RECEIVED
7 NOV 1995
REGISTRY OF
TRADE MARKS & PATENTS
SINGAPORE

AUTOMATIC PROGRAM GENERATOR

BACKGROUND OF THE INVENTION

FIELD OF THE INVENTION

The present invention relates to an automatic program generating system which can generate all application programs at a constant quality level and a standardized design document in perfect correspondence to the generated program.

DESCRIPTION OF THE RELATED ART

Effective utilization of a computer requires a suitable application program tailored to the user's service but this application program has heretofore been prepared chiefly in dependence on the personality of the program designer.

When such an application program has been perfected, the user is generally provided with the object codes and operation manual for the program and operates the application program according to the operation manual. If and whenever the application program is found not operating normally, a call for maintenance is made in order that the program may keep operating normally.

On perfection of the program, the user is at times supplied with a source code list and a document showing the contents of design but this document has to be

manually prepared and since much time must be expended for its perfection, the user is not necessarily provided with a complete document.

Meanwhile, even if it is a normally operating application program, the user may often wish to modify the contents of its processing to keep abreast of the progress of the times and changes in his service but the conventional hardware has the drawback that an application program once perfected cannot be easily modified.

Thus, in the event that the source code list of the program is absent or the source code list has lost correspondence with the object codes due to, for example, maintenance of the program, even a minor modification of the processing contents of the program requires an enormous length of time, with the result that a new application program has to be reconstructed from the beginning as it has often been the case in the past.

Moreover, even when a complete source code list is available, each program has a personality originating from the designer's ability and skill so that a third person cannot easily grasp the processing contents. Therefore, it takes much time for him to modify the contents and in some cases the whole program ceases to function properly owing to such modification.

Furthermore, should the various documents be available from the archives, oftentimes the actual processing contents are not exactly reflected in the documents and even if the contents have been accurately described, all are stated using different formats so that it is quite difficult for a third person to grasp the contents of the program.

Since, it is essential to modify application programs with the progress of the times, the program supplied to the user should provide for the ease of revision and addition at later times. To be specific, application programs should be of uniform quality, not dependent on the designer's personality and the processing contents of any program supplied to the user must be such that anyone can accurately understand them. Having been developed according to the above rationale, this invention has for its object to provide an automatic program generator which can automatically generate application programs of constant quality which are not dependent on the personality of the designer and outputs standardized design documents setting forth the contents of design.

SUMMARY OF THE INVENTION

To accomplish the above object, the automatic program generator of this invention comprises (1) a

skeleton memory means storing skeletons which are semi-finished source programs, (2) a design document memory means storing semi-finished design documents describing the processing contents of the respective skeletons, (3) an individual information input means which functions at the design stage of each application program to receive individual information varying with different application programs by dialog, (4) a program perfecting means which, based on the individual information received, edits said skeletons and generate source codes for respective application programs, (5) a design document perfecting means which, based on the individual information received, edits said design document in semi-finished state and perfects a design document corresponding to each application program, and (6) an output means which acts in response to appropriate manipulation to output the source code list and/or design document of the finished application program.

The skeleton memory means stores semi-finished source programs (skeletons) so that various kinds of application programs may be implemented. Those skeletons are typically classified into first group programs which register data in a transaction file, second group programs which search the data of a data file and displays it on a terminal screen, third group programs

which register data in a master file, fourth group programs which search the data of a data file and displays it in the window of a terminal screen, fifth group programs for printing out the contents of the data file on a business form, and sixth group programs which update the contents of a master file and/or a temporary file according to the data of a transaction file. In the present invention, all application programs are generated by using these first group through sixth group programs in suitable combinations.

In the present invention, a design document corresponding to each skeleton is memorized in semi-finished state. And based on the individual information entered by dialog, the design document in semi-finished state is edited to provide a design document completely corresponding to each application program. Since the skeletons are classified into six major groups, the design document format and the items of entry can be easily standardized. Moreover, because the design documents have been standardized, it is also easy for a third person to understand the contents of each design.

BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWING

Fig. 1 is a block diagram showing the automatic

program generator as an embodiment of this invention;

Fig. 2 is a schematic diagram showing the construction of a skeleton;

Fig. 3 is a diagram showing differences among design pattern 1 through design pattern 7;

Fig. 4 is a view showing a typical purchase voucher;

Fig. 5 is a view showing a typical transaction file in which the data of the purchase voucher illustrated in Fig. 4 are registered;

Fig. 6 is a flow chart showing the contents of processing of design patterns 1 and 2;

Fig. 7 is a view showing a typical window display for implementing a change in action mode;

Fig. 8 is a view showing a typical transaction file in which the data of the purchase voucher illustrated in Fig. 4 are registered;

Fig. 9 is a flow chart of showing the contents of processing of design patterns 3, 4 and 5;

Fig. 10 is a diagram showing an example of edit type data;

Fig. 11 is a view showing a typical transaction file in which edit-type data are registered;

Fig. 12 is a flow chart showing the contents of processing of design patterns 6 and 7;

Fig. 13 is a flow chart showing the contents of processing of skeleton SK010;

Fig. 14 is a flow chart showing the contents of processing of skeleton SK010;

Fig. 15 is a view specifically showing some of the contents of processing of skeleton SK010;

Fig. 16 is a flow chart showing the conents of processing of design pattern 11;

Fig. 17 is a flow chart showing the contents of processing of design patterns 12, 13 and 14;

Fig. 18 is a view showing a typical master file;

Fig. 19 is a flow chart showing the contents of processing of design patterns 15-29;

Fig. 20 is a view for explaining a copying function;

Fig. 21 is a flow chart showing the contents of processing of skeleton WN010;

Fig. 22 is a view showing a window display;

Fig. 23 is a diagram showing the contents of processing of skeleton LT020;

Fig. 24 is a diagram showing the contents of processing of skeleton LT021;

Fig. 25 is a view showing a hard copy of the contents of a transaction file as obtained with skeleton LT021;

Fig. 26 is a diagram showing the contents of processing of skeleton LT010;

Fig. 27 is a diagram showing the contents of processing of skeleton LT030;

Fig. 28 is a diagram showing the contents of processing of skeleton LT040;

Fig. 29 is a diagram showing the contents of processing of skeleton BU020 (design pattern 35);

Fig. 30 is a diagram showing the contents of processing of skeleton BU010;

Fig. 31 is a diagram showing the design procedure for a physical file;

Fig. 32 is a view showing a physical file (data base) design document;

Fig. 33 is a view showing the source code list relevant to the physical file;

Fig. 34 is a view showing the design procedure for a screen file;

Fig. 35 is a view showing a portion of the screen file design document;

Fig. 36 is a view showing another portion of the screen file design document;

Fig. 37 is a view showing the source code list relevant to the physical file;

Fig. 38 is a diagram showing the procedure for

rough design of a program (unit information registration);

Fig. 39 is a diagram showing the procedure for detailed design of a program;

Fig. 40 is a view showing a part of skeleton MM011;

Fig. 41 is a view showing a part of the source code list for an automatically generated application program;

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Fig. 1 shows an automatic program generator as one embodiment of this invention, which generator essentially comprises a system body 1 and work stations WS1-WSn. While the automatic program generator of Fig. 1 can be constructed with the IBM AS/400 System, a similar generator can also be constructed using any other brand computer.

