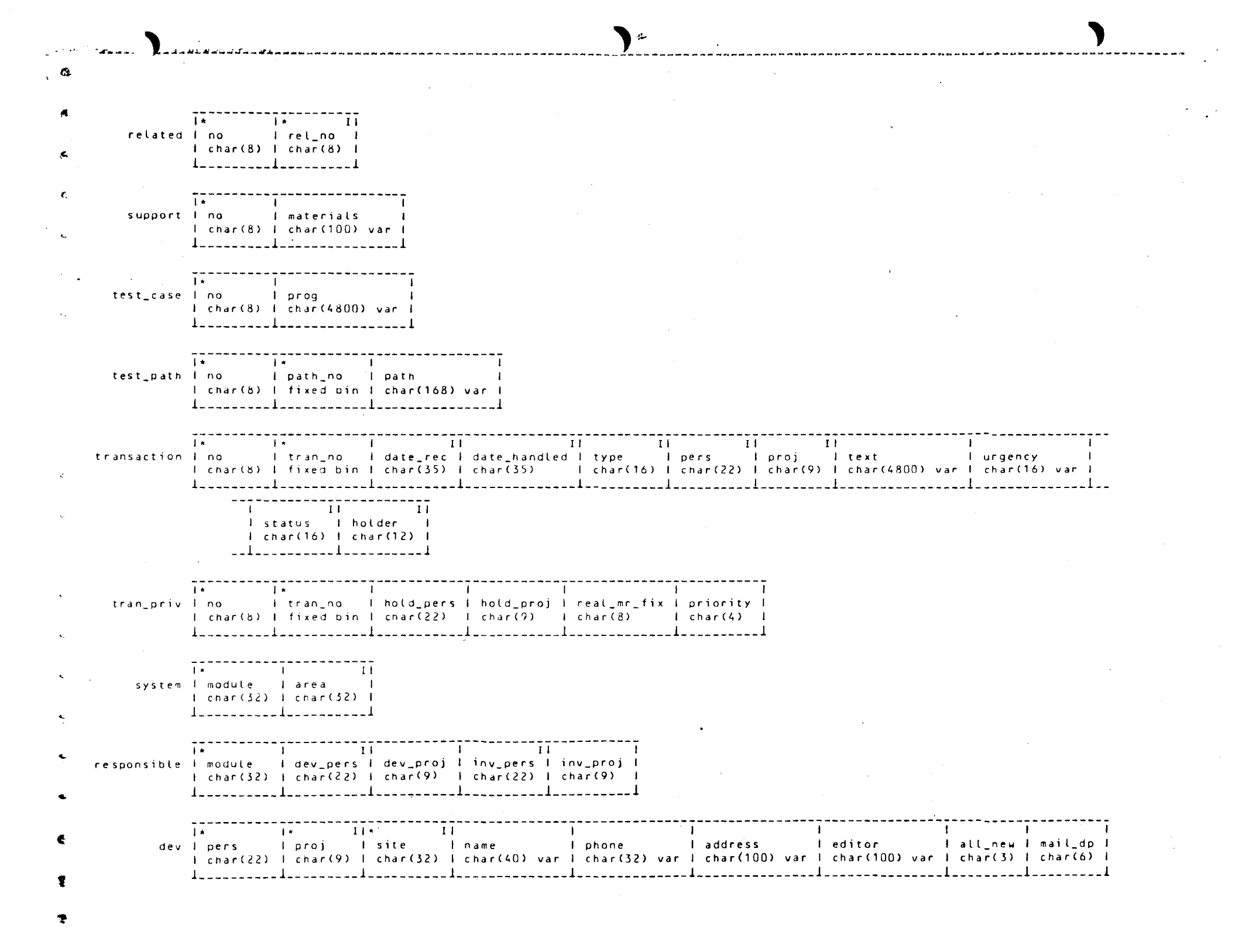
bypass	* no	date	text
	char(8)	char(35)	char(4800) var

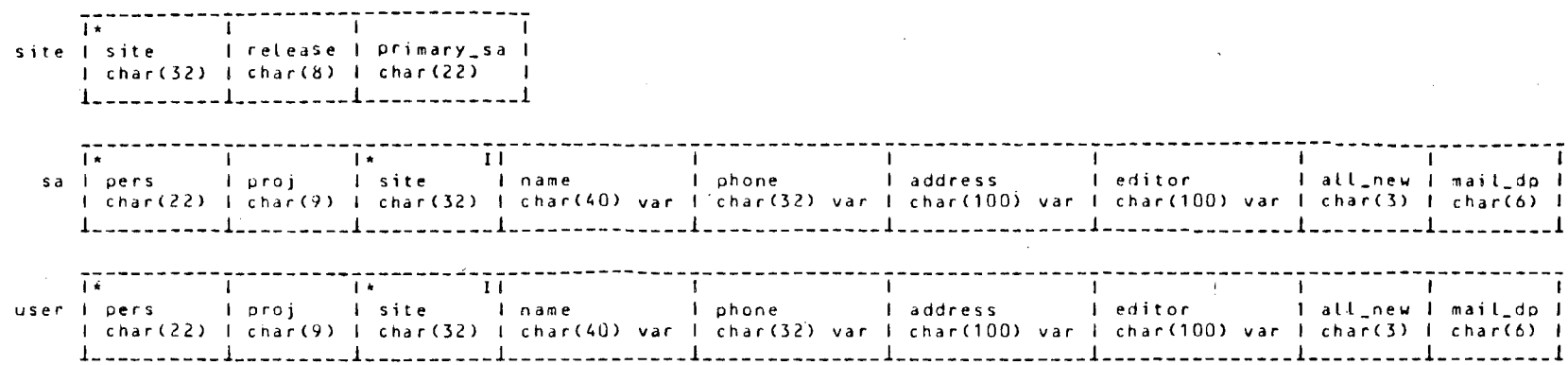
cause	* no	date	text
	char(8)	char(35)	char(4800) var

fix	* no	date	text
	char(8)	char(35)	char(4800) var

misc	* no	* misc_no	date	text
	char(8)	fixed bin	char(35)	char(4800) var

module	* no	* mod_no	area	module
	char(8)	fixed bin	char(32)	char(32)


related
no
char(8)
rel_no
char(8)
support
no
char(8)
materials
char(100) var
test_case
no
char(8)
prog
char(4800) var
test_path
no
char(8)
path_no
fixed bin
path
char(168) var
transaction
no
char(8)
tran_no
fixed bin
date_rec
char(35)
date_handled
char(35)
type
char(16)
pers
char(22)
proj
char(9)
text
char(4800) var
urgency
char(16) var
status
char(16)
holder
char(12)
tran_priv
no
char(8)
tran_no
fixed bin
hold_pers
char(22)
hold_proj
char(9)
real_mr_fix
char(8)
priority
char(4)
system
module
char(32)
area
char(32)
responsible
module
char(32)
dev_pers
char(22)
dev_proj
char(9)
inv_pers
char(22)
inv_proj
char(9)
dev
pers
char(22)
proj
char(9)
site
char(32)
name
char(40) var
phone
char(32) var
address
char(100) var
editor
char(100) var
all_new
char(3)
mail_dp
char(6)


site
site
char(32)
release
char(8)
primary_sa
char(22)
sa
pers
char(22)
proj
char(9)
site
char(32)
name
char(40) var
phone
char(32) var
address
char(100) var
editor
char(100) var
all_new
char(3)
mail_dp
char(6)
user
pers
char(22)
proj
char(9)
site
char(32)
name
char(40) var
phone
char(32) var
address
char(100) var
editor
char(100) var
all_new
char(3)
mail_dp
char(6)

Every report begins with a 'report' tuple. (1) This contains the following basic information about the report.

no a number which identifies the report. It is of the form: pic"xxx99999" where xxx identifies the site at which the report was submitted. For example: phx00145 or mit00258.

date_orig gives the date on which the report was submitted by the user, in the form: year-mm-dd__HH:MM:SS.UUUUUU_z_da (the format returned by the new calendar_clock active function). Note that dates in this format sort into chronological sequence when sorted by ASCII collating sequence.

type type of report: suggestion, trouble, query or rejected.

site site at which reported problem, etc occurred.

release Multics release on which reported problem occurred.

sa Person_id (at the site where report was submitted) of Site SA reporting the problem, suggestion, etc. The Site SA must be registered for the site named above.

pers Person_id (at site of problem occurrence) of user on whose behalf the Site SA is submitting the problem; or Person_id (at submission site) of user (not a Site SA) submitting the problem for himself.

proj Project_id of aforementioned user.

summary gives the briefest possible (1 line) description of the report. This description is used in TR summaries to identify the content of the report.

symptom is the body of the original report, describing the symptoms of a problem or the text of a suggestion or query.

urgency Urgency of report, as specified by submitter. It is unclear how this field could/should be used.

