THE 2nd CIB W78+W74 SEMINAR
September 1990, Tokyo, Japan.

Non-Procedural Systems : the key to the Successful Implementation of IT in Construction Management

J.P. Pantouvakis*, M.Eng., M.Sc., Ph.D., C.Eng.
PLEGMA S.A. - Management Consultants,
41, Ethnikis Antistaseos st.,
152 32 Athens, Greece.

Abstract

This paper outlines the basic concepts and principles involved in the development of non-procedural systems for construction management. A non-procedural system allows the user to express his intentions about any of its features in a "what is required" (as opposed to the traditional "how to do it") manner. In this way the overall usability of the system is increased since its development and/or modification can be undertaken by the user himself. The case of using such non-procedural systems in construction management is strong due to the very nature of construction which may necessitate the use of different data and procedures from one project (and/or one company) to another. According to this paper the development of a non-procedural system is based upon two distinct, yet interrelated software modules: "SYS_SPE" which assists the user in the definition of the required system and "SYS_DEF" which implements the specified system on a particular software and hardware configuration.

1. Introduction

Construction management is as old as the practice of building under a contract. It incorporates such diverse functions as planning, estimating, cash flow forecasting, valuations, cost control and accounting. The need to handle large quantities of data and to produce results speedily has led to the introduction of computer systems in the field. A number of specialized programs for construction management have been developed over the years (CICA, 1979; Wager and Scoins, 1984) and a plethora of them is now available (CICE, 1990). These programs are based upon a model of the actual process which is implemented on a particular software and hardware configuration.

The fundamental problem with these construction management systems lies in the very nature of their designs; systems must be formalized before they are implemented. Moreover, these formalizations are hard-coded and, therefore, cannot change easily. The formalization of data and processes, however, is in principle contradictory to the nature of construction which aims at the production of unique projects. More precisely, there may be situations in which different data and procedures should be employed due to the peculiarities of the project in hand. In these cases the underlying formalization of the computer system hinders the effective management of the project. This may affect drastically the competitiveness of the construction company and/or the profitability of the project. The validity of the above argument in real world situations is demonstrated by the well-known difficulty of construction professionals to locate suitable construction management software (CICA, 1987, 1990).

The problem of developing computer systems for construction management in such a way as to ensure that different practices can be employed is addressed in this paper. The solution is sought in the development of non-procedural systems. Non-procedural systems allow the user to express his intentions about their functions or features in a "what is required" (as opposed to the traditional "how to do it") manner. In this fashion the following three objectives are fulfilled:

* The work presented in this paper started at the Department of Computer Science, University of Nottingham, U.K.

(a) The flexibility and generality of approach of the computer system are ensured.
(b) Any incorporated formalizations are defined by the user himself who is considered to be an expert in his field.
(c) When modifications to the underlying formalizations are required, these can be implemented easily (i.e. in a non-procedural manner) by the user.

In the reminder of this paper, the basic principles involved in the development of non-procedural systems will be presented. The necessary software will also be identified and defined. Finally, the benefits accruing from the development of non-procedural systems for construction management will be discussed.

2. The Development of a Non-Procedural System

The development of a non-procedural system is based upon the following principle:

The specification of a computer system, i.e. its menu structure, input, processes and output can be thought of as data (termed system definition data). This data is the result of a process by which the user defines his intentions about the features of the computer system. This data is also the input of a second process by which the required system is implemented.

The process is presented schematically on Fig. 1.