The system body 1 comprises a central processor 2, a disk memory means 3 and a main memory means 4, and optionally a printer 5 as operatively connected to said body 1. The work station WS is an input/output terminal device equipped with a CRT display 6 and a keyboard 7, among others, and where necessary, a printer 8 is connected to the terminal.

In the automatic program generator shown in Fig.

1, the source and object codes of an application program are automatically generated simply as the user goes through the design procedures of physical file design, screen file design, business form file design and program design by dialog manipulation at the work station WS. Moreover, the source code list faithfully corresponding to the generated object codes and various design documents faithfully corresponding to said object codes (data base design document, screen file design document, program design document) are also outputted as hard copies. Incidentally, the program language that can be used is not particularly restricted. Thus, various languages such as the RPG language, C language, COBOL language, etc. can be employed. In the following description of the embodiment, however, generation of an application program in the RPG (Report Program Generator) language is explained.

With the automatic program generator illustrated in Fig. 1, for implementation of the above processings, all computer processings are categorized into 36 different solutions and the programs for implementing these solutions are stored as 34 different skeleton programs in the disk memory means 3. Each of the 34 different skeletons is a source program in semi-finished state and consists of an individual portion B

which is changed for each processing and a body portion A which is not changed (Fig. 2). The individual portion B is determined according to the information entered from the keyboard 7 in the design stage of physical file design, picture file design, book file design or program design.

Since each skeleton is a high-quality program prepared by a skilled programmer and the application program design is implemented by dialog at the work station WS, the automatic program generator of this invention enables even a novice to easily and rapidly complete a high-quality application program.

Furthermore, since all programs are constructed by various combinations of 34 different skeletons, the format of various design documents can be standardized and, in addition, various design documents can be memorized in semi-finished state in the disk memory means 3. For these reasons, once an application program is completed, the various design documents faithfully corresponding to the program can be printed out. And, since these documents have been standardized in format, anyone can easily and accurately grasp the contents of processing of the application program. In the past, such various design documents were not supplied to the user or, if supplied, they were not

perfectly corresponding to the program. Since, with the program generator of this invention, design documents faithfully corresponding to the application program are provided, anyone can easily revise or modify the application program so that the asset value of the application program supplied to the user is further increased.

Thus, in the device of this invention, the desired application program can be constructed by any of the 36 different solutions or a combination thereof. The 36 different solutions are roughly classified into the categories of 1. entry system processing (design patterns 1-7), 2. inquiry system processing (design patterns 8-14), 3. master maintenance system processing (design patterns 15-22), 4. window enquiry system processing (design patterns 23-26), 5. voucher printing system processing (design patterns 27-34), and 6. batch updating system processing (design patterns 35-36).

1. Entry system processing (design patterns 1-7)

The term 'entry system processing' means a processing relating to a transaction file which accumulates data for updating a master file. According to the characteristics of data handled, this entry system processing is classified into 7 kinds of processings (design patterns 1-7) and these processings of design

patterns 1-7 are implemented by skeletons ET010, ET011, ET050, ET051, ET052, ET150 and ET153, respectively (Fig. 3).

As shown in Fig. 3, the entry system processing is divided into design patterns 1-5 and design patterns 6-7. The design patterns 1-5 are solutions suitable for handling voucher-type data, while design patterns 6-7 are solutions suitable for handling edit-type data.

The term 'voucher-type data' means a group of data designated by voucher number, and such that the data in the specification part are assigned with line numbers and registered in the transaction file. A case in point is the data of the purchase voucher 11 shown in Fig. 4. This purchase voucher 11-1 1-3 consists of a specification part 11b indicating the "purchase item", "unit purchase price" and "purchase quantity" and a header part 11a containing information common to the data in specification part 11b. In the case of Fig. 4, the information in the header part 11a includes the "supplier", "date of purchase" and "person in charge of purchase".

The voucher number and line number are now ex plained in further detail. The three purchase vouchers shown in Fig. 4 are identified by the voucher numbers of 0001-0003 and the data in the specification part 11b

of each voucher are assigned with line numbers and registered in the transaction file. Therefore, these data can be processed for the same "purchase item" even when they are different in "unit purchase price". Taking the purchase voucher 11 of Fig. 4 as an example, design pattern 1 through design pattern 5 are now described.

[Design patterns 1, 2] Skeletons ET010, ET011

The program of design patterns 1, 2 is completed by establishing the individual portion B (Fig. 2) of skeletons ET010 and ET011 by entry of information from the work station WS. To register voucher-type data in the transaction file, the use of design pattern 1 through design pattern 5 is preferred and particularly patterns 1, 2 are suitable when the quantity of information in the header portion 11a is large. And, as shown in Fig. 5A, the data in the header part 11a are registered in a header file 12, and the data in the specification part 11b are registered in a specification file 13 (Fig. 5B).

Taking the header file 12 of Fig. 5A as an example, the purchase voucher 11 of Fig. 4 has the fields of voucher number 12a, voucher date 12b, supplier name 12c, person in charge of purchase 12d, sum of purchase prices 12e. And, for example, "voucher number",

"voucher date" and "supplier name" are used as keys for data search.

On the other hand, the specification file 13 has, for example, the fields of voucher number 13a, voucher date 13b, supplier name 13c, voucher line number 13d, purchase item 13e, unit purchase price 13f, and purchase quantity 13g. And, for example, "voucher number", "voucher date", "supplier name", and "voucher line number" are used as data search keys. Since the "voucher line number" is registered and is a key for data search, it is possible to specify each one record equivalent of data by line number and perform deletion and copying on a line-by-line basis.

Now, referring to the flow chart of Fig. 6, the processing of design pattern 1 (skeleton ET010) is now briefly described. While this design pattern 1 implements the processing of registration, revision or deletion of voucher-type data, the actions of a "purchase voucher processing program" dealing with the data of the purchase voucher 11 of Fig. 4, as an embodiment of design pattern 1, are now explained.

When the "purchase voucher processing program" is selected in the menu (ST1), a screen for entry of key data for the header file 12 is displayed at the work station WS (ST2). Since the initial action mode of the

"purchase voucher processing program" is the registration mode, the processing for fresh registration of the data of the purchase voucher 11 is now explained.

In step ST2, the screen for entry of information in the voucher number, voucher date, and supplier name columns 12a, 12b and 12c is displayed. Therefore, the operator enters the corresponding information (hereinafter referred to as ID codes) using numerals and symbols. Depending on the design of the individual portion of skeleton ET010, "voucher number" is automatically selected and, in that case, entry of "voucher number" is not required.