---

(1) For readers unfamiliar with MRDS terminology, a tuple can be thought of as a record in a file. The file is called an MRDS relation. Fields in the record are called attributes.

holder identifies who is currently working on a given report, in general terms. This attribute may be blank (no one working on a closed report), or it may be: submitter, developer, investigator.

status is the current status of the report. It may be: open, closed, or under investigation.

mr_fix is the Multics release in which a problem will be fixed, or a suggestion implemented. This attribute is only present for some closed reports.

date_mod is the date on which the last transaction occurred for this report. It is stored in the same format as date_orig.

tran_no number of transactions which have occurred for this report. Every report has at least one transaction, the originate transaction. See discussion of the 'transaction' relation below for details.

path_no number of pathnames of test cases and data files associated with this report. Must be a nonnegative integer.

misc_no number of miscellaneous items of information associated with this report. Must be a nonnegative integer.

mod_no number of area/module combinations assocated with the report. Must be a nonnegative integer.

The 'rpt_priv' relation contains the priviledged information basic to all reports. It includes the following attributes.

no report number (related tuples in tr_priv to those in tr relation).

hold_pers
the Person_id (at the submission site) of the developer currently holding a report when the holder field = "developer". The Person_id of the investigator when holder = "investigator".

hold_proj
Project_id of the developer or investigator

real_mr_fix
interim Multics release in which a problem will be fixed or suggestion implemented. This is priviledged because knowledge of interim releases (such as MR 6.5) cannot be distributed to the field.

priority Priority level for fixing problem or implementing suggestion, as assigned by management. It has the form: pic"99v.9"

The 'module' relation describes which area(s) of the system are affected by the problem, and more specifically, it names the modules which are affected in those areas. Typical areas might be: commands, run time environment, languages, answering service, supervisor, administrative ring, communications, metering tools, library maintenance tools, system programming tools, etc. We can invent as many system areas as are deemed necessary and useful. Since system areas subdivide trouble reports into major groupings, it will probably be useful to have as many, clearly-defined descriptively-named areas as possible. Then fewer probes of the TR data base will be required to find a given report.

The 'cause', 'bypass', and 'fix' relations describe the actual cause of a problem (as opposed to its symptoms), a bypass for the problem until it is corrected, and the fix for the problem. None, some, or all of these relations may be associated with a given report. In general, a fix relation will be present only if the fix is short, and can described simply as a source language change. This potentially becomes an ad hoc mechanism for distributing small fixes to the field, though it is not particularly intended for this.

The 'support' relation names any supporting materials (listings, terminal output, etc) which the submitter may be sending in support of his report. This relation is optional, and probably won't appear for most reports.

The 'related' relation names reports related to the current report. This may perform the function of mapping old TR9999 numbers into the new numbers, or of group several reports covering the same general topic.

The 'test_case' relation contains the source for a small test case which illustrates the problem. The test case is actually stored in the data base. The 'test_path' relation stores the pathname of a larger test case or data file. Only one test_case relation may be associated with a given report, but many test_path relations may be given.

The 'misc' relation contains the text of miscellaneous information associated with the report.

Each time a report is processed, a 'transaction' relation is added for the report. This relation contains the following, general category information.

no — the report number.

tran_no — the number of this transaction. Transactions are numbered sequentially, with 1 being the originate transaction, etc. The total number of transactions associated with a report is stored in prt_priv.tran_no.

date_rec — date on which this transaction was received.

date_handled — date on which this transaction was processed by TR Administrator.

type — type of transaction. This may be: originate, answer, more info, info.

pers — Person_id of user supplying info in an info type transaction.

proj — Project_id of this user.

text — text of the transaction. For originate: TR Administrator's evaluation of the report. For answer: text of the answer. For more info: description of the additional information required. For info: the requested information.

urgency — urgency associated with report as a result of processing this transaction.

status — status of the report, as result of processing this transaction.

holder — holder of the report, as a result of processing this transaction.

The priviledged information which is part of a transaction is stored in the 'tran_priv' relation. This relation contains the following attributes.

no the report number.

tran_no the transaction number.

hold_pers
the Person_id (at the submission site) of the developer or investigator associated with this transaction.

hold_proj
Project_id (at the submission site) of the developer or investigator.

real_mr_fix
fix release associated with report as a result of processing this transaction.

priority priority associated with report as a result of processing this transaction.

The remaining relations are registration relations. The 'dev' relation gives the Person_id and Project_id (at the submission site) of Multics system developers, their primary site of location, their name, phone number, address, the way to invoke their default editor used when submitting or answering reports, an all_new indicator telling whether to send all or only new parts of each report, and a mail_dp indicator telling whether to mail or dprint reports or both.

Similar information is maintained in the 'user' relation which registers users referenced in the report and transaction relations; also in the 'site_sa' relation which registers Site SAs. Since these users are primarily off-site, reports are always mailed to them and no mail_dp indicator is maintained.

The 'site' relation registers the site names, their current release, and their primary Site SA.

The 'system' relation registers the names of the areas of the system, and the modules included within each area.

The 'responsible' relation registers which developer is responsible for maintaining modules of the system. It also records which local person is charged with initially investigating a problem if the TR Administrators are unsure the report is correct.

It is interesting to note that the data base described above serves many (but not all) of the same functions as bug files describing bugs in a given area of the system. It will admirably provide information on outstanding system problems which our customers have request. However, it does not and should not replace the bug lists currently maintained by project leaders and other developers. These bugs lists, maintained by the developer, containing information he requires in a format he specifies, are still the best way for a developer to track problems in his software..

3.2 Format of a Trouble Report

The new trouble report format is illustrated on the next page by reformatting the typical trouble report which was shown above.

New Format of Typical Trouble Report

Subject: phx00001 (trouble report)
Modules: execute_epilogue_ (run time)
Summary: Loop in execute_epilogue_ at process termination when quits not allowed

Sympton:
We have encountered a problem in MR 6.1 which I haven't reported previously because I haven't had detailed information to report. I still have not obtained such information, but thought I would report the data I do know in case someone already knows about the problem.

We have seen instances where the FNP seems to forget about a port. Any input directed to the system through that port seems to be ignored until the user hits quit, at which point everything seems to return to normal. As I said, I haven't really got sufficient information on these problems, but if I do get that information, I will pass it along.

Status: closed (fixed MR 7.0) as of 1978-07-26__14:43:27_mst_Wed
Entered: 1978-05-26__10:41:48_mst_Fri
Site: USGS-Denver (running MR 6.1)
SA: Dave Jordan (DJordan.ssa)
US Geological Survey (303-234-4810)
For: Ken Runyon (Runyon.GS @ USGS-Denver)
USGS Computation Section (X 4456)
Urgency: normal

Fixed: MR 6.5
Priority: 12.0
==========

TRANSACTION 1: ORIGINATE by Gary Dixon (GDixon.sm)

We have not seen your problem here on System M. I will forwarded your report to the developers for consideration.

Status: open
Holder: developer
Handled: 1978-07-07__10:58:44_mst_Fri

To: Robert Coren (Coren.m @ CISL)
Priority: 8.0
==========

TRANSACTION 2: ANSWER from developer

This is probably the result of "exhaust" status (flooding the FNP with more input or status changes than it can handle) in which case the line drops out of receive mode. In MR 7.0, the handling of this condition is somewhat more elegant. After a short time (~10 sec), the system generates a QUIT for you. I think the condition is also somewhat less likely to arise in MR 7.0.

Status: closed (fixed MR 7.0)
Received: 1978-07-17__07:37:48_mst_Mon
Handled: 1978-07-26__14:32:34_mst_Wed

From: Robert Coren (Coren.m @ CISL)
Priority: 12.0
Fixed: MR 6.5
==========

One thing is apparent at the outset. The trouble report is longer in the new format than in the old. This is due primarily to the added information given in each report.

The report begins with a summary section giving:

- the report number (in a read_mail subject line)
- type of report (trouble, suggestion, query)
- system module in question (and area of system in which module resides)
- brief summary of report.

Next comes the body of the report, including paragraphs describing the symptoms or suggestion or query. Two blank lines separate the symptoms from sections which follow. Though not shown in the sample above, the report body can also include the following optional sections:

- a cause section, describing actual cause of program if not apparent from the symptom statement
- a bypass section, giving a bypass for the problem
- a fix section, giving any fix for the problem
- a test case section, giving a short test case, or pathnames of longer test cases or data files
- a supporting materials section, indicating what supporting materials are being transmitted to the TR Administrator under separate cover.
- a miscellaneous section, containing miscellaneous information associated with the report.

These sections comprise the body of the report.

Following the report body is the major statistics section, giving: current status of the report, including fix release if status is closed; date report entered; submitter's site and release; submitter's name, User_id at the site on which report was entered (system M or MIT, etc), address and phone number; optional name of user at submitter's site on whose behalf a SiteSA is reporting the problem; and submitter's estimated urgency of the problem.

