Technological University Dublin

# ARROW@TU Dublin

Conference papers

Digital Media Centre

1992-01-01

# A Case for the use of Application Generators in the Creation of Software for the Hotel Industry

Peter O'Connor
*Technological University Dublin*

Ciaran McDonnell
*Technological University Dublin*, cmdonnell@tudublin.ie

Follow this and additional works at: https://arrow.tudublin.ie/dmccon

Part of the Databases and Information Systems Commons, Hospitality Administration and Management Commons, and the Technology and Innovation Commons

## Recommended Citation

# A Case For The Use Of Application Generators In The Creation Of Software For The Hotel Industry.

**Peter O'Connor**
B.Sc. (Mgmt), M.Sc., H. Dip.in H & C Mgmt, MIHCI, LICS.

**Ciaran McDonnell**
B.Sc., M.Sc., Ph.D. (Comp  Science), MICS
Research  Supervisor

**Department Of Computing and Information Technology, Dublin College Of Catering.**

## Current Practice In The Irish Hotel Industry.

Recent research has shown that the Irish hotel industry has become relatively computerised[1]. Nearly 90% of the hotels surveyed now use computers of some sort in their day-to-day operations, with approximately half of the systems having been installed in the year prior to the survey.

The most commonly computerised function was Back Office accounting - debtors, creditors, general ledger and payroll systems - with 87% of hotels surveyed using computers in this area. Front Office systems, whose normal functions are to handle advanced reservation, register the guest, and to maintain house status and guest folios[2], were used in 47% of hotels. Stock Control Packages, which assist in purchasing, storing and issuing stock[3], were used to control drink stocks in 61% of cases, and food stock in 50%.

## The Problem

The survey revealed dissatisfaction with many of the software packages in use in hotels. The majority of the complaints have two basic causes:

Firstly/ the software in use was mostly of an off-the-shelf nature - it was designed with the needs of the industry as a whole in mind and is thus general in function and methods of operation[4]. This leads to several problems. Some packages do not meet the data storage needs of some users, having

to be supplemented by manual systems. For example, many hotels have to store reservation information concerning extra beds and cots manually as their Front Office systems cannot be modified to incorporate the extra information. Others use methods of operation which are not suited to the needs of the industry. For example, some stock control packages were originally developed for use in other industries. Such systems are unable to take into account hotel industry-specific factors such as deteriorating food quality during storage and shrinkage during cooking. In many cases, the manual procedures used by the hotel have had to be modified to allow for the limitations of the software product.

Managers in the industry feel that instead of making life simpler and more efficient, the introduction of the computerised system makes it more difficult[5].

Secondly, modifications which could more closely match the program to a client's needs - thereby solving some of the above problems - appear to be difficult and costly to obtain. For example, these modifications must be carried out by skilled programmers and can normally involve a long time-delay. The leading supplier of hotel and catering software to the Irish hotel and catering industry estimates that it will spend the next three years servicing existing installations before it can begin to develop new systems. In addition, such modifications can introduce inconsistencies which make the software "brittle" and ultimately unreliable[6].

As the existing software packages have such problems, the obvious answer would appear to be to have all new software package custom written to meet the needs of a particular hotel from the outset. In this way, the hotel's exact requirements are met and the system interfaces perfectly with existing manual procedures.

The normal approach to such a development has been to use traditional programming languages, such as BASIC or OCBOL. (Indeed BASIC, in one of its many flavours, has been used to develop all the software produced by the main supplier of software to the Irish hotel and catering industry.) A specification definition is drawn up from the needs and the program code is written from this by programmers over a period of time. A further

period is spent testing and debugging the code to ensure that the package actually works and matches the specification[7].

