

# EXPERIENCES OF USING A 4GL FOR MANUFACTURING PLANNING

*Karen Townrow, Ian Cannings, Nairn Kingfisher Limited and Jean Tunnicliffe-Wilson and Mike Pidd, Lancaster University*

## THE COMPANY

Nairn Kingfisher Limited, a member of the FORBO Group, specialises in the manufacture of wallcoverings. Widely established throughout the UK and abroad, Nairn Kingfisher Limited employs 500 people on the Lancaster site and generates annual sales valued at 20M pounds from approximately 1400 different wallcovering patterns.

Wallpaper is a fashion product, and the majority of patterns have only a two year life, however a guaranteed supply is promised to customers throughout the two years. As the company has diversified and extended its product range, the routing of the products through the factory has become more complex. Substantial recent investment has simplified the production of certain patterns and generally speeded up the production process. The products require relatively few raw materials compared to the number of different wallcoverings produced and are mainly supplied from stock.

## THE PROJECT

### Production Planning System : an overview

In order to reduce inventory and to provide better customer service, a project was set up to implement a Production Planning System. Its three sub-systems attempt to match the production requirements with the available machine capacity and represent three levels of a planning hierarchy:-

- i) Monthly planning looks several months ahead in terms of groups of wallcovering patterns, starting from the annual company budget.

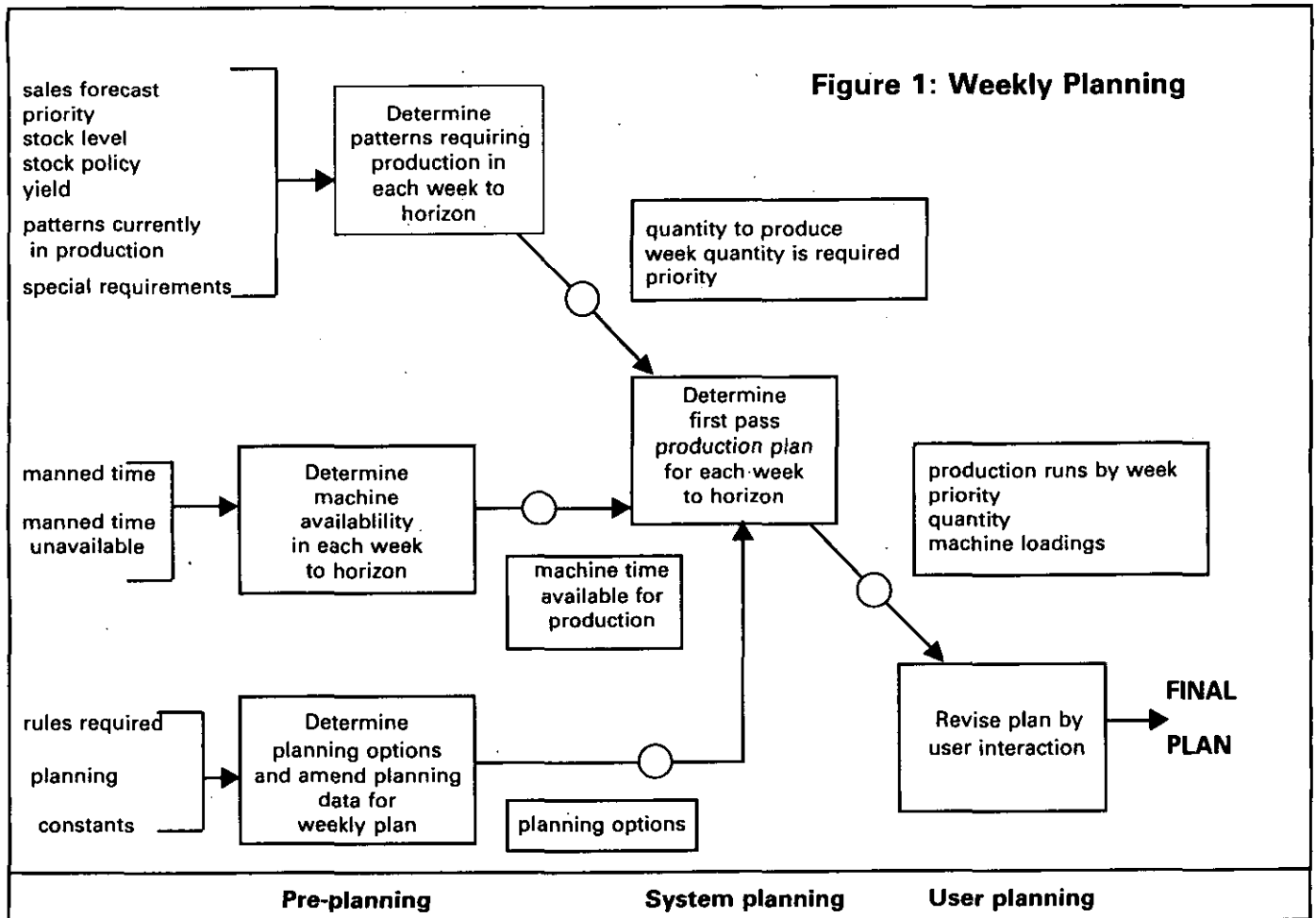
- ii) Weekly Planning goes into the detail of the individual wallcovering patterns and the total weekly capacity of each production machine, looking specifically for bottlenecks.
- iii) Daily Scheduling takes the weekly plan and produces a schedule showing which pattern will be processed on which machine at what time in the week.