As "voucher date", "supplier name", etc. are entered at the work station, a screen for entry of information other than the key data of the header file 12 is displayed (ST3). Since, in this example, the "person in charge of purchase" is to be entered, the operator enters the corresponding ID code for the "person in charge of purchase". Incidentally, if a function key F12 is pressed in step ST3, the sequence returns to step ST2 so that the screen for entry of the key data of header file 12 is displayed.

On the other hand, the Enter key of the work station is pressed in step ST3, the sequence proceeds to the step (ST4) for entry of the data of specifica-

tion part 11b of the purchase voucher 11 so that the operator enters the corresponding ID codes for "purchase item", "unit purchase price" and "purchase quantity". Upon completion of input of the data of specification part 11b of the purchase voucher 11, the operation for multiplication of "unit purchase price" by "purchase quantity" and the operation for addition of the products and the results of the respective operations are displayed at the work station (ST5). At the same time, a message urging the operator to confirm the input data is displayed (ST5).

As the operator presses the Enter key, the data displayed on the screen are registered in the header file 12 and in the corresponding areas of the specification file 13 (ST6). Where necessary, the contents of the master file (for example, an inventory master file) are updated accordingly (ST7). If a function key F12 is pressed in the stage of processing for entry of specification data (ST4) or confirmation message display processing (ST5), the sequence returns to the processing of step ST3. If a function key F3 is pressed during the processing of step ST2-ST5, the menu display of step ST1 resumes.

In the case of the program of design pattern 1, like other design patterns, window processing can be

performed. The term "window processing" means a processing which supports the processing of step ST2-ST5 by utilizing a part (window area W) of the CRT screen at the work station (Fig. 7). The window processing consists of essential processing and optional processing and the optional processing is selectively adopted or omitted in establishing the individual portion of skeletons ET010, ET011. Window processings which are optionally selected have been prepared by skeletons WN010, WN011, WN012 and WN020 which are described hereinafter.

In design patterns 1, 2, mode change processing with a function key F5 and line processing with a function key F9 are essential operations, while code search processing with a function key F4 is an optional processing selected at establishing the individual portion of the skeleton.

The term "mode change processing" means a processing for changing the current action mode of the program to the data registration mode, the revision mode or the deletion mode, and as a function key F5 is pressed in the processing stage of step ST2, mode change by window W processing is performed. For example, assuming that "2" is entered in the state shown in Fig. 7, the action mode of the "purchase voucher processing program" is

changed to the revision mode, and through the processings of step ST2-ST5, the data registered in the transaction files 12 and 13 are revised (ST6). On the revision mode, the data of the purchase voucher which have been registered in the transaction files 12, 13 are displayed at the work station during the processings of step ST2 and step ST4.

Assuming that "3" is entered in the state illustrated in Fig. 7, the action mode of the "purchase voucher processing program" is changed to the deletion mode and the purchase voucher data specified by the processing at step ST2 is deleted from the transaction files 12, 13 (ST6). On the deletion mode, only the registered purchase voucher data are displayed in the processing stage of step ST4, and the processing of step ST5 is not executed but skipped over.

The term "line processing" with function key F9 means a processing performed under the management of each one record equivalent of data in the specification file 13 by "line number" and typically an operation of copying the data on a given line to a position having a different line number corresponds to this processing.

The term "code search processing" with function key F4 means a processing for retrieval of the ID codes that must be entered. As mentioned above, all the

necessary information in step ST2-ST5 is to be entered using ID codes. Therefore, if the ID codes for "ABC Industrial Co.", "printer board" and "Mr. Sato" are not known, the corresponding ID codes can be searched by pressing the function key F4.

While the contents of operation of the "purchase voucher processing program" constituted by skeleton ET010 (design pattern 1) are as described above, substantially the same applies to skeleton ET011 (design pattern 2) and the only difference is that the operation for addition is not performed in step ST5. Therefore, skeleton ET011 (design pattern 2) is suitable for processing the registration, revision or deletion of voucher-type data and where it is not necessary to display the total sum or the like at the work station.

[Design patterns 3-5] Skeleton ET050, ET051, ET052

Design patterns 3-5 are characterized in that no header file is provided. They are suitable for registering both the data of header part 11a and the data of specification part 11b in the specification file 14, typically in the case of the voucher-type data shown in Fig. 4. A case in point is illustrated in Fig. 8. Thus, the specification file 14 consists of a header part 14A and a specification part 14B, with the header

part 14A having the discrete fields of voucher number 14A1, voucher date 14A2, supplier name 14A3, and person in charge of purchase 14A4 and the specification part 14B having the discrete fields of line number 14B1, purchase item 14B2, unit purchase price 14B3, purchase quantity 14B4 and total value 14B5.

The processing of design pattern 3 (skeleton ET050) is as shown in the flow chart of Fig. 9. Comparison of Fig. 9 with Fig. 6 indicates clearly that the processings at step ST1'-ST6' of Fig. 9 are substantially identical with the processings at steps ST1-ST6 of Fig. 6, and design pattern 3 is different from design pattern 1 in that no header file is provided.

Design pattern 4 (skeleton ET051) performs nearly the same processings as those of design pattern 3 shown in Fig. 9 but this skeleton ET051 is different from skeleton ET050 in that no operation for addition is performed in the processing at step ST5' of Fig. 9.

Design pattern 5 (skeleton ET052) also performs nearly the same processings as those of design pattern 3 shown in Fig. 9 but this skeleton ET052 is different from skeleton ET050 in that the processing at step ST3' of Fig. 9 is absent. Thus, design pattern 5 is the processing in the case where all the data in the header

part 14A of the specification file 14 are keys. A case in point is the case where, refering to the specification part 14A shown in Fig. 8, the voucher number 14A1, voucher date 14A2 and supplier name 14A3 are used as keys for data search and the field of person in charge of purchase 14A4 is not provided.

[Design patterns 6-7] Skeletons ET150, ET153

Design patterns 6-7 are suited for editing (registration and revision) or deletion of edit-type data. The term "edit-type data" means data not assigned with voucher number or line number, that is to say data such that specification data are used as keys for data search. A typical example of edit-type data is the data relevant to the case in which the daily record of maintenance and inspection made by a service man is registered in a transaction file. As illustrated in Fig. 10, while "Mr. Nakao", a service man, renders services such as "inspection" and "maintenance" for "RCC", "NTT" and other clients, a record of his daily activity must be registered in a transaction file, the charges for each one-month period be totalled for each client and a debit note be prepared. In such cases, the data of the service record to be registered in the transaction file does not require a voucher number or the like, nor need the specification data be provided

with line numbers. Therefore, the use of design patterns 6 and 7 is preferred.

Fig. 11 shows a transaction file 15 for registering the service record made by a service man. This specification file 15 consists of a header part 15A and a specification part 15B, with the header part 15A having the fields of service date 15A1, and service man 15A2 and the specification part 15B having the fields of client name 15B1, service content 15B2, service time in hours 15B3, unit service charge 15B4 and total value 15B5. And the service date 15A1, service man 15A2 and client name 15B1 are assumed here to be keys for data search.

The contents of action of a "service record registration program" constituted by design pattern 6 (skeleton ET150) are now described with reference to the flow chart of Fig. 12.