Statistics below the dashed line (---) are privileged information which is maintained in the data base but not released to Site SA, users or customers. Included in this privileged information is: the User_id (at the site of submission) of the developer holding an open report; the real release in which a problem is expected to be fixed in; a management-defined priority for the problem. The real release number may differ from the announced release

number when: we are not sure a fix will actually make the real release; when the real release is not an announced release which marketing has accepted (as in the case of MR 6.5).

Following the statistics are the transactions. Transactions occur: when a report is originated (first processed); when it is answered by a developer, etc; when more information is requested from the submitter; when the submitter (or another user) provides additional information to the report.

Each transaction begins with a heading line giving the transaction number, type of transaction, source of transaction ('developer', name of TR Administrator or submitter of added info). Following the heading is the text of the transaction and statistics associated with the transaction. These statistics provide a history of the report, showing changes of status, naming TR Administrators, investigators and developers who have processed the report, etc. Statistics appearing below the dashed line (---) are privileged information, not distributed to Site SA, users or customers. Statistics from the final transaction are the source of the major statistics which follow the body.

4. NEW TR TOOLS

The following steps are involved in entering and processing data in the new trouble report data base.

- Initial entry of data when a report is submitted.
- Automatic processing of new reports on a daily basis.
- Manual handling of new reports by TR Administrator.
- Forwarding new reports to developer or submitter.
- Developer answering a report; submitter supplying more information for a report.
- Automatic processing of answers and added information.
- Manual handling of report answers by TR Administrator.
- Scanning of report data base to examine reported troubles in a given area of the system, or to gather status of outstanding reports.
- Reporting status of reports to developers and management as a mean of providing more timely response to problems, and of prioritizing our work.
- Reporting problems to sites, for use by their site maintenance personnel and user communities.

These steps are outlined in somewhat more detail in the paragraphs which follow. Several new or revised programs are proposed for entering and answer reports, for processing reports and interfacing with the new TR data base, for providing report status, and for distributing reported problems to sites.

4.1 Initial Data Entry (enter_trouble_report)

A new version of the trouble_report command will be provided to prompt for the new kinds of information which can appear in a report, and to obtain information about the submitter and his site from the TR data base so that this information need not be typed in each new report. Because we wish to separate the process of entering new reports from that of answering or adding info to previously submitted reports, the new command will be renamed to enter_trouble_report (etr).

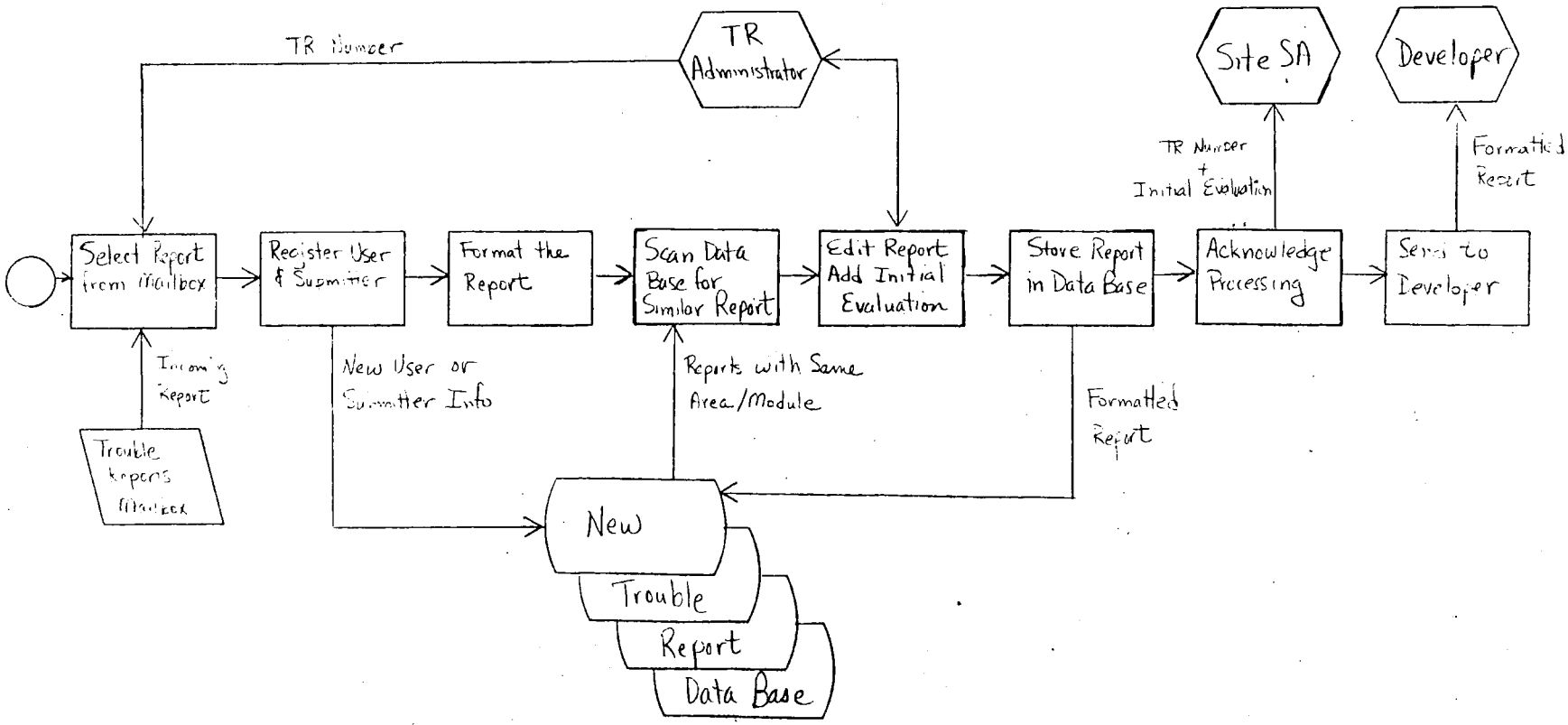
etr will:

- process a pretyped segment for those who dislike prompting. The segment will be parsed to insure that all required fields are given. Missing fields will be prompted for. Unknown field identifiers will be diagnosed.
- require about the same amount of typing as the existing trouble_report command. Some typing is eliminated because we are getting data about the submitter and his site from the TR data base; but these decreases are offset by new fields such as the "Summary" line which trouble_report does not include.
- allow the submitter to edit the report prior to submission. The favorite editor of each submitter will be included in his registration information in the TR data base. After prompting for the basic data, the submitter will be asked if he wants to edit the information. If so, his editor will be invoked on the information gathered by prompting. For pretyped segment, if any prompting occurs because of missing field or if any unknown fields are diagnosed, the submitter will be asked to edit the report.
- validate the existence and proper access to test cases named in the report. Somewhat more stringent requirements may have to be imposed on the way pathnames are given in a report.
- assign report numbers when the report is submitted. A report number obtained from a Ring 1 data base will be assigned just prior to mailing the report to the new_reports mailbox. The user will be informed of the report number for use in further correspondence regarding the report.
- maintain a TR submission log in the submitter's home directory. The log will include: the report number, date/time of submission, and the summary line. This information is sufficient to remind the submitter: of reports he hasn't received a reply from; of report numbers used in supplying more info for a report; etc. The log will be an ASCII segment which can be edited (via standard Multics editor) to remove obsolete entries. New entries will be stored at the top of the log segment, for ease of perusal.

These features are summarized by the diagram on the following page.

A New Tool for Processing Incoming Reports:

handle_new_trouble_report



4.2 Automatic Processing of New Reports

On a daily basis, an absentee job will use the AML read_mail command to copy all new reports in the new_reports mailbox into an unprocessed_new_reports mailbox. At the same time, reports will be written into an ASCII segment for backup purposes, and this segment will be dprinted for distribution to other Multics development sites.

read_mail will then be used to write all reports in the unprocessed_reports mailbox into an ASCII segment. This segment will be dprinted to allow initial perusal of new reports by TR Administrator.

4.3 Handling of New Reports (handle_new_trouble_report)

The TR Administrator will process incoming new trouble reports using a new command, handle_new_trouble_report. (2) This command will:

- invoke the AML read_mail command to select the report to be processed. Reports will be selected by giving their report number, as shown in the list of unprocessed reports previously dprinted (see 4.2 above). The new report will be written into a temporary segment in the working directory to allow further processing.
- parse the report into fields (extracting info from the mail headers, etc), and put the report in standard TR format.
- scan the TR data base for other reports in this area.
- allow the TR Administrator to edit any of the fields, or reorganize the information into different fields, etc. Particular attention will be paid to the summary line, to insure that it is a correct, complete, concise summary of the report. The area of the system and modules affected will be added to the report. Small test cases will be copied directly into the report. An ORIGINATE transaction will be added, answering the report if the TR Administrator knows the answer, or giving an initial appraisal of the report.
- parse the edited report into fields again.
- copy longer test cases and/or test data into the TR Administrator's processing hierarchy for ease of location and safekeeping.
- update any new or changed information about the submitter or his site into the submitter/site registration parts of the TR data base.
- add the new report and its originate transaction to the data base.
- forward the report to the appropriate developer (if required) to obtain an answer.
- send copy of the formatted report with initial transaction, etc to the submitter as an initial acknowledgment.