There are well-documented problems with this development process. Firstly, such software is expensive as all the costs of development must be borne by the purchaser. Secondly, even where a user "knows" exactly what he requires the software package to do, there may be problems in practice in successfully communicating these requirements to the Systems Analyst / Programmer[8].
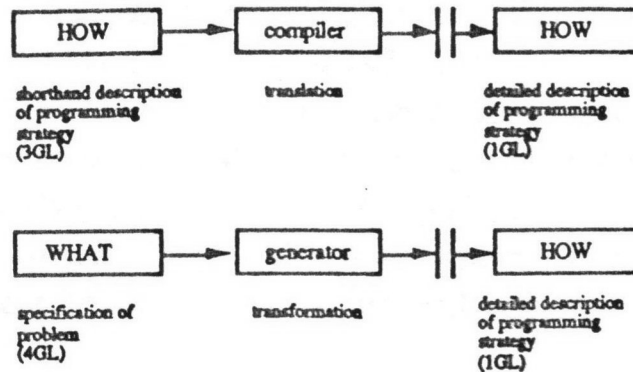
Lastly, even with all other things going well, there may be a significant time lag between the program's specification and its implementation. During this delay, the user's requirements may change, resulting in applications which are obsolete before they are even implemented[9].
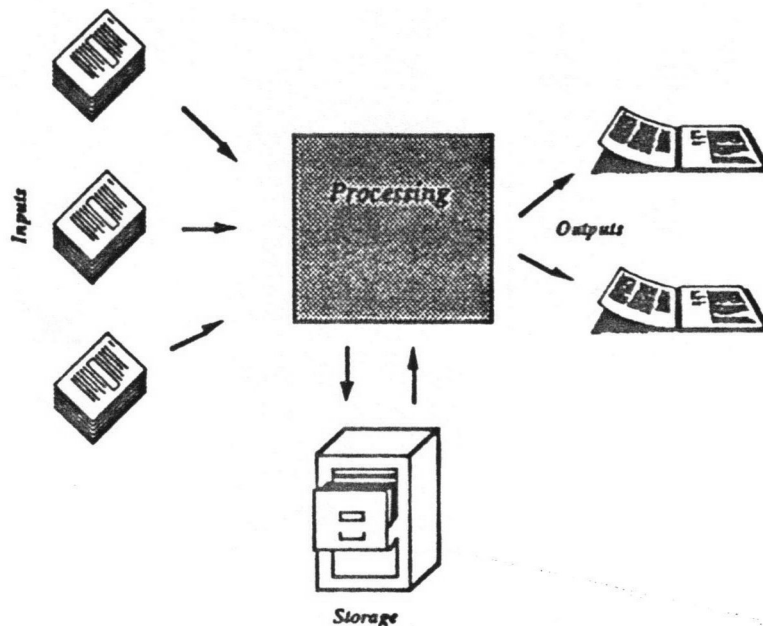
**The Solution?**

This article proposes that more effective software could be produced using application generators. There is no one definition of an application generator. Instead, it is held to be a collection of related tools and facilities which together allow the creation of software applications without the use of conventional programming languages[10]. These facilities include a relational database for the storage of data, a forms generator for the creation of data entry screens and a report generator for the creation of reports based on the information stored in the tables of the database.

The process of creating an application using an application generator differs greatly from more traditional approaches, as can be seen from figure one. Using traditional or third generation programming languages, the computer must be instructed how to perform each individual action necessary to complete a process. However, when an application generator
is used, the computer need only be told what is to be done, using a relatively simple menu driven interface (such as the Control Centre in Dbase IV), and it will generate the step by Step instructions to carry out the task itself. Naturally the possibilities for errors and the development time for an application are greatly reduced as a result[11].

## Figure One
### Difference between Third and Fourth Generation programming languages[12].



| HOW | → | compiler | →\|\|→ | HOW |
| --- | | --- | | --- |
| shorthand description of programming strategy (3GL) | | translation | | detailed description of programming strategy (1GL) |

| WHAT | → | generator | →\|\|→ | HOW |
| --- | | --- | | --- |
| specification of problem (4GL) | | transformation | | detailed description of programming strategy (1GL) |

## Figure Two
### A Typical Business System.



Application generators cannot be used to create every type of program. They are, however, most useful in creating what could be referred to as "business systems" (Figure two) - _ systems where data is inputted, processed in various ways, stored in both the raw and the processed states and output in response to requests from the user.

A research project was undertaken in the Dublin College Of Catering to determine the feasibility of creating hotel related software. Initially a prototype Front Office system was created using a variety of

application generator-like tools. The application generation tools which proved most suitable were then used to successfully create an integrated suite of hotel software packages including a Front Office System, a Hotel Accounting System, a Stock Control Package, a Recipe Costing System and a Conference / Banqueting System. These are of a standard comparable with commercially available packages but have the advantage of flexibility in that they can be easily and successfully modified to take different user requirements into account.

Using application generators also has other advantages over traditional methods:

Firstly the use of application generators can solve the requirements identification and definition problem noted earlier, as the flexibility of the application generator allows the specification to be changed, both during and after development. Therefore incorrectly defined or changed requirements can be corrected, thus, in most cases, reducing the danger of obsolesence.