When the "service record registration program" is selected in the menu (ST10), a screen for entry of key data of the header part 15A is displayed at the work station (ST11). Since the "service date" and "service man" are keys in the above example, the operator inputs the corresponding numerals and ID code. Let it be assumed that the service record of September 24 which applies to the service man "Nakao" is entered (ST11).

Then, a screen for entry of information other than the keys of header part 15A is displayed (ST12). In this example, there is no information to be entered at step ST12, but the name of the section to which "Nakao" belongs may be entered. If a function key F12 is pressed in step ST12, the sequence returns to step ST11 for entry of "service date" and "service man".

On the other hand, if the Enter key is pressed in the step of ST12, a screen for entry of the data of specification part 15B of the specification file 15 is displayed. In this example, since the service records from September 21 to September 23 have already been registered for the service man "Nakao", this registered information is displayed at the work station. Therefore, in succession to this displayed information, the operator enters the ID codes and numerals corresponding to the "client name", "service content", "service time in hours" and "unit service charge" (ST13). If it is desired to revise or delete the registered data, correction of the data is made at this junction.

Upon completion of entry of the data of specification part 15B, the operation for multiplication of "unit service charge" by "service time in hours" and the operation for addition of the products are performed and the results of the respective operations are

displayed at the work station (ST14). At the same time, sorting is performed in the order of ID codes for "client name", and the information on the service from September 21 to September 24 for the service man "Nakao" is sorted by clinent and displayed at the work station. And a message urging the operator to confirm the entered data is displayed (ST14).

As the operator presses the Enter key, the data are registered in the specification file 15 (ST15). If there is a master file which need be updated, the contents of the master file are accordingly updated (ST16).

In the specification data entry processing (ST13) and the confirmation message display processing (ST14), if a function key F12 is pressed, the sequence returns to the processing at step ST12. If the function key F3 is pressed during the processing at step ST11-ST14, the sequence returns to menu screen processing (ST10).

The contents of action of the "service record registration program" on the edition mode has been described. When the action mode is the deletion mode, the data of the specified "service date" for the corresponding "service man" are deleted en bloc. The window processing here is identical with that of design pattern 1. Thus, mode change processing with function

key F5 and line processing with function key F9 are essential processings, while code search processing with function key F4 is an optional processing which is selected at establishing the individual portion of the skeleton.

Design pattern 7 (skeleton ET153) is substantially the same as design pattern 6, the only differences being that the processing at step ST12 is absent and the operation for addition at step ST14 is not performed. Thus, in the case where all the data in the header part 15A are key data and the addition operation for the specification part 15B is not required, the use of design pattern 7 is a judicious choice.

2. Inquiry system processing (Design patterns 8-14)

Design patterns 8-14 are processings pertinent to cases in which a command for information retrieval is made from the work station WS to the disk memory means 3. These patterns are classified into two major groups, namely design patterns 8-11 which are useful for situations where the data to be searched cannot be clearly specified and design patterns 12-14 which are useful for situations where the data inquired can be specified substantially clearly.

Design patterns 8-11 are constituted by combining skeleton SK010 with skeletons SK020, SK021, SK040 and

SK041, while design patterns 12-14 are constituted of skeletons SK050, SK051 and SK060, respectively. Thus,

Design pattern 8 = SK010 + SK020

Design pattern 9 = SK010 + SK021

Design pattern 10 = SK010 + SK040

Design pattern 11 = SK010 + SK041

Design pattern 12 = SK050

Design pattern 13 = SK051

Design pattern 14 = SK060.

Here, skeletons SK020, SK021, SK050 and SK051 are suitable for enquiring the data of the transaction file, while skeletons SK040, SK041 and SK060 are suitable for enquiring the data of the master file.

[Design pattern 8, design pattern 9] Skeletons SK010 + SK020, SK021

As shown in Fig. 13, skeleton SK010 is a skeleton comprising a processing (ST22) for entering the condition for enquired data and processings (ST23-ST25) for specifying the enquiry program. While design patterns 8, 9 are combinations of skeleton SK010 with skeletons SK020, SK021, the skeletons SK020 and SK021 are suitable for display of the transaction file data registered by the programs of design patterns 1-7.

Assuming that a "purchase voucher enquiry program" consists of skeletons SK010 and SK020 in combination,

the actions of design pattern 8 and design pattern 9 are now briefly described (Fig. 13). It is also assumed that the data of the purchase voucher 11 (Fig. 4) have already been registered in the header file 12 and specification file 13 as shown in Fig. 5.

While the "purchase voucher enquiry program" is selected from the menu (ST21), a screen for entry of the condition is first displayed at the work station (ST22). In this state the operator must enter the condition for the data inquired. It is assumed that the operator enters the date range of, for example, "October 3 October 10 of the year 1994" (ST22).

Then, the header file 12 is searched with the above data range used as a key and, as a result, all the corresponding data are displayed on the CRT screen at the work station (ST23; Fig. 14). As shown in Fig. 14, the list displayed at the work station contains an "action code" column. The term "action code" means a code which determines which of the enquiry programs outputting a display of enquired data should be used in implement the enquiry processing. Here, action codes "1" and "2" correspond to inquiry programs "PRG1" and "PRG2", respectively. In this example, it is assumed that the enquiry program "PRG1" is comprised of skeleton SK020 while inquiry program "PRG2" is comprised of

skeleton SK021. In response to designation of an action code, the processing of either the enquiry program "PRG1" or the enquiry program "PRG2" is started (ST25).

It is assumed that the second line of the screen of Fig. 14 is selected by moving the cursor and the enquiry program "PRG1" is selected with the action code. Then, with voucher number "1", voucher date "1994/10/3", and "ABC Industrial Co." used as search keys, the specification file 13 is searched and the corresponding voucher-type data are displayed at the work station. The above are the contens of processing of design pattern 8.

Substantially the same applies when the enquiry program "PRG2" is selected with the action code. The only difference is that the operation for multiplying unit purchase price by purchase quantity for each purchase item and the operation for addition of the products are performed and the results are displayed at the work station. The above are the contents of processing of design pattern 9. This skeleton SK021 (design pattern 9) is different from skeleton SK020 (design pattern 8) only in that said operations are performed and the results displayed.

[Design pattern 10] Skeletons SK010 + SK040

Design pattern 10 consists of skeletons SK010 and skeleton SK040 in combination. Skeleton SK040 is suitable for displaying the master file data registered by design patterns 15-22. For example, in the case where a "client master file" of the type illustrated in Fig. 18 is enquired, assuming that the enquiry condition is the client code "0020-0030" (ST22 in Fig. 13), a list of the corresponding firm names is displayed as shown in Fig. 15 (ST23). Therefore, the cursor is moved to the position of the client enquired and the action code is entered (ST24), whereupon the processing of the enquiry program specified by the action code is started and the corresponding information is displayed at the work station (ST25). In design pattern 10, this enquiry program is constituted by skeleton SK040, while skeleton SK040 implements the processing for displaying the master file data at the work station.