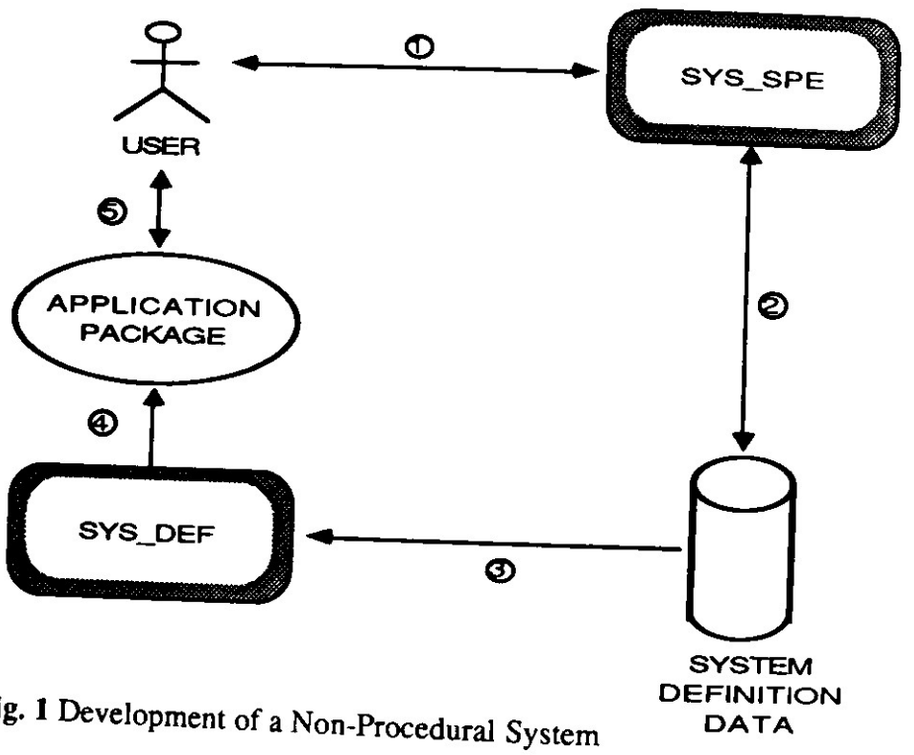

Fig. 1 Development of a Non-Procedural System

Initially the user interacts with a specially developed software module (named "SYS_SPE") in which the required characteristics of the system are defined. Subsequently "SYS_SPE" performs the required calculations (e.g. normalization of data, integrity and consistency checking etc) in order to convert the user intentions to the logical model of the system. The logical model of the system is stored in the "System Definition Data" file.

A separate software module (named "SYS_DEF") accepts as input the logical model of the system and generates as output the corresponding implementation of the system on a particular computer configuration.

The user will then interact with the so-produced implementation (termed "application package" in Fig. 1).

In a similar manner, the user can modify a non-procedural system. "SYS_SPE" will provide the system definition data in an intelligible (to the user) format. This data can then be edited by the user so

that the required modifications will be defined. The system will then be re-generated in a similar manner to that outlined above.

It should be emphasized at this stage that the development environment of a non-procedural system (i.e. the software modules "SYS_SPE" and "SYS_DEF" and the system definition data files) is used to specify the characteristics of the required system. The application program produced is independent of the development environment and can be thought of as a stand-alone application. The implication is obvious; one does not need to be capable of specifying a computer system to be able to use the application package produced by a non-procedural development environment. In fact, the application package produced should be similar in functionality, ease of use and speed of operation to other construction management systems already available. The great benefit of the approach, however, lies in the fact that any implementation of the application package can be modified easily to cater for different requirements.

In the next two sections, the design of the system specification ("SYS_SPE") and the system definition modules ("SYS_DEF") will be presented in more detail. It should be noted at this stage that the development of these two systems was based upon the following two assumptions:

(a) That the end-user interface of the produced application package abides to pre-defined rules. More precisely that the screens presented by the application package are divided in all cases in three sub-parts :
   1. The upper part which provides information about the position of the user within an application (i.e. where he is and/or what he is doing).
   2. The middle part which is used to display data and/or results and to enquire for input.
   3. The bottom part which is used to display error and/or help messages and to display verification questions when they are needed (e.g. "Press ESC to abandon" or "Are you sure you want to delete this file? (Y/N)" etc).