---

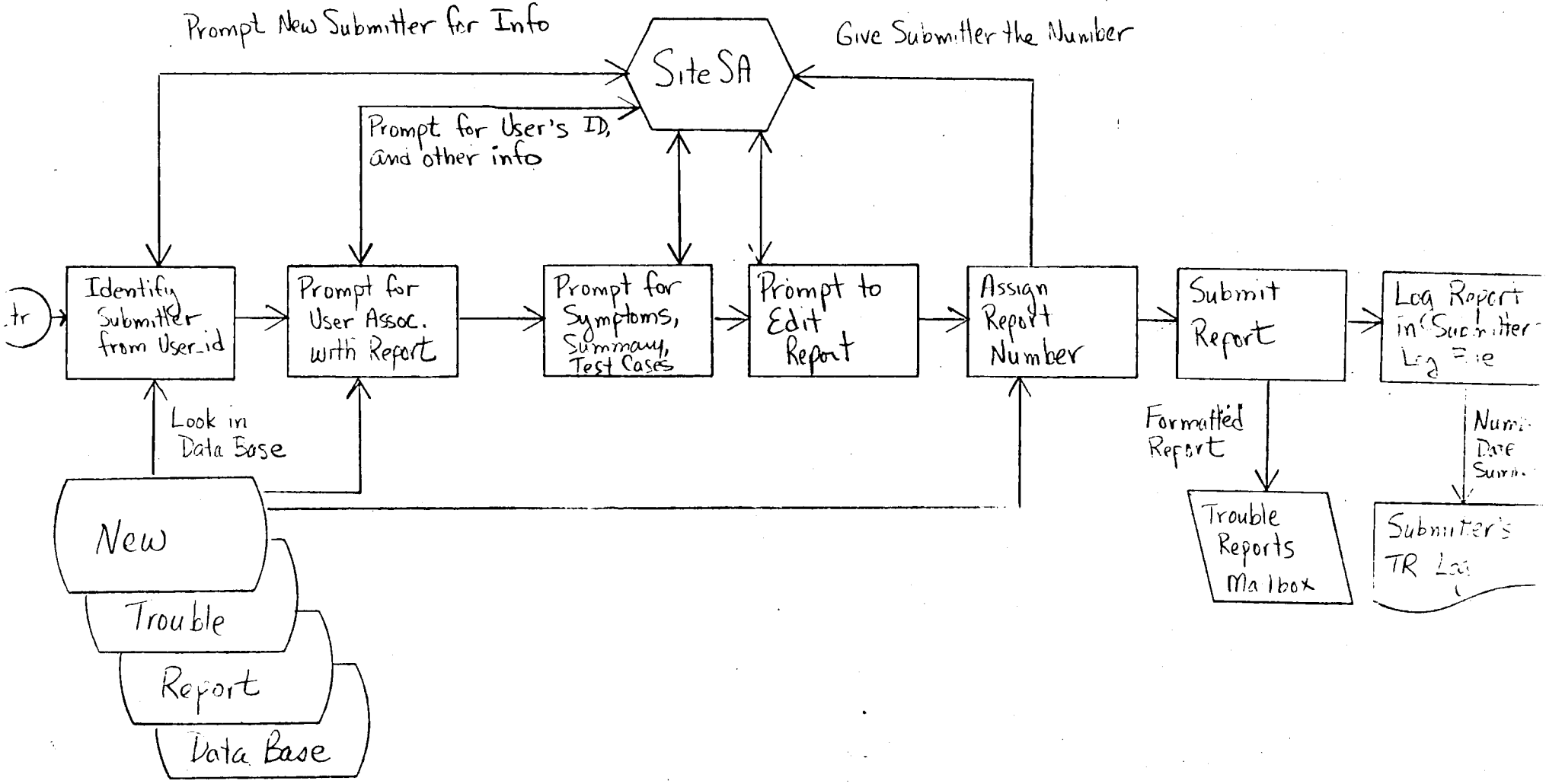
(2) Since this command is a tool, it has no short name.

The steps involved in forwarding to a developer, copying test cases, etc. will be automated, depending upon the kind of initial transaction provided by the TR Administrator. For instance, if the TR Administrator answers the report, test cases will not be copied, the report will not be forwarded to a developer, and the status will be set to closed. If the initial transaction is an ORIGINATE transaction, the report will automatically be forwarded to an investigator or developer appropriate for the area and modules affected by the report. Test cases will automatically be copied, and the submitter will be so notified.

The diagram on the following page illustrates these steps.

A New Tool for Submitting Trouble Reports:

enter_trouble_report, etr



4.4 Forwarding Reports

Reports are usually transmitted to submitters via Multics mail. This is the most convenient mechanism, especially for offsite users. However, onsite submitters who prefer to have their reports dprinted can do so by setting a switch in their registration information. In fact, a 3-way switch setting allows mailing, dprinting or both.

Some submitters keep the text of TRs they have submitted, and therefore want to reduce the size of letters sent to their mailboxes by receiving only parts of the report which they have not yet seen, rather than receiving the entire report anytime some part of it changes. The choice between these two modes will be specified by a switch in the submitter's registration information.

Onsite developers usually prefer to receive reports by Multics mail. So do most offsite developers who log in to the submission site on a regular basis. However, some offsite developers rarely log into the remote submission site. They prefer to have reports dprinted instead. Developers will have a choice of mail/dprint in their registration.

Similarly, some developers prefer to receive only new parts of a report, rather than the entire report when it has passed several times from submitter to developer. An all/new choice is also available in their registration info.

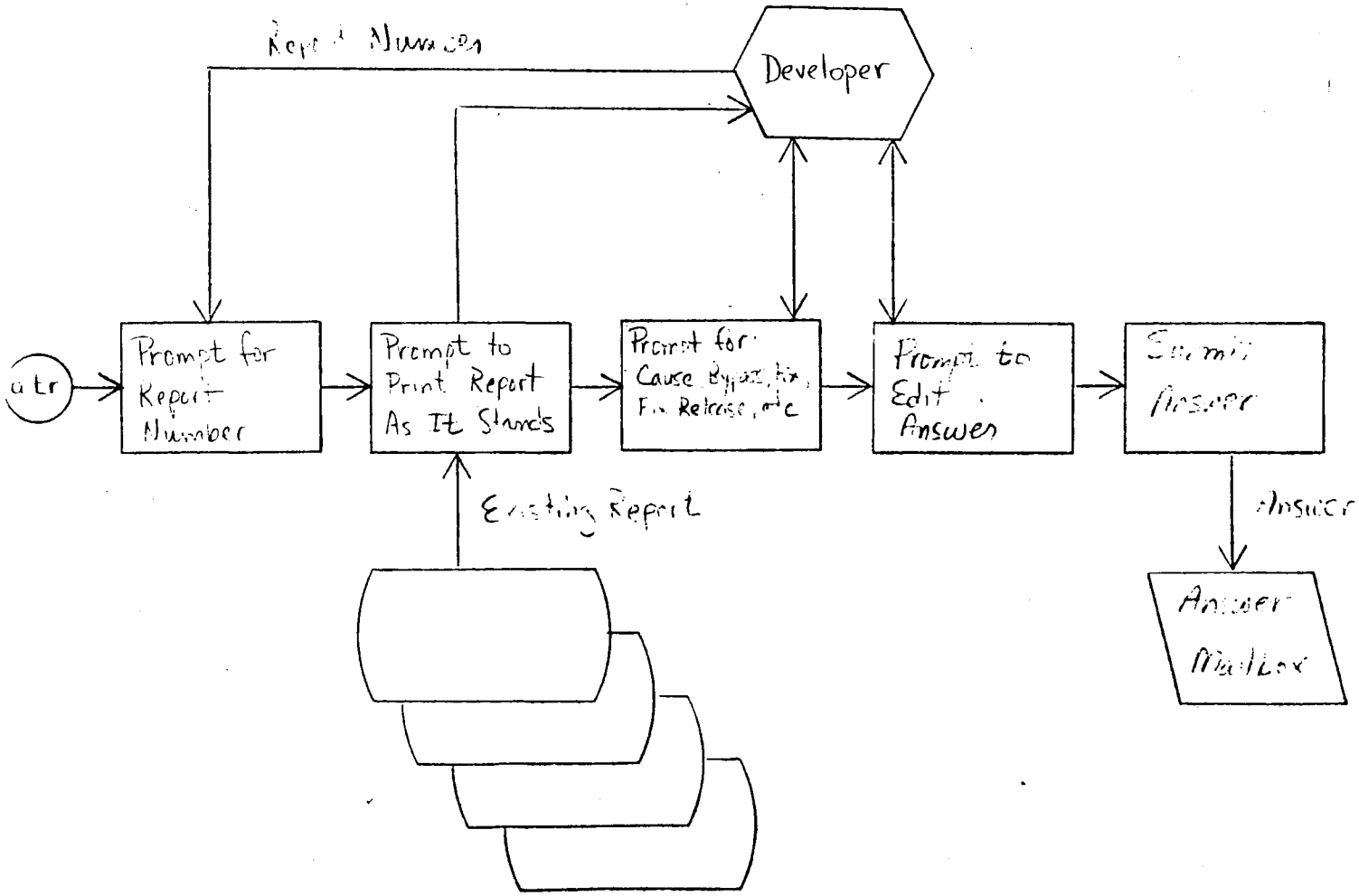
4.5 Answering or Adding to Reports (answer_/add_to_trouble_report)

A new answer_trouble_report (atr) command will prompt the developer for a report answer and for new status and priority values for the report. The report number must be known to answer a report. Report answers will be mailed to a special report_answers mailbox to which users and Site SAs do not have access. This will protect the anonymity of the developer.