[Design pattern 11] Skeletons SK010 + SK041

Design pattern 11 is a design pattern consisting of skeleton SK010 and SK041 in combination and performs substantially the same processing as design pattern 10 for displaying the master file data at the work station. Fig. 16 is a flow chart showing an outline of processing of skeleton SK041. The difference of this skeleton from skeleton SK040 is that the data are

displayed on two screens at the work station. Therefore, design pattern 11 is effective in cases where one record abounds in data and cannot be accommodated within a single screen. As shown in Fig. 16, function keys F8 and F7 are used for switch-over between a first screen and a second screen.

[Design patterns 12, 13, 14] Skeletons SK050, SK051, SK060

Design patterns 12-14 are characterized in that a list display processing, such as done by skeleton SK010, is not performed. Thus, whereas design patterns 8-11 are useful for situations where data to be searched cannot be definitely specified, design patterns 12-14 are useful for situations where data enquired can be more or less definitely specified. And design patterns 12 and 13 are substantially the same as design patterns 8 and 9 except that the corresponding data are not displayed in a list form, and are suited for reference to the contents of the transaction file. Design pattern 14 is nearly identical with design pattern 10 and suitable for reference to the contents of the master file.

In the case of the program of design pattern 12 (skeleton SK050), a screen for key entry is first displayed at the work station (ST42) as shown in Fig.

17. It is assumed, for instance, that with regard to the specification file 14 of Fig. 8, the "voucher number", "voucher date" and the ID code of "supplier" are entered. Then, the specification file 14 is searched using them as keys and the specification data of the corresponding voucher number are displayed at the work station. If, in the state where the searched specification data are on display (ST43), a function key F12 is pressed, the sequence returns to step ST42 with the entered key data saved. Irrespective of the states, pressing the function key F3 erases the display and the menu (ST41) resumes. Code search processing with function key F4 is an optionally selected window processing and with any application program with this processing selected, ID codes can be searched in the processing stage of step ST42.

Skeleton SK051 (design pattern 13) performs substantially the same processing as skeleton SK050 but is different from the latter in that an operation for addition of enquired data is performed and the result displayed.

Skeleton SK060 (design pattern 14), which is substantially identical with skeleton SK040, is suitable for displaying the master file data registered by design patterns 15-22. For example, on entry of keys,

the corresponding information is retrieved from the master file and one record equivalent of data is displayed at the work station.

3. Master maintenance system processing (Design patterns 15-22)

The term "master maintenance system processing" means a processing for registration, revision, deletion or display of one record equivalent of data in the master file. For the master maintenance system processing, design patterns 15-22 are available for selective use according to the quantity of one record equivalent of data and skeletons MM010, MM011, MM020, MM021, MM030, MM031, MM050 and MM051 correspond to these design patterns, respectively. Thus, skeletons MM010 and MM011 are used for displaying one record equivalent of data for which one screen suffices. Skeletons MM020 and MM021 are used for displaying one record equivalent of data which requires two screens, skeletons MM030 and MM031 for one record equivalent of data which requires three screens, and skeletons MM050 and MM051 for one record equivalent of data which requires four screens.

Now, a further explanation is made taking the "client master file" of Fig. 18 as an example of said master file. This client master file has the fields of

client code 16a, firm name 16b, firm type 16c, address 16d, telephone number 16e, and FAX number 16f, with the client code being designated as the key for data search.

[Design pattern 15] Skeleton MM010

Now, the contents of action of the "client master maintenance program" constituted by skeleton MM010 are now described with reference to the flow chart of Fig. 19. Incidentally, this program is designed to register one record equivalent of data in the client master file 17 shown in Fig. 18.

When the "client master maintenance program" is selected in the menu (ST51), a screen display for key entry appears at the work station (ST52). Therefore, the operator enters the client ID code (ST52).

The action mode of the program is assumed to be the "registration mode". Then, a screen for entry of "firm name", "firm type", "address", "telephone number", and "FAX number" appears at the work station. Therefore, the operator enters the necessary information (ST53) and, then, presses the Enter key. Thereupon, a message for confirmation is displayed (ST54) and as the Enter key is pressed again, the data just entered is added to the client master file (ST55).

The "registration mode" has been described above.

If the function key F5 is pressed in the stage of step ST52, the action mode of the program can be changed by window processing. If the key is entered (ST52) after the action mode of the program has been changed to the "revision mode", the corresponding one record equivalent of data is displayed on the screen and, therefore, the corresponding position can be revised (ST53). And if the Enter key is pressed while a message for confirmation is on display (ST54), the contents of the client master file are revised (ST55).

On the "deletion mode", the designated one record equivalent of data is thoroughly deleted on key entry. On the "display mode", the designated one record equivalent of data is simply displayed at the work station and the processings at steps ST54 and ST55 are skipped over.

[Design pattern 16] Skeleton MM011

Design pattern 16 is characterized in that a copying function can be used in the stage of processing at step ST52 of Fig. 19 and otherwise it is identical with design pattern 15. The term "copying function" is a function by which one record equivalent of data already registered can be copied into a new record. Specifically, in design pattern 16, a screen for entry of "the original record to be copied" appears in the

processing stage of step ST52 (Fig. 20). When the client code, e.g. "0001", is entered in the input column of "the original record to be copied", one record equivalent of data on "ABC Industrial Co." is displayed on the screen at the work station. Therefore, this design pattern is convenient when the data to be added resembles the data on "ABC Industrial Co.", for only the portion differing from the data on "ABC Industrial Co." need be entered.

[Design patterns 17, 18] Skeletons MM020, MM021

Design pattern 17 is suitable for use when the entry of one record equivalent of data requires two screens and is characterized in that the processings at steps ST53 and ST54 of Fig. 19 are repeated twice. Design pattern 18 is characterized in that a copying function can be used and otherwise it is identical with design pattern 17.

[Design patterns 19, 20] Skeletons MM030, MM031

Design pattern 19 is suitable for situations where entry of one record equivalent of data requires three screens at the work station, and is characterized in that the processings at steps ST53 and ST54 of Fig. 19 are repeated for a total of 3 times. Design pattern 20 features a copying function and is otherwise identical with design pattern 19.

[Design patterns 21, 22] Skeletons MM050, MM051

Design pattern 21 is suitable for situations where entry of one record equivalent of data requires 5 screens at the work station, and is characterized in that the processings at steps ST53 and ST54 of Fig. 19 are repeated for a total of 5 times. Desion pattern 22 features a copying function and is otherwise identical with design pattern 21.

4. Window enquiry system processing (Design patterns 23-26)

The term "window enquiry system processing" means search processing using window W (Fig. 7) and for implementing the solutions of design patterns 23-26, skeletons WN010, WN011, WN012 and WN020 are provided. Processings of design patterns 22-26 are invariably data search processings which function when the function key F4 is pressed in the execution of a calling program, and while skeletons WN010, WN011 and WN012 are suited for situations where the data registered in the master file is searched, skeleton WN020 is suitable for situations where the data registered in the transaction file is searched.