(b) That the target computer configuration is standard (e.g. PC/AT compatible hardware, with a EGAVGA graphics card, under the MS-DOS operating system). It should be noted that other "SYS_DEF" modules will be needed for other computer configurations.

3. The System Specification Module

The system specification module leads the user to an eight step process by which the characteristics of the required system are defined. These steps are presented in outline in Fig. 2 and are explained below.

Step 1 : The user provides the names of the data items needed by his application. These data items may have been produced by using standard systems analysis techniques (e.g. SSADM- see Cutts (1987)). As an example, a simple bill of quantities system is presented in Fig. 2. The data items needed are the project identification code ("project"), the item code ("item"), the description of the item ("description") and the corresponding quantity ("quantity").

Step 2 : The user provides the relationships between the data items specified in step 1. A relationship (or in database jargon a "functional dependency") exists when the value of a data item depends on the value of another data item. For example, the data items "project" and "item" determine the value of "quantity" (see Fig. 2). This means that for a particular value of "project" and "item" there is only one value for "quantity".

Step 3 : The characteristics of the data items defined in step 1 are given. For example, (see Fig. 2), the data item "quantity" will be termed "qty" by the computer system, and it will be a 10 character wide number with two decimal places.

Once these three steps have been completed, the system will eliminate redundancy and inconsistency and it will organize the structures of the appropriate data files (see also Pantouvakis (1988)). This information will be stored in the data dictionary.

SN	INPUT	FORMAT	EXAMPLE	OUTPUT
1	Data Item Names	fieldi, fieldj, ...	Project, Item, Description, Quantity	DATA DICTIONARY
2	Data Relationships	FOR (fieldi, fieldj, ...) THERE IS EXACTLY ONE (fieldk, fieldl, ...)	FOR (Project, Item) THERE IS EXACTLY ONE (Quantity)	
3	Data Characteristics	fieldi[name:type:width:dec]	Quantity[qty:Numeric:10:2]	
4	Integrity Constraints	Logical Expressions	project.qty >= 0	INTEGRITY DICTIONARY
5	Data Input Screens	Definition via a specially developed program by which the user can "draw" the screen layout	Project Code Item Description Quantity	SCREEN DICTIONARY
6	Process Definitions	Usage of a Very High Level Language (see Fig. 3)	* FILE : INCREASE.PRG * Increase Quantities by 10 qty := qty + 10	PROGRAM DICTIONARY
7	Report Definitions	Definition via a specially developed program by which the user can "draw" the report layout	Item | Description | Quantity	REPORT DICTIONARY
8	Menu Structure Definition	Definition via a specially developed program by which the user can define the menu structure	Increase Quantitities ... Exit	MENU STRUCTURE DICTIONARY

Fig. 2 The SYS_SPE Module

Step 4 : Given the data dictionary and the semantics of the application, the user can decide upon the kind of constraints that should be imposed on the data stored in the files of the system. In Fig. 2, such an integrity constraint is stated as : "project.qty >= 0" and should be read as : "the field 'qty' of the file 'project' should be greater than or equal to zero". An error message can also be defined to appear every time the integrity constraint is about to be violated. Similarly to functional dependencies, integrity constraints are checked for redundancy and inconsistency. The user can also define an error message at this stage which will appear every time this constraint is violated. All this information (i.e. integrity constraint and error message) is stored in the integrity dictionary.

Step 5 : The user defines any number of input screens to be used in conjunction with a data file. These screens provide the end-user interface every time data should be appended, deleted or modified to a file. The input screens are named so that they can be distinguished. An example is shown in Fig. 2. The name of the screen, its layout and the file and fields of that file with which it is associated are stored in the screen dictionary.

Step 6 : The user defines the processes of the system. For this purpose a special very high level language was developed. This language has been based upon previous experiences of the writer of this paper in developing non-procedural systems for construction management (see also Pantouvakis (1990)). An overview of this language is presented in Fig. 3. Note that this language is very simple (it contains only nineteen commands) and strictly non-procedural (it does not contain any repetition and/or recursive commands). As such, it should be very easy to learn and use. The language is accompanied by a set of functions (such as SUM() etc-see also Fig. 4). A simple example of a process file developed in this language is presented in Fig.2. The command "qty := qty + 10" should be understood as "increase all the 'qty' fields in file 'project' by 10". These programs along with the names of the data files and the fields the operate upon are stored in the program dictionary.