The three sub-systems all interact with a common database and aim to aid human decision making, not to replace it. The database, together with the Weekly and Monthly Planning Systems, is implemented on a DEC VAX 8350 (using NATURAL/ADABAS software [7]). Daily Scheduling uses an interactive planning board shell which has a PC version or can be run on the VAX network.

### Weekly Planning

Weekly Planning was chosen to be the first planning level to receive detailed attention as it offered the most immediate benefits in improved stock planning. The prior identification of bottlenecks was impossible under the old system but is central to the new Weekly Planning.

Implementation has been deliberately gradual with time for the planner to become familiar with each step. The first part of the Weekly Planning System has been in use for several months with extremely high levels of user satisfaction. It is again divided into three sections - Pre-Planning, System Planning and User Planning (Figure 1).



**Pre-Planning** assembles and evaluates the various requirements for production of individual patterns based on sales forecasts and stock levels. The planner reviews the suggested requirements and specifies machine availability and planning options.

**System Planning** checks the production requirements against the machine capacity available in order to identify bottlenecks. If bottlenecks are found, planning options (specified in Pre-Planning) are applied to relieve the overload(s). When any overloads have been removed, production requirements are finally allocated to the machines which will process them.

**User Planning** allows the planner to examine the results of Systems Planning via a series of screens. These permit small changes to the plan as it is examined on a machine or product basis.

#### **Daily Scheduling**

In parallel with the later developments in Weekly Planning, a start has been made on Daily Scheduling. The increasing realisation of the need for an acceptable speed for interactive use, coupled with the requirement for a good graphics facility on a colour display, led us to purchase GENETIK, a planning board shell, from Insight International. The database is accessed for the first week's plan from the weekly system and a detailed schedule provided from it.

### **FOURTH GENERATION LANGUAGES (4GLs) AND MANUFACTURING SYSTEMS**

#### **Data Processing and Decision Support**

The tasks required of most commercial computer systems allows the division of most such systems into two broad groups.

**Data Processing Systems (DP):** those which receive and process information on a routine basis, which generally perform relatively simple mathematical calculations but which must maintain accurate and timely records. These are typified by a sales order entry system in which customers can enquire about current stock levels and can place orders.

**Decision Support Systems (DSS):** those which perform much more complex algorithms and are used to manipulate data from existing databases so as to explore the consequences of alternative decisions. A production planning system which takes stock and demand records and investigates feasible schedules would be a good example of a DSS.

Fairly obviously, there is an increasing number of systems which include both large scale data processing and complex decision support. Manufacturing is one such area. A DP system is needed for shop-floor data-capture but production planning requires a DSS.

#### **Evolution of Fourth Generations Systems**

Computer languages have become more and more specialised as they have developed, with DP and DSS evolving along separate paths. A typical third generation approach to a DP problem like Sales Order Processing is to use COBOL or PL/1, while a DSS to model production could be written in FORTRAN or APL. Similarly there are 4GLs designed for DP, like NATURAL, and, for DSS, Executive Information Systems like Comshare's Commander EIS.