An add_to_trouble_report (attr) command will prompt submitters and other users for additional information regarding a report. The report number must be known to add info to a report. The submitter can obtain the report number from his personal TR log segment. Other users wishing to comment on a report will have seen the report, and can obtain the number from the report itself. Additional information will be mailed to a special report_info mailbox to which everyone has read access.

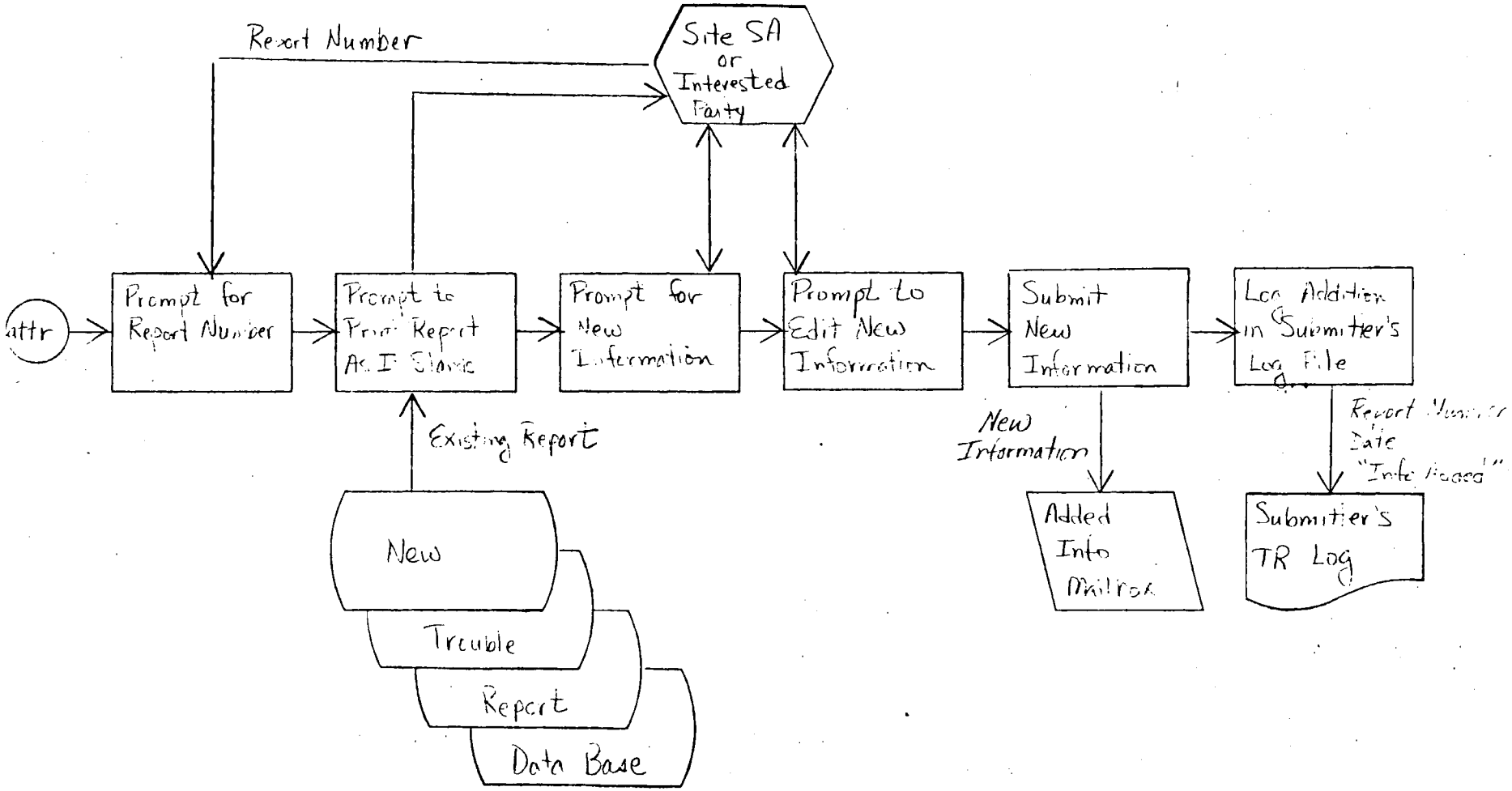
The use of these commands is illustrated in the diagrams on the following pages.

A New Tool For Answering Reports:
answer_trouble_report, atr



New Tool for Adding Information to Reports:

add_to_trouble_report, attr



4.6 Automatic Processing of Answers and Added Information

On a daily basis, an absentee job will use the AML read_mail command to copy answers and added information from the report_answers and report_info mailboxes into an unprocessed_answers mailbox to which only TR Administrators have access.. Also, all letters will be written into an ASCII segment for backup purposes.

read_mail will then be used to write all unprocessed_answers into an ASCII segment. The segment will be printed to allow initial perusal of answers and added information by the TR Administrator.

4.7 Handling Report Answers (handle_old_trouble_report)

Trouble report personnel will process incoming answers and added information using a new command, handle_old_trouble_report. This command will:

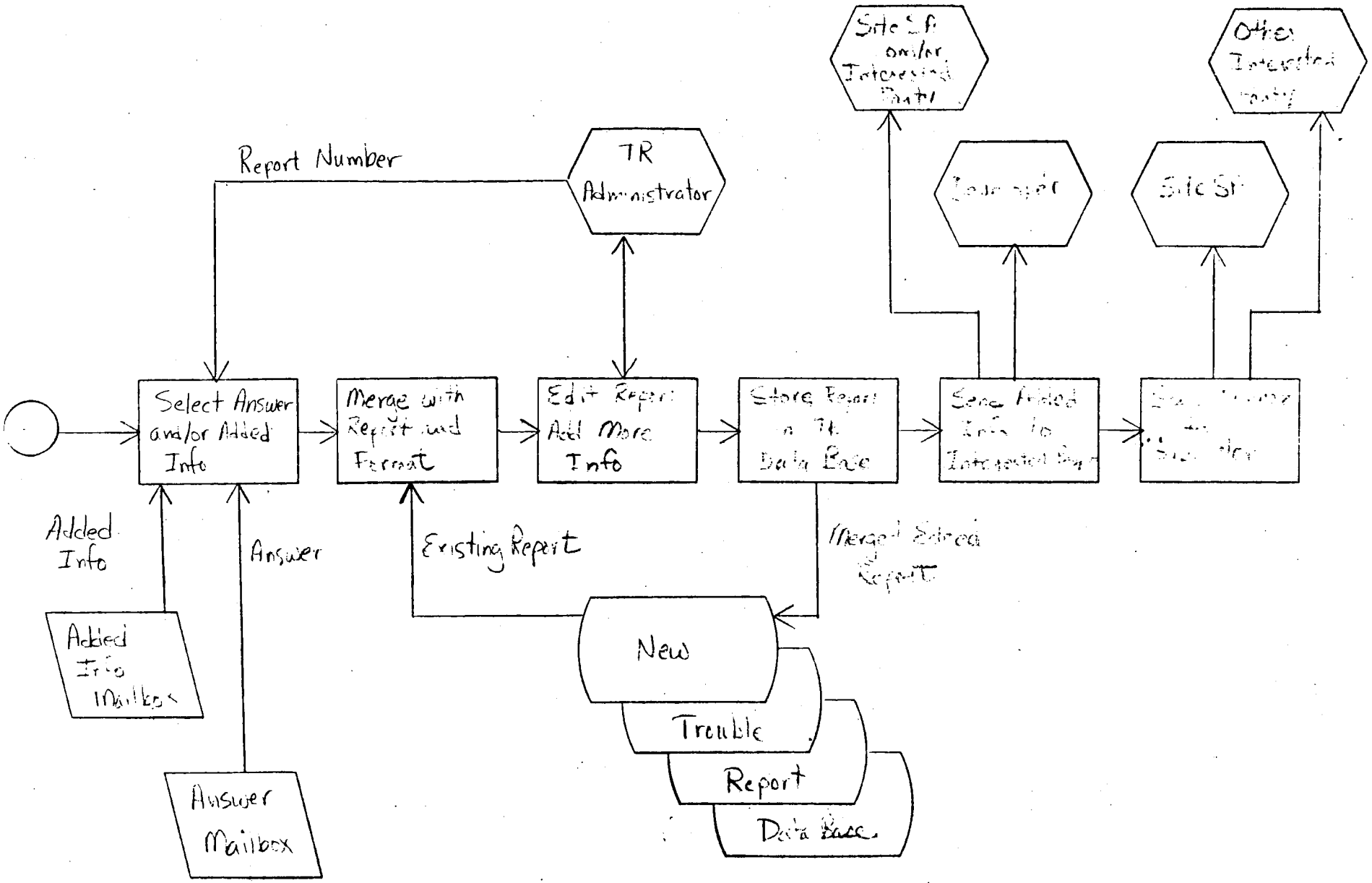
- invoke read_mail to select the answers to be processed. Answers will be selected by giving their report number. Each selected answer is written into a temporary segment in the working directory. If more than one piece of information is available for a given report, the various pieces are written into separate segments.
- parse answer(s) into transaction fields, putting data into proper format. If more than one piece of information is available for a given report, the transactions are sorted into chronological order by date of arrival in the report_answers or report_info mailbox.
- merge the transactions with previous report data.
- allow the TR Administrator to edit any of the fields in any transaction. He must verify that any status, fix release or priority given in an answer is valid.
- parse the edited report into fields again.
- update the report in the data base.
- forward the answered report to the submitter (and to any other users who may have commented on the report). Forward added info to the developer, and to any other user who may have commented on the report.
- if the report is closed, delete any test cases which were previously copied.