Step 7 : The user defines the format of the output to be produced by the system. The format of the output is termed a "report". An example is presented in Fig. 2. The report layout, its name, the file(s) with which it is associated and the fields of that (those) file(s) that it contains are stored in the report dictionary.

Step 8 : The user specifies the menu structure of the required application. For each option he specifies the text to be displayed (e.g. "Increase Quantities" in the example of Fig. 2) and the program file (or command) that should be executed (e.g. "increase.prg" in this case). The information relating to the menu structure of an application is stored in the menu structure dictionary.

4. The System Development Module

The system development module ("SYS_DEF") accepts as input the system definition data (i.e. the logical model of the system) and produces as output the implementation of the required system on a particular software and hardware configuration. The main functions of "SYS_DEF" are as follows:

(a) The implementation of the logical data model on a particular computer system (i.e. the implementation of the data files)
(b) The implementation of the data entry programs (addition, deletion and update) incorporating the specified data input screens and any integrity constraints.
(c) The implementation of the processes incorporating any integrity constraints and report definitions (i.e. the conversion of the processes as defined in the very high level language (see Fig. 3 and Fig. 4) to executable files).
(d) The implementation of the required reports of the system.
(e) The implementation of the menu structure of the application.

5. The Benefits of Non-Procedural Systems

The main benefits of non-procedural systems are:

(a) Productivity Improvement : A construction management package can be developed much more speedily and, therefore, at a significantly reduced cost.
(b) Reduction of Skill Requirements : The skills available to design complex computer systems are in very short supply. By using the "SYS_SPE" and "SYS_DEF" modules the skill requirements are greatly reduced.
(c) Quality and Consistency Improvement : All the systems generated by "SYS_DEF" are of the same quality. The consistency of the end-user interface is also guarantied.
(d) Early Prototyping Support : By using the system, the production of prototypes is facilitated. This means that more than one designs can be examined in order to select the one that meets the requirements of the application precisely.
(e) Maintainability of the Application Package : The modules "SYS_SPE" and "SYS_DEF" can handle effectively the modifications of the application package which may be requested due to changes in practices, legislation etc.

SN	COMMAND	DESCRIPTION
1	DISPLAY <message> [GET <variable> [INITIAL <initial_value> [FORMAT <format>]]]	Print <message> on the screen and get from the user the value of <variable> which has initial value <initial_value>. The value of <variable> is to be printed/stored in the format specified.
2	A := B	Assignement
3	IF <condition> THEN <action1> ELSE <action2>	IF ... THEN ... ELSE ... Construct
4	FOR <condition> IN <file> DO <action>	For the records in <file> that satisfy the <condition> do <action>
5	SELECT <file>.<field>	When executed, leads to a full-screen operation in which the user is allowed to select records from <file> by (a) selecting records one by one (b) by specifying a condition (c) by using wildchars
6	WITH SELECTION <file>.<field> DO <action>	With the selected records (see command 5 above) do <action>
7	CALL <file>	Transfer control to <file>
8	... FOR <condition>	Defines the scope of a command
9	ADD INTERACTIVELY TO <file> SCREEN <screen> [CONSTANT <fieldi> = <expressioni>, <fieldj> = <exprtessionj>...] [REPLACE <fieldk> WITH <expressionk>, <fieldl> WITH <expressionl>...]	The Data Entry Operations (addition, deletion, update). They lead to full-screen operations SCREEN : defines the name of the screen to be used CONSTANT:defines that the values of <fieldi>,<fieldj> will have a constant value during the operation REPLACE: replaces the value of <fieldk>, fieldl> ... with the expressions provided VERIFY OFF : indicates that the user will not be asked to verify the deletion of a record
10	DELETE INTERACTIVELY FROM <file> SCREEN <screen> [VERIFY OFF]	
11	UPDATE INTERACTIVELY <file> SCREEN <screen> [REPLACE <fieldi> WITH <expressioni>, <fieldj> WITH <expressionj> ...]	
12	APPEND TO <file>.<fieldi> = <expressioni>, <fieldj> = <expressionj> ...	Data Entry operations which do not lead to full-screen operations and thus can be used by a program
13	DELETE RECORDS FROM <file>	
14	MODIFY <file>.<fieldi> WITH <expressioni> <fieldj> WITH <expressionj> ...	
15	COPY <filei>.<fieldj>, <filek>.<fieldl> ... TO <filem>.<fieldn>, <filep>.<fieldq>	File operations. More specifically: COPY fields from one file to another CREATE a file from the given definitions DELETE a file RENAME a file
16	CREATE <file> with <fieldi>:<typei>:<widthi>:<deci>:<fieldj>:<typej>:<widthj>:<decj>	
17	DELETE <file>	
18	RENAME <file1> TO <file2>	
19	BEGIN ... END	Defines blocks of commands (in a similar manner to the corresponding statements in PASCAL)