[Design pattern 23] Skeleton WN010

The program of design pattern 23 is, for example, such that when the function key F4 is pressed in the

stage of step ST52 of the "client master maintenance program" (Fig. 19), "the client master file" is searched to retrieve the ID code of the "client". In the "client master maintenance program", the ID code of the client must be entered at step ST52. Whenever the function key F4 is pressed with the ID code input column left blank or the function key F4 is pressed with the ID input column filed with the code (ST52 and ST60), skeleton WN010 starts functioning (Fig. 21).

In skeleton WN010, it is first enquired whether a parameter exists (ST61) and if it does not exist, the processing sequence proceeds to step ST62 and if does, the sequence advances to step ST63. The term "parameter" is used in the above context to mean a search start code set at pressing the function key F4 in the operation of a calling program and, in the present example, the client ID code entered in the processing stage of step ST52 of the "client master maintenance program" corresponds to the search start code.

In the "client master maintenance program", assuming that the function key F4 is pressed without entry of the client ID code, the decision at step ST61 is performed and, then, a window display for entry of the search start condition appears (ST62). Since the client ID code is now searched in the present example,

the display shown in Fig. 22, for instance, corresponds to the window display for entry of the search start condition.

As shown in Fig. 22, in the program of design pattern 23, both a column 17a for entry of the client ID code and a column 17b for entry of the client name (initial) in kana character are displayed. It is sufficient to enter either the ID code or the kana character in the corresponding column. Assuming, for example, the ID code "0010" is entered in the input column 17a, the ID code and the corresponding client name are displayed side by side for all the codes including and subsequent to the ID code "0010" (ST63). On the other hand, the kana character "KA" is entered in the input column 17b, all the client names with initials including and subsequent to "KA" are displayed alongside their ID codes (ST63).

In the "client master maintenance program", where the client's ID code has already been entered, the processing at step ST62 is skipped over and the ID codes including and subsequent to the entered ID code are displayed in correspondence with the client names (ST63). Since, in either case, a plurality of client names and corresponding ID codes appear in the window display W, the operator can move the cursor to the

corresponding position and press the Enter key (ST63).

Assuming that the corresponding client ID code is "0005", the sequence returns to the "client master maintenance program" with the ID code saved (ST64), with the result that "0005" has already been entered in the client ID code input column when the calling program screen is displayed. If the function key F3 or the function key F12 is pressed in the processing stage of step ST62 or step ST63, the calling program resumes spontaneously. Thus, when the calling program is displayed, its client ID code input column remains blank.

[Design pattern 24] Skeleton WN011

Design pattern 24 (skeleton WN011) is substantially identical with design pattern 23 (skeleton WN010), the only difference being that the search start condition cannot be designated using a kana character in the processing stage of step ST 62. Therefore, design pattern 24 is useful for situations where the master file not employing kana characters is searched.

[Design pattern 25] Skeleton WN012

Design pattern 25 (skeleton WN012), like design patterns 23 and 24, is a processing for searching the master file data in the window of the screen. The contents of action are similar to those of design

pattern 24 (skeleton WN011) but this design pattern is characterized in that a search fixed value can be designated, in the processing stage of step ST62, independently of the search start code. The search fixed value is a value fixing one of search conditions. Taking the search of an "employee master file" containing individual data on the employees as an example, the design pattern 25 is effective when only the personnel manning a given department, e.g. Business Department, is searched.

The above may be specifically explained as follows. In the case of design pattern 25 (skeleton WN012), when the ID code of Business Department which is a search fixed value and the employee code "0020" which is a search start code are entered (ST62), all the data on the employees belonging to Business Department and assigned with employee codes including and subsequent to "0020" are displayed at the work station (ST63).

[Design pattern 26] Skeleton WN020

Design pattern 26 (skeleton WN020) is suitable for searching a transaction file, that is to say the data registered by the programs of design patterns 1-7.

Its action, illustrated in the flow chart of Fig. 21, is characterized in that a plurality of search

conditions can be entered at step ST62. For example, when the ID code of supplier name "ABC Industrial Co." and the voucher date "10/3" are entered in searching the specification file 14 of Fig. 8 (ST62), all the data satisfying the conditions are displayed in the window (ST63).

5. Voucher printing system processing (design patterns 27-34)

Design patterns 27-34 are processings for printing vouchers or slips. These design patterns are comprised of the following combinations of skeletons LT020, LT021, LT030 and LT040.

Design pattern 27 = LT020

Design pattern 28 = LT021

Design pattern 29 = LT010 + LT020

Design pattern 30 = LT010 + LT021

Design pattern 31 = LT010 + LT030 + LT020

Design pattern 32 = LT010 + LT030 + LT021

Design pattern 33 = LT040 + LT020

Design pattern 34 = LT040 + LT021

[Design pattern 27] Skeleton LT020

The processing outline of design pattern 27 (skeleton LT020) is shown in the flow chart of Fig. 23. This design pattern is suited for outputting a hard copy of contents of a file not containing demarcations

of data, such as a master file.

First, one record is read from the corresponding file (ST71) and in the case of EOF (end of file), the processing ends (ST72), and otherwise (if not EOF), an editing procedure for printing is performed (ST73). The editing procedure for printing is a processing for establishing an appropriate printing position or the like for one record equivalent of read data. Thereafter, it is queried whether one page equivalent of printing has been completed or not (ST74) and if the answer is affirmative, a header line for the next page is printed (ST75).

In the event a header line for another page is unnecessary because only one page suffices, one page equivalent of the edited data is printed as it is (ST76), the next one record equivalent of data is then read out (ST77) and the sequence returns to the processing at step ST72.

[Design pattern 28] Skeleton LT021

The processing outline of design pattern 28 (skeleton LT021) is shown in the flow chart of Fig. 24. This design pattern is characterized in that a processing (ST76') for printing a specification header HD is interposed between the processing at step ST75 and the processing at step ST76. The term "specification

header HD" means, for example, the contents of voucher number 14 A1, voucher date 14A2, supplier name 14A3 and person in charge 14A4 in the specification file 14 of Fig. 8, and every time the contents of one voucher have been printed, the contents of the header part 14A are printed (ST76').

The flow chart of Fig. 24 is now further described with reference to the hard copy shown in Fig. 25. First, one record is read from the corresponding file (ST71) and, if EOF (end of file), the processing ends (ST72), and otherwise (not EOF) an editing procedure for printing is performed (ST73). Thereafter, it is queried whether one page equivalent of printing has been completed (ST74) and if the answer is affirmative and page change has been done, the header line for another page is printed (ST75).

Where the procedure for page change is unnecessary, it is queried whether one voucher equivalent of data has been printed and if the answer is affirmative, the contents of the header part 14A as well as the heading is printed (ST76'). And one record equivalent of edited data is printed (ST76). Then, the next one record equivalent of data is read out (ST77) and the sequence returns to the processing of step ST72.

[Design patterns 29, 30] Skeletons LT010 + LT020, LT021]

Design patterns 29 and 30 are so constituted that the processings of skeletons LT020 and LT021 are respectively preceded by the processing of skeleton LT010.