Secondly, a prototyping approach can be used to make the above-mentioned flexibility more effective. Applications are quickly developed - thereby reducing the time delay - and can subsequently be adapted to take changing requirements into account. This has a further advantage in that the user quickly sees the look of the completed application and the manner in which it will behave[13]. The user can apply his industry specific knowledge to the prototype and changes can be incorporated if necessary, making the resulting software package more useful, efficient and effective[14].

The amount of technical knowledge necessary to modify applications is also greatly reduced[15]. While it would generally be beyond the capabilities of a "normal" manager to carry out modifications to an application using a traditional programming language, it should be possible for a computer literate manager to customise an application created using an application generator by, for example, creating new reports as information needs change. In this manner, the manager's changing information needs are satiated as old reports can be modified and new reports created whenever required.


**Conclusion**

The two methods of purchasing computer software - off-the-shelf and custom written - have proven to be

unsatisfactory in the Irish hotel industry. Software purchased using the former method is too general in nature and is difficult to modify to more closely match users' needs. On the other hand, use of the latter method is costly and still does not guarantee effective software packages.

The use of Application Generators is suggested as a solution. These allow certain types of software packages to be created quickly and cheaply. Furthermore, these packages are flexible and can be modified easily whenever necessary to take into account changing user requirements.

**Biographical Note:**

Peter O'Connor originally trained and worked in hotel management. Winner of the P.V. Doyle Post-Graduate research scholarship, he completed a Masters Degree in the area of Application Generators applied to information systems in the hospitality industry. He currently lectures in Computer applications in the Dublin College of Catering.

Ciaran McDonnell has worked within the computer industry, and education for twenty years. He holds a Ph.D. in Computer Science and has worked for the past eight years at the Dublin College of Catering as Head of Computing and Information Technology.

**References:**

1.	O'Connor, Peter, "The Use Of Micro-Computer Based Program Generators  In The  Design And Creation Of Information  Systems For The Irish Hotel And Catering Industry", M.Sc. Thesis, University Of Dublin, 1991.

2.	Gamble Paul R., "Small Computers And Hospitality Management", Hutchinson, 1984.

3.	Mitchell, Ian D, Phipps, Diana K. and Austin Elaine, "The HOTECH Directory", Oxford Polytechnic, Vol. 2, No. 1, January 1989.

4.	Durbin, G., "Off The Rack: The Case For Packaged Systems", Information Strategy: The Executive's Journal, Summer 1987, pp 11-15.

5.	O' Connor, Peter,  "The Use Of Micro Computer Based Program  Generators  In  The Design And  Creation  Of Information Systems  For  The  Irish  Hotel  And

Catering Industry", M.Sc.   Thesis, University Of Dublin,1991

6.    McFadden F. and Discenza R., "Confronting The Software Crisis", Business Horizons, November / December 1987, pp 68-74


7.    O'Donnell D.J. and March S.T., "End user Computing Environments – Finding a Balance Between Productivity and Control", Information and Management, No. 13, 1987, pp 77-84.

8.    Dos Santos, B.L., "MIS Project Management: A Contingency Approach", Management Decisions, Vol. 26, No. 6, pp 22-28.

9.    Meng C.E., "Prototyping, Participative and Phenomenological Approaches to Information Systems Development", Journal Of Systems Management, 12 August 1987, pp 12-5.

10.    Hamilton R., "Fourth Generation Languages And End Users – DP Management Attitudes", Industrial Management And Data Systems, No. 1, 1989, p13.

11.    Chang-Yang Lin, "User-Friendly Applications Development In A Relational Database Environment", Journal Of Systems Management, Spring 1988, pp 45-8.

12.    McLoughlin B and Keane D, "Fourth Generation Languages – Fact or Fiction", Irish Computer, June 1987, pp 35-37.

13.    Matos V.M., Misra S.k. and Jalics P.J., "End-User Criteria For Selecting Fourth Generation Report Generators", Journal Of Systems Management, September 1989, pp 28-32.

14.    Sprague R.H., Jr. and McNurrin B.C., "Information Systems Management In Practice", 2nd edition, Prentice Hall, USA, 1989 p 266.

15.    Matos V.M., Misra S.k. and Jalics P.J., "End-User Criteria For Selecting Fourth Generation Report Generators", Journal Of Systems Management, September 1989, pp 28-32.