The majority of these operations will occur automatically, with little or no intervention required by the TR Administrator.

The diagram on the following page illustrates these operations.

A New Tool for Processing Added Info and Answers:

handle_old_trouble_report



4.8 Scanning the Trouble Report Data Base (scan_trouble_report)

A new command will be needed to scan the data in the trouble report data base when looking for selected information. Initially, the linus command and a set of linus macros may have to be used for this purpose. However, when time permits, a more tailored interface for performing this frequent task would be desirable.

A scan_trouble_report (str) command will scan for reports on the following fields: submitter, site, release, area/module, date_orig, date_mod, type, status, developer (for users with access to privileged data). Scanning and sorting will be allowed on any or all of the above categories. Eventually, scanning should be allowed in almost any of the report fields.

4.9 Reporting Status of Trouble Reports

Various managements reports on the status of outstanding reports must be produced so that we can respond to problems in a more timely way, and better prioritize our work to more directly meet user needs. Hopefully, MRPG can be used to produce most of these reports, but some special reporting programs may have to be written.

Some typical reports current considered include:

- a weekly report to each developer listing all open reports which he is holding.
- a monthly report to project leaders listing all suggestions for changes to software in their area.
- a monthly report to management listing all open reports, sorted by priority; another listing sorted by date_originated.

I would welcome suggestions for additional reports which might be helpful.

4.10 Reporting Problems to Sites (display_trouble_reports)

Customers have asked that we distribute a list of known problems with each new Multics release. The information required for this list will be available in the Trouble Report Data Base. However, we cannot distribute the data base itself. Some sites do not purchase MRDS, and so could not use the data base. Also, the data base contains privileged information which should not be distributed to sites.

To bypass these problems, selected data from the data base will be formatted as a series of trouble segments. One segment will be created for each area of the system. Each segment will begin with a summary of the problems known in that area. The summary lines will include: report number; release in which problem was encountered; summary field from the report. Following the summary will be the full text (excluding privileged information) of the reports in that area.

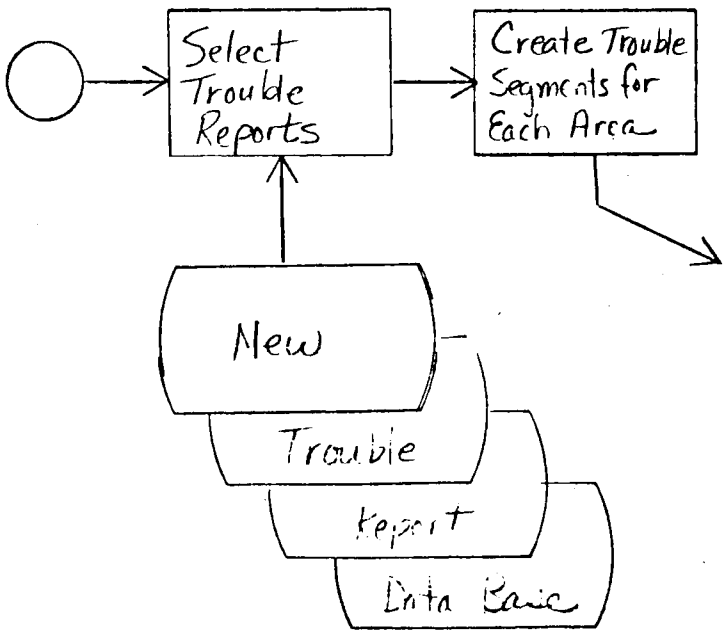
The trouble segments will be formatted as info segments, with multiple infos per segment. A new display_trouble_report (dtr) command will interface with the new help_ subroutine to find and display the desired trouble report. display_trouble_report will be a slightly modified version of the help command.

A create_trouble_segments tool will be required to select and format trouble reports as trouble segments.

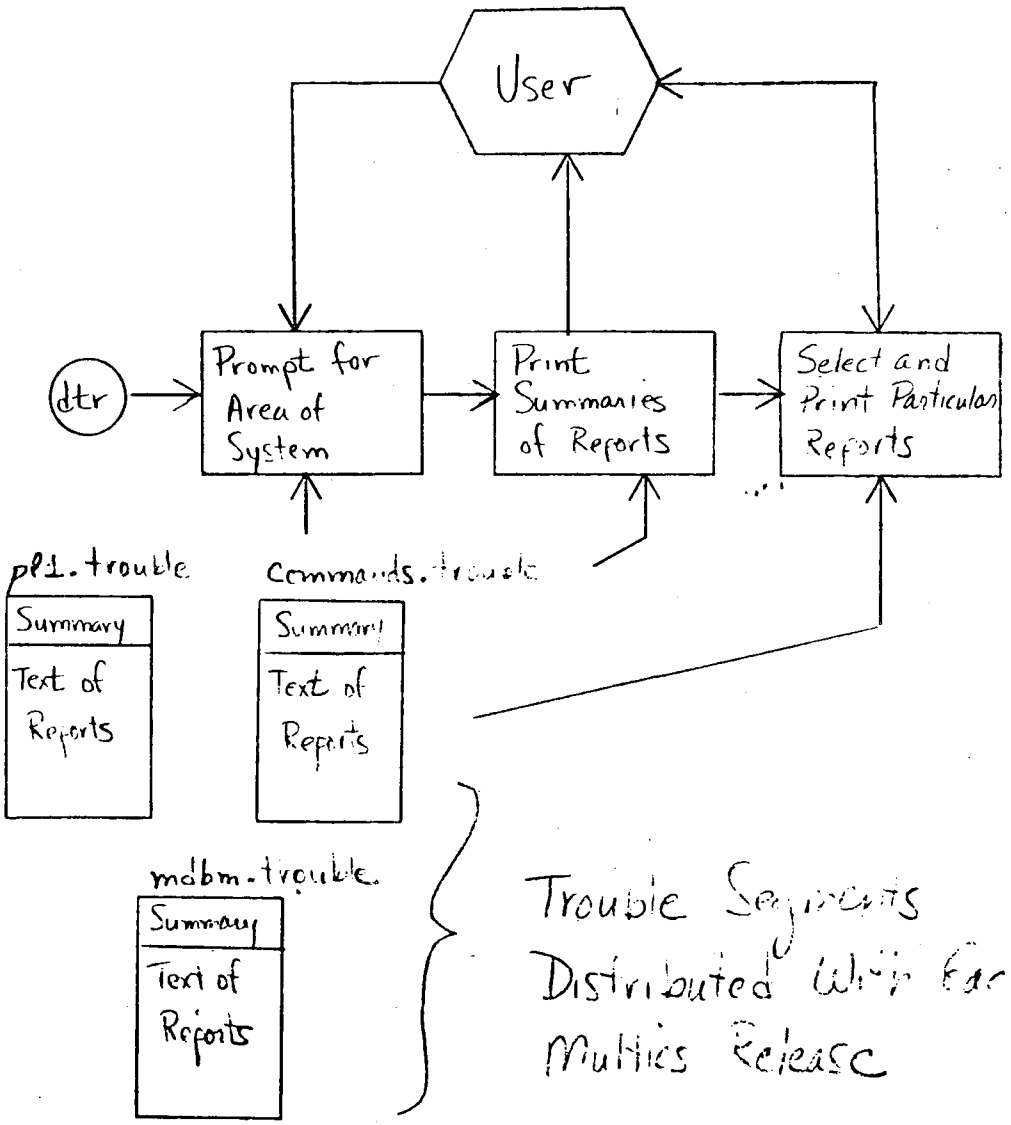
The following diagram illustrates the trouble report display process. The final diagram gives an overview of the new Trouble Reporting System in its entirety.