As illustrated in Fig. 26, skeleton LT010 implements a processing for designating the range of the voucher to be printed. First, a screen for entry of the printing condition appears at the work station (ST81). Now, design pattern 29 is described taking the printing of contents of the "client master file" of Fig. 18 as an example. The operator for instance enters the ID code "0001-0010" as the printing condition (ST81), whereupon a message for confirmation is displayed (ST82). Then, a message reading "in printing" appears (ST83) and the processing of, for example, a "voucher printing program" constituted of skeleton LT020 starts operating (ST84).

While design pattern 29 has just been described, design pattern 30 is different from design pattern 29 only in that the voucher printing program called at step 84 is constituted of skeleton LT021.

[Design patterns 31, 32] T010 + LT030 + LT020, LT021

Design patterns 31 and 32 are such that the

processing of skeleton LT030 is included in the course of processing of design patterns 29 and 30. The processing of skeleton LT030, as shown in Fig. 27, is characterized in that the main file is thoroughly read once and the work file updated. The work file mentioned just above is typically a file for the adding operation.

Taking the printing of contents of the specification file 14 of Fig. 8 as an example, one record equivalent of data is read from the specification file 14 (ST91), and if not EOF, the read data are edited for printing (ST93). Then, the addition operation is performed for the products of "unit purchase price" multiplied by "purchase quantity" (ST94). Thereafter, the next one record equivalent of data is read out (ST95) and the sequence returns to the processing of step ST92.

After the processing of skeleton LT030, actual printing is implemented by the processing of either skeleton LT020 or skeleton LT021.

[Design patterns 33, 34] LT040 + LT020, LT021

Design patterns 33 and 34 are such that the processing of skeleton LT040 is performed before the processing of design patterns 27 and 38. As shown in Fig. 28, the processing of skeleton LT040, like that of

skeleton LT010, designates the condition for the data to be printed. The difference between skeletons LT010 and LT040 is that, in the case of skeleton LT040, a list of data satisfying the condition is displayed. Now, the flow chart of Fig. 28 is explained taking the printing of contents of the transaction files 12, 13 shown in Fig. 5 as an example.

First, a display for entry of the printing condition appears at the work station. It is now assumed that the operator designates the condition of "1994/10/1 -- 10/5" (ST101).

Then, all the data satisfying the above condition are displayed as shown in Fig. 14 (ST102). At the same time the action code input column also appears on the screen at the work station and, therefore, the operator inputs an appropriate action code (ST103).

Thereupon, a message for confirmation is displayed (ST104). Then, a message reading "in printing" appears (ST105) and the business form program is called (ST106). The business form program has been prepared either with skeleton LT020 or skeleton LT021 and the selection of design pattern 33 or design pattern 34 is dependent on which of skeleton LT020 and skeleton LT021 was used.

6. Batch updating system processing (Design patterns 35, 36)

[Design pattern 35] Skeleton BU020

Design pattern 35 (skeleton BU020) is a processing for situations where a file is updated without dialog at the work station. Typically, the case in which transaction file data are read out one record equivalent after another for updating the contents of a master file corresponds to this processing. Moreover, the procedure of constructing a temporary file (work file) corresponding to a voucher print image (e.g. Fig. 25) prior to print-out of the data accumulated in the transaction file also corresponds to this processing.

The above processing is now explained with reference to the flow chart of Fig. 29. First, the initial one record of the transaction file is read out (ST111) and, if not EOF, specification editing is performed (ST113). The specification editing means a processing for editing one record equivalent of read specification data into the format of the master file, temporary file or the like which is to be updated.

Then, based on one record equivalent of edited data, the master file, temporary file or the like is updated (ST114). Then, the next one record equivalent of data is read from the transaction file (ST115) and the sequence returns to the processing of step ST112.

[Design pattern 36] Skeleton BU010 + BU020

Design pattern 36 is a processing in which the updating condition is entered prior to the processing of design pattern 35.

As shown in Fig. 30, a display for entry of the updating condition first appears on the screen (ST121). Here, updating of a master file for preparing a debit note on the basis of data from September 21 to October 20 among the data registered in the transaction file of Fig. 11 is taken as an example. For example, the operator enters the updating condition of "9/21 -- 10/20" at the work station (ST121).

Then, a message for confirmation appears (ST122). As the operator presses the Enter key, a message reading "updating" appears (ST123) and the updating program constituted by skeleton BU020 is called (ST124). Thereafter, according to the procedure of design pattern 35, only the corresponding record of the transaction file of Fig. 11 is read out and the contents of the master file are updated.

Design patterns 1-36 have been described in detail. Now, the procedure for automatic program generation by dialog is now described. As an example, the contents of processing up to completion of an "employee master maintenance program" are now described. The "employee master maintenance program" is

an application program which is operated by the computer system of Fig. 1. It is a program by which the "employee master file" stored in the disk memory 3 is accessed from work stations WS1-WSn to register, revise or delete data on the individual employees.

The automatic program generation according to this invention is divided into the processing stages of (1a) physical file design (1b), screen image file design, (1c) business form file design, (2) registration of unit information, and (3) detailed program design. However, in the case of an "employee master maintenance program", where no voucher printing process is involved, automatic generation of the "employee master maintenance program" is completed only through physical file design (Fig. 31), screen image file design (Fig. 34), registration of unit information (Fig. 38) and detailed program design (Fig. 39).

(1a) Physical file design (Fig. 31)

With the computer set to the action mode of physical file design, design of a physical file is carried out by dialog. Since the task in this example is to design an "employee master file", it is sequentially decided what fields are to be provided for one record, how many bytes should be used for the region allocated to each field, and what field should be

selected for the data to be used as a data search key, and so on (ST131).

After completion of entry of all information, a data base design document is printed out in response to pertinent manipulation (ST132). Fig. 32 shows a data base design document. Since the "field name" 20a, "field ID" 20c, "field type" 20b, "field digit number" 20d, "data type" 20e, etc. are displayed according to a predetermined format, the contents of design can be ascertained at a glance. Moreover, as long as this data base design document is stored, anyone can easily know the contents of the "employee master file".

Upon completion of physical file design by dialog, source codes of the physical file are automatically generated (ST133). And in response to appropriate manipulation, a physical file source list is printed (ST134). Fig. 33 shows the source list. Thus, a hard copy of a source list in perfect correspondence to the design document of Fig. 32 is obtained.

Then, the source codes are compiled and a region for the "employee master file" is secured in the disk memory 3 (ST135). In this example, the ID of the "employee master file" is "RCMEMPP", and the ID fields of "EMPCDE"--"EMPUDT" are defined.

(1b) Screen image file design (Fig. 34)

Then, as the computer is set to the action mode of screen image file design, design of a screen image file is performed by dialog (ST141). In screen image file design, the skeleton to be used must be first determined. In this example, the "employee master maintenance program" is prepared by utilizing skeleton MM011 (design pattern 16).