Fig. 3 Concise Description of the Very High Level Language

SN	FUNCTION	DESCRIPTION
1	SUM(<file>.<field>)	Returns the sum of the the field <field> of the file <file>
2	MATCHES(<filei>.<fieldj>, <filek>.<fieldl>)	Returns true if the values of the filei.fieldj are equal to the values of filek.fieldl
3	UPPER(<variable>)	Returns the string <variable> given in upper case letters
4	LTRIM(<variable>)	Returns the string <variable> without leading epaces
5	RTRIM(<variable>)	Returns the string <variable> without trailing spaces
6	LEN(<variable>)	Returns the lenght of the <variable>
7	LEFT(<variable>, pos)	Returns the first <pos> characters of the variable <variable>
8	RIGHT(<variable>, pos)	Returns the last <pos> caharacters of the variable <variable>

Fig. 4 Basic Functions

6. Conclusion

The traditional approach to the development of computer systems for construction management is inadequate for the needs of the construction industry. A new approach is, therefore needed. This paper suggests that the solution should be sought in the so-called non-procedural systems. These systems can be specified and implemented without necessitating the development of code by the user himself. A development approach and the design of appropriate software is also presented in this paper. The main thesis put forward by this paper is that such non-procedural systems for construction management can assist significantly in the development of more usable, functional, user-friendly and economical software.

References

(CICA, 1979) Construction Industry Computer Association (1979) *Construction Programs for Construction Management*, Cambridge, U.K.

(CICA, 1987) Peat Marwick McLintock and Construction Industry Computer Association (1987) *Building on IT, A survey of Information Technology Trends and Needs for the Construction Industry*, Cambridge, U.K.

(CICA, 1990) Peat Marwick McLintock and Construction Industry Computer Association (1990) *Building on IT for the 1990s*, Cambridge, U.K.

(CICE, 1990) Construction Industry Computer Association (1990) *Construction Industry Computer Exhibition*, Barbican Centre, London, U.K.

(Cutts, 1987) Cutts, G. (1987) *Structured Systems Analysis and Design Methodology*, Paradigm Press, U.K.

(Pantouvakis, 1988) Pantouvakis, J.P (1988) *Software Tools for the Implementation of Relational Databases Describing the Building Product*, 1st CIB W78+W74 Seminar, Lund, Sweden.

(Pantouvakis, 1990) Pantouvakis, J.P. (1990) *Declarative - Configurable Estimating Systems for the Construction Industry*, Ph.D. Thesis, University of Nottingham, U.K.

(Wager and Scoins, 1984) Wager, D. and Scoins, S. (1984) *More Construction Programs for Construction Management*, Construction Industry Computer Association , Cambridge, U.K.