New Tools to Create and Display Trouble Segments:

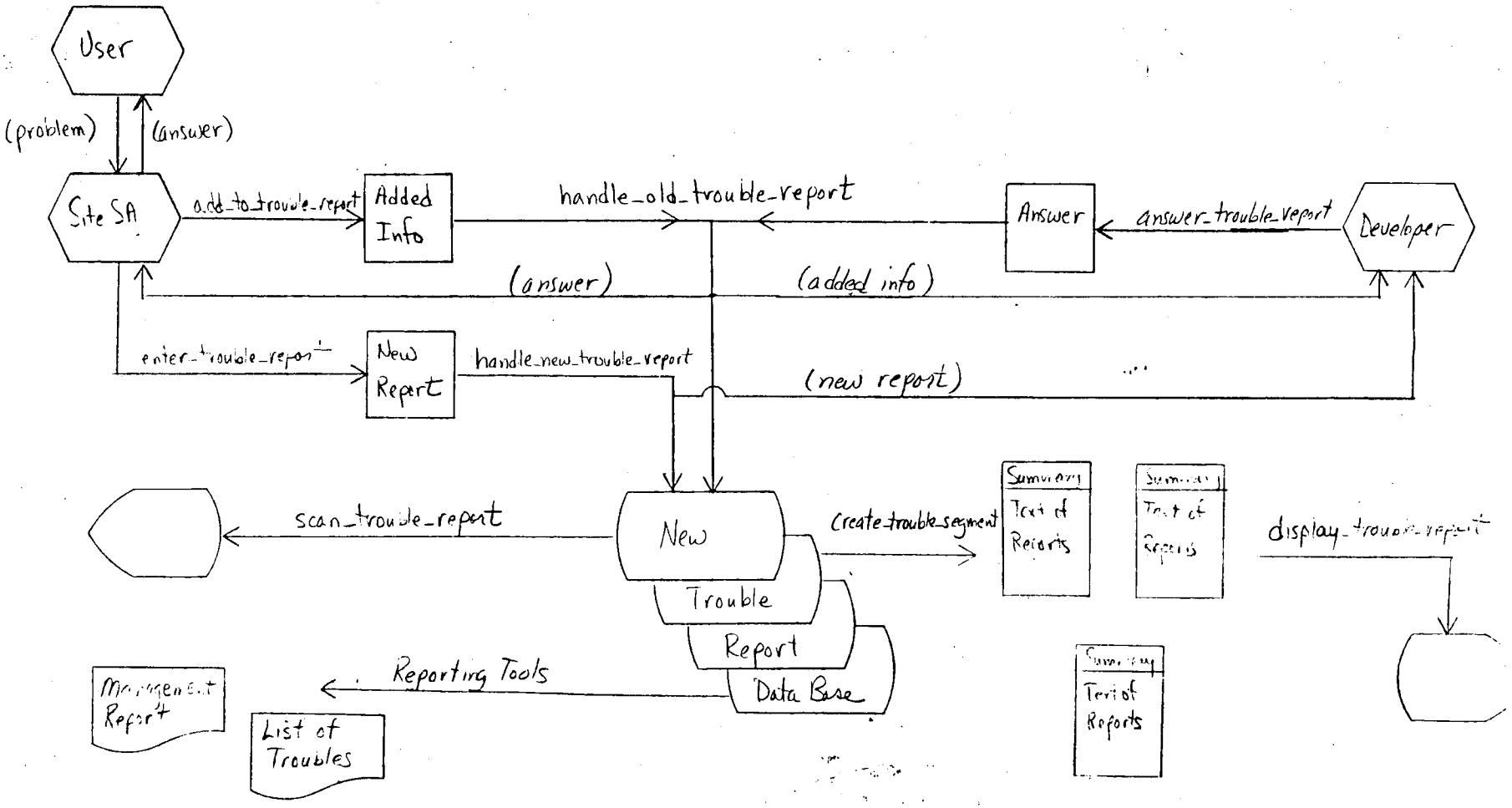
create_trouble_segments



display_trouble_report, dtr



Overview of the New Trouble Reporting System



APPENDIX A

TR_log.cmdb

The following create_mrds_db source file defines the proposed Trouble Reporting data base.

```
domain:
   address  char(100) var,       /* Address of Site SA or User             */
   all_new  char(3),             /* Send 'all' of TR to user/dev/site_sa   */
                                 /* or just 'new' parts.                   */
   area     char(32),            /* Multics Dev. Project area              */
   date     char(35),            /* year-mm-dd_HH:MM:SS.UUUUUU_z_da        */
   editor   char(100) var,       /* way to invoke editor preferred by      */
                                 /* user or Site SA or developer.          */
   holder   char(12),            /* 'investigator', 'developer' or 'user'  */
   dev_inv_no fixed bin,         /* number of developers assoc with an     */
                                 /* area/module combination if positive.   */
                                 /* number of investigators assoc. with    */
                                 /* area/module combination if negative.   */
   mail_dp  char(6),             /* 'mail' TR to dev, 'dprint' or 'both'   */
   materials char(100) var,      /* list of supplementary materials        */
   misc_no  fixed bin,           /* number of misc. items assoc. w/ TR.    */
   mod_no   fixed bin,           /* number of module items assoc. w/ TR.   */
   module   char(32),            /* module w/in area assoc. w/ TR          */
   name     char(40) var,        /* Name of user/site_sa/developer         */
   no       char(8),             /* TR Number - phx99999                   */
   path     char(168) var,       /* path name of test case file            */
   path_no  fixed bin,           /* number of path names assoc. w/ TR      */
   pers     char(22),            /* Person_id part of Multics User_id      */
   phone    char(32) var,        /* Phone no of user/site_sa/developer     */
   priority char(4),             /* 00.0 to 99.9                           */
   prog     char(4800) var,      /* text of test case program              */
   proj     char(9),             /* Project_id part of Multics User_id     */
   release  char(8),             /* Multics Release assoc. w/ TR           */
   site     char(32),            /* Name of Site originating TR.           */
   status   char(16),            /* 'opened', 'closed'                     */
   summary  char(256) var,       /* TR Summary - VERY BRIEF                */
   symptom  char(9600) var,      /* TR Full Description                    */
   text     char(4800) var,      /* text of cause, bypass, fix, or         */
                                 /* transaction tuple                      */
   tran_no  fixed bin,           /* No. Transactions assoc. w/ this TR     */
   type     char(16),            /* type of report -- 'suggestion',        */
                                 /* 'trouble', 'query', 'reject'           */
                                 /* type of transaction - 'originate',     */
                                 /* 'answer', 'more_info', 'info',         */
                                 /* 'investigate'                          */
   urgency  char(16) var;        /* Urgency?                               */

attribute:
```

```
   date_handled      date,     /* date transaction handled              */
   date_mod          date,     /* date TR last changed                  */
   date_orig         date,     /* date TR submitted                     */
   date_rec          date,     /* date transaction received             */
   dev_pers          pers,     /* Person_id of developer                */
   dev_proj          proj,     /* Project_id of developer               */
   hold_pers         pers,     /* Person_id of person holding TR        */
   hold_proj         proj,     /* Project_id of person holding TR       */
   inv_pers          pers,     /* Person_id of investigator             */
   inv_proj          proj,     /* Project_id of investigator            */
   mr_fix            release,  /* Multics release TR to be fixed in     */
                               /* as announced to general public        */
   primary_sa        pers,     /* SA primarily responsible for site     */
   real_mr_fix       release,  /* Multics release TR really fixed in    */
   rel_no            no,       /* related TR number.                    */
   sa                pers;     /* Person_id of Site SA assoc. w/ TR     */

relation:
   report (no* date_orig type site release sa pers proj
           summary symptom urgency holder status mr_fix date_mod
           tran_no path_no misc_no mod_no),
                               /* basic relation of data base.  One     */
                               /* tuple per TR.  Contains all info      */
                               /* about TR which is fixed-length &      */
                               /* which can be told to gen'l public.    */

   rpt_priv (no* hold_pers hold_proj real_mr_fix priority),
                               /* One tuple per TR.  Contains info      */
                               /* about TR private from gen'l public.   */

   module (no* mod_no* area module),
                               /* Many tuples per TR.  Contains names   */
                               /* of area/modules assoc w/ TR.          */

   cause (no* date text),      /* One tuple per TR.  Gives cause of     */
                               /* TR if not stated in symptom.          */

   bypass (no* date text),     /* One tuple per TR.  Gives bypass for   */
                               /* problem stated in TR.                 */

   fix (no* date text),        /* One tuple per TR.  Gives brief        */
                               /* description of problem fix.           */

   related (no* rel_no*),      /* Many tuples per TR.  Gives TR no.     */
                               /* of TRs related to this one.           */

   test_case (no* prog),       /* One tuple per TR.  Gives short test   */
                               /* case which causes the problem.        */

   test_path (no* path_no* path),
                               /* Many tuples per TR.  Gives path name  */
                               /* of longer test programs, test data    */
```

```
  support (no* materials),      /* One tuple per TR.  Indicates that    */
                                /* listings, console log, etc assoc.    */
                                /* w/ TR                                */

  misc (no* misc_no* date text),
                                /* Many tuples per TR.  Gives misc.     */
                                /* info about the TR.                   */

  transaction (no* tran_no* date_rec date_handled type
               pers proj text urgency status holder),
                                /* Many tuples per TR.  Give public     */

  tran_priv (no* tran_no* hold_pers hold_proj real_mr_fix priority),
                                /* One tuple for every tuple in         */
                                /* transaction.  Private info not to    */
                                /* be given to general public.          */

  site (site* release primary_sa),
                                /* One tuple per site.                  */

  sa (pers* proj site* name phone address editor all_new mail_dp),
                                /* One tuple pr Site SA/Site combo.     */

  user (pers* proj site* name phone address editor all_new mail_dp),
                                /* One tuple per User_id at given site  */

  system (module* area),        /* area of system which includes each   */
                                /* system module.                       */

  responsible (module* dev_pers dev_proj inv_pers inv_proj),
                                /* Many tuples per area/module combo,   */
                                /* defining developer(s) responsible    */
                                /* for that area/module.                */

  dev (pers* proj* site* name phone address editor all_new mail_dp);
                                /* One tuple per developer.  Gives      */
                                /* developer's local Project_id &       */
                                /* attributes                           */

index:                          /* Secondary Indices                    */
  report (type status site release mr_fix sa pers holder date_orig date_mod)
  rpt_priv (hold_pers real_mr_fix priority),
  module (area module),
  related (rel_no),
  transaction (date_rec date_handled type pers proj holder status),
  sa (site),
  user (site),
  dev (site proj),
  responsible (dev_pers inv_pers),
  system (area);
```