As illustrated in Fig. 19, skeleton MM011 (design pattern 16) requires a screen for key entry (ST52), a screen for entry of specification data (ST53) and a screen displaying a message for confirmation with a concurrent display of specification data (ST54). Therefore, these are determined step by step.

After completion of all screen image design by the processing of step ST141, an appropriate manipulation is made, whereupon a screen image file design document is printed out (ST142). This screen image file design document sets forth detailed information on all the screen displays (ST52, ST53, ST54), the contents of the screen image design can be accurately ascertained from this screen image file design document. Figs. 35 and 36 show "image layouts" as parts of the screen image file design document. It is clear that at step ST52 of Fig. 19, the screen at the work station is as shown in Fig. 35 (FMT01 + FMT91) and that at steps 53 and 54 of

Fig. 19, the screen at the work station is as illustrated in Fig. 36 (FMT02 + FMT01 + FMT91).

Upon completion of design of the screen image file, source codes for the screen image file are automatically generated (ST143). And in response to appropriate manipulation, a physical file source list is printed out (ST144). Fig. 37 shows a part of the screen image file source list. Particularly, the 17th through 52nd lines show a source list relating to the screen image (FMT01) for entry of key data at step ST52 (Fig. 19).

Thereafter, the source codes are compiled and the object codes of the screen image file are generated (ST145). The name of this screen image file is "employee master maintenance" and the ID of the file is "RC00D001".

(2) Registration of unit information (Fig. 38)

Now, the computer should be set to the unit information registration mode by appropriate manipulation. On the unit information registration mode, an outline of the "employee master file maintenance program" is designated by dialog.

As shown in Fig. 38, the name of the program to be prepared and the skeleton to be used must be specified in the first place (ST151). Here, a program called

"employee master maintenance" is generated using skeleton MM011 as mentioned above. Therefore, these information can be simply entered from the work station (ST151).

Then, if the application program to be generated is called from a different program, the parameter to be received from the other program is determined (ST152). In this example, there is no parameter to be received from any other program.

Then, the files to be used by the "employee master maintenance program" are determined (ST153). In this example, the "employee master (RCMEMPP)" is used as the physical file and the "employee master maintenance (RC00D001)" is designated as the screen image file.

Finally, an external program to be called during the operation of the "employee master maintenance" program is determined (ST154). In this example, code search processing with function key F4 (Fig. 19) is used and, therefore, an external program for code search is designated. It is assumed that the program for code search has already been prepared using any of skeletons WN010, WN011 and WN012.

(4) Detailed program design (Fig. 39)

Completion of unit information registration is followed by the processing for detailed program design

shown in Fig. 39. Here, detailed information varying with individual processings is entered by dialog (ST161). Thus, since skeleton MM011 is divided into body portion A and individual portion B (Fig. 2) in order that a variety of processings can be made according to the procedure of Fig. 19, the intrinsic part of the "employee master maintenance program" is established in detail at the stage of step ST161.

After completion of the above processing, the program design document is printed in response to pertinent manipulation (ST162). In the automatic program generator shown in Fig. 1, a program design document for each of design patterns 1-36 is stored in semi-finished state in the disk memory 3. Therefore, after completion of the above detailed program design (ST161), a program design document in perfect correspondence with the particular program can be made available as a hard copy.

Further, when the detailed program design (ST161) has been completed, the source codes of the "employee master maintenance program" are automatically generated (ST163) by combining the information entered so far with the source program of skeleton MM011 which has been registered beforehand. And the source codes are compiled and an object program is generated (ST165).

Moreover, in response to pertinent manipulation, a program source list is printed (ST164).

Fig. 40 shows a part of the source program of skeleton MM011 and Fig. 41 shows the source codes, in part, of the printed "employee master maintenance program". While the source codes of the "employee master maintenance program" have columns filled in with the mark "//" or "/", the line with the mark "//" is a line representing a direct copy from the source program of skeleton MM011 and the line with the mark "/" or left blank is a line representing the automatic generation based on information input by dialog. The line designated "PLIST COMMON" in Fig. 40 is an insert of the lines 136-143 of Fig. 41, and the lines designated "KLIST COMMON" and "MAKING NG" are inserts from the lines 150-155 of Fig. 41. Moreover, the line designated "COPYRIGHT" of Fig. 40 corresponds to the line 159 of Fig. 41.

In the automatic program generator of Fig. 1, all application programs are generated by combining skeletons which are source programs in semi-finished state. These skeletons are high-quality programs prepared by expert programmers skilled in the particular programming language, and program design proceeds by dialog. Therefore, even a novice can implement an

automatic generation of application programs of constant quality easily and quickly.

Moreover, since a complete design document in perfect correspondence with the finished application program becomes available in the form of hard copy, anyone can accurately grasp the contents of processing of the program. Therefore, even when a change occurs in the operation or service of the organization, anybody can easily revise or make additions to the application program.

What is claimed is:

1. An automatic program generator comprising

a skeleton memory means storing skeletons which are source programs in semi-finished state,

a design document memory means storing design documents in semi-finished state which describe the contents of processing of said skeletons,

an individual information input means which functions in the design of application programs to receive individual information which varies with different application programs by dialog,

a program finishing means which edits said skeletons based on received individual information and generates source codes for respective application programs,

a design document finishing means which edits said design documents in semi-finished state on the basis of received individual information and finishes design documents corresponding to respective application programs, and

an output means which functions in response to pertinent manipulation to print out a source code list and/or design document of each finished application program.

2. The automatic program generator according to

* C L M S *

claim 1 wherein said skeleton memory means stores, in semi-finished state, at least first group programs which register data in a transaction file, second group programs for searching the data of a data file and display it on a terminal screen, third group programs for registering data in a master file, fourth group programs for searching the data of a data file and displaying data in the window of a terminal screen, fifth group programs for printing contents of a data file on a business form, and sixth group programs for updating contents of a master file or a temporary file using data of a transaction file.

3. The automatic program generator according to claim 1 wherein said design document memory means stores at least a physical file design document, a screen image file design document, a business form file design document and a program design document each in semi-finished state.

4. The automatic program generator according to claim 2 wherein said first group programs are classified into programs which deal with voucher-type data provided with a voucher number and a line number and programs which deal with edit-type data provided with neither a voucher number nor a line number.

5. The automatic program generator according to

claim 2 wherein said second group programs are classified into programs having a function of urging the operator to enter the condition of enquired data and displaying a list of all data satisfying said condition and programs not having the function of displaying said list.

6. The automatic program generator according to claim 2 wherein said third group programs are classified into a plurality of programs based on the quantitative relationship of one record equivalent of data and the quantity of data that can be displayed on a single terminal screen.

7. The automatic program generator according to claim 2 wherein said fourth group programs are classified into programs for searching a master file and programs for searching a transaction file.

8. The automatic program generator according to claim 2 wherein said sixth group programs are classified into programs having a function of urging the operator to enter the condition of data to be printed and displaying a list of all data satisfying that condition and programs not having the function of displaying said list.

9. The automatic program generator according to claim 1 wherein the programming language of said

* C L M S *

application programs is any of RPG language, C language, and COBOL language.