APPENDIX B

TR_log.incl.pl1

The following include file defines the tuples (records) in each relation (file) of the proposed Trouble Reporting data base.

```
/* ************************************************
   *                                              *
   * BEGIN tr_log.incl.pl1                        *
   *       created: 09/19/78  1542.0 mst Tue      *
   *            by: create_mrds_dm_include (1.0)  *
   *                                              *
   * Data model >udd>m>gd>trouble_reports>tr_log   *
   *       created: 09/19/78  1536.8 mst Tue      *
   *       version: 3                             *
   *            by: GDixon.SysMaint.a             *
   *                                              *
   ************************************************ */

dcl 1 report aligned,
      2 no character (8) unaligned,                    /* Key */
      2 date_orig character (35) unaligned,            /* Index */
      2 type character (16) unaligned,                 /* Index */
      2 site character (32) unaligned,                 /* Index */
      2 release character (8) unaligned,               /* Index */
      2 sa character (22) unaligned,                   /* Index */
      2 pers character (22) unaligned,                 /* Index */
      2 proj character (9) unaligned,
      2 summary character (256) varying aligned,
      2 symptom character (9600) varying aligned,
      2 urgency character (16) varying aligned,
      2 holder character (12) unaligned,               /* Index */
      2 status character (16) unaligned,               /* Index */
      2 mr_fix character (8) unaligned,                /* Index */
      2 date_mod character (35) unaligned,             /* Index */
      2 tran_no real fixed binary (17,0) aligned,
      2 path_no real fixed binary (17,0) aligned,
      2 misc_no real fixed binary (17,0) aligned,
      2 mod_no real fixed binary (17,0) aligned;

dcl 1 rpt_priv aligned,
      2 no character (8) unaligned,                    /* Key */
      2 hold_pers character (22) unaligned,            /* Index */
      2 hold_proj character (9) unaligned,
      2 real_mr_fix character (8) unaligned,           /* Index */
      2 priority character (4) unaligned;              /* Index */

dcl 1 bypass aligned,
      2 no character (8) unaligned,                    /* Key */
```

```
      2 date character (35) unaligned,
      2 text character (4800) varying aligned;

dcl 1 cause aligned,
      2 no character (8) unaligned,                    /* Key */
      2 date character (35) unaligned,
      2 text character (4800) varying aligned;

dcl 1 fix aligned,
      2 no character (8) unaligned,                    /* Key */
      2 date character (35) unaligned,
      2 text character (4800) varying aligned;

dcl 1 misc aligned,
      2 no character (8) unaligned,                    /* Key */
      2 misc_no real fixed binary (17,0) aligned,      /* Key */
      2 date character (35) unaligned,
      2 text character (4800) varying aligned;

dcl 1 module aligned,
      2 no character (8) unaligned,                    /* Key */
      2 mod_no real fixed binary (17,0) aligned,       /* Key */
      2 area character (32) unaligned,                 /* Index */
      2 module character (32) unaligned;               /* Index */

dcl 1 related aligned,
      2 no character (8) unaligned,                    /* Key */
      2 rel_no character (8) unaligned;                /* Key, Index */

dcl 1 support aligned,
      2 no character (8) unaligned,                    /* Key */
      2 materials character (100) varying aligned;

dcl 1 test_case aligned,
      2 no character (8) unaligned,                    /* Key */
      2 prog character (4800) varying aligned;

dcl 1 test_path aligned,
      2 no character (8) unaligned,                    /* Key */
      2 path_no real fixed binary (17,0) aligned,      /* Key */
      2 path character (168) varying aligned;

dcl 1 transaction aligned,
      2 no character (8) unaligned,                    /* Key */
      2 tran_no real fixed binary (17,0) aligned,      /* Key */
      2 date_rec character (35) unaligned,             /* Index */
      2 date_handled character (35) unaligned,         /* Index */
      2 type character (16) unaligned,                 /* Index */
      2 pers character (22) unaligned,                 /* Index */
      2 proj character (9) unaligned,                  /* Index */
      2 text character (4800) varying aligned,
      2 urgency character (16) varying aligned,
```

```
      2 status character (16) unaligned,                    /* Index */
      2 holder character (12) unaligned;                    /* Index */

dcl 1 tran_priv aligned,
      2 no character (8) unaligned,                         /* Key */
      2 tran_no real fixed binary (17,0) aligned,           /* Key */
      2 hold_pers character (22) unaligned,
      2 hold_proj character (9) unaligned,
      2 real_mr_fix character (8) unaligned,
      2 priority character (4) unaligned;

dcl 1 system aligned,
      2 module character (32) unaligned,                    /* Key */
      2 area character (32) unaligned;                      /* Index */

dcl 1 responsible aligned,
      2 module character (32) unaligned,                    /* Key */
      2 dev_pers character (22) unaligned,                  /* Index */
      2 dev_proj character (9) unaligned,
      2 inv_pers character (22) unaligned,                  /* Index */
      2 inv_proj character (9) unaligned;

dcl 1 dev aligned,
      2 pers character (22) unaligned,                      /* Key */
      2 proj character (9) unaligned,                       /* Key, Index */
      2 site character (32) unaligned,                      /* Key, Index */
      2 name character (40) varying aligned,
      2 phone character (32) varying aligned,
      2 address character (100) varying aligned,
      2 editor character (100) varying aligned,
      2 all_new character (3) unaligned,
      2 mail_dp character (6) unaligned;

dcl 1 site aligned,
      2 site character (32) unaligned,                      /* Key */
      2 release character (8) unaligned,
      2 primary_sa character (22) unaligned;

dcl 1 sa aligned,
      2 pers character (22) unaligned,                      /* Key */
      2 proj character (9) unaligned,
      2 site character (32) unaligned,                      /* Key, Index */
      2 name character (40) varying aligned,
      2 phone character (32) varying aligned,
      2 address character (100) varying aligned,
      2 editor character (100) varying aligned,
      2 all_new character (3) unaligned,
      2 mail_dp character (6) unaligned;

dcl 1 user aligned,
      2 pers character (22) unaligned,                      /* Key */
      2 proj character (9) unaligned,
```

```
      2 site character (32) unaligned,                    /* Key, Index
      2 name character (40) varying aligned,
      2 phone character (32) varying aligned,
      2 address character (100) varying aligned,
      2 editor character (100) varying aligned,
      2 all_new character (3) unaligned,
      2 mail_dp character (6) unaligned;

/* END of tr_log.incl.pl1    ************************************/
```