4GLs for DP originated from the backlog of new applications facing many departments. With maintenance of existing systems taking up to 80% of data processing staff time, systems professionals realised that something had to be done. These 4GLs were not created by disinterested committees or academics, but by systems professionals. They needed, and therefore created, languages and tools which reduce the overall development and maintenance time for new systems, moving away from commercial programming as a craft towards a systems engineering viewpoint in which a system can be specified and designed like any other artefact.

#### **Claims for 4GLs**

There are three main claims that are associated with 4GLs:

**Reduction of System Development Time.** Less than half the system development time with a 3GL is spent on programming [3,4,8]. A 4GL must be embedded in a development environment to achieve the claimed reduction in development time of at least one order of magnitude.

**Reduction of System Maintenance Time.** With 4GLs, the time spent on maintenance is reduced because their high level and free-format nature makes programmes easier to understand, aiding the identification and implementation of changes. If a Data Dictionary or cross-referencing utility is available then the impact of any changes can be identified. This is a significant advantage over most 3GLs.

**Increased User Involvement in Systems Development.** A 4GL can be used for prototyping systems at the requirements definition and design stages. Prototyping of systems is rapid and encourages interaction between the developer and the user ensuring the developer fully understands the user's requirements. It also encourages user ownership of systems.

### **QUALITIES REQUIRED IN AN IDEAL 4GL**

#### **Language Factors**

Any programming system used to develop large scale system needs to incorporate the following factors, to ensure adequate long term maintenance: Modularity, syntax consistency, variety of data types and language constructs, robustness, library and version control [1,5,6]. In addition a 4GL should allow data to be transferred to and from other systems/packages like Lotus 1-2-3 and DBase. It should also support the calling of executable modules written in a different language if these are needed for speed-critical code and applications where the 4GL is inappropriate. The run-time speed of a 4GL often appears slow because the power of the language tempts the system developer into much more complex operations than would be considered with a 3GL.

The language at the core of most 4G environments is intended to be both expressive and powerful. An expressive language is one whose syntax is close to the problem being tackled by the programmer using the language. A powerful language is one whose syntax allows a large number of low level computing operations to be carried out from a single, simple, high-level instruction. Thus 4GLs are easier to learn if they are expressive and allow rapid programme development if they are powerful.

#### **Environment tools**

The full benefit of using a 4GL can only be gained when complemented by some additional tools. Ideally, but not realistically at present, these tools should cover the entire systems development life-cycle. The tools should also be as integrated as possible such that a change at any 'level' is reflected throughout the system. But beware these tools are still in their infancy. In the last few years a number of tools have emerged claiming to be CASE tools (Computer Aided Software Engineering) and now there is a growing passion for software suppliers to claim ICASE (integrated CASE). Next on the line is IPSEs (Integrated Project Support Environments) which support the control of the project as well as the product.

Some CASE tools claim to support or automate the entire systems development life-cycle. A recent study "CASE Analyst Work-Benches: A Detailed Product Evaluation" (R. Rock-Evans, Ovum) comparing 66 such products concluded that only 10% of the tasks are supported and under 1% automated.

Typical tools making up a 4G development environment are:

**DBMS (Database management system):** Nearly all 4GLs support the accessing of data from a database system. The DBMS may be integrated with the 4GL or provided by a different manufacturer. It is likely that more efficient

database access and consistent interface is provided by an integrated DBMS and 4GL.

**Data dictionary:** A data dictionary is a central store of information about data items such as identification, relationships to other data, ownership, usage, meaning and format. The data dictionary assists in the management of data and systems development.

**Data flow diagram editors:** These editors aid the design stages of systems development. Some 4GLs allow the automatic generation (to varying degrees) of programmes from these design tools.

**Testing tools:** These tools may include programmes to identify areas of codes not executed by test data test case generators and checks on conditions that variables should obey.

**Intelligent programme editors:** Programme editors within 4G environments offer varying levels of facilities and integration with the development environment. These may include such things as automatic identification and checking of syntax. Of primary importance is that an editor should allow the fast manipulation of text such that the developer can work at his or her pace, not the machine's.

**Screen design:** Screen design editors allow the creation of screens by 'painting' a picture of how the screen should be laid out. Screens can be modified easily and consistency in the user interface can be aided by providing a library of standard screens.

**Report generation:** Report generators permit data extraction from database files and data formatting into a report. Default values for such things as page size, length and numbering should be provided.

**Query language:** The provision of a query language allows end-users to access data base files for information using a small set of commands and operators. These query languages are often similar to the high level database statements within 4GLs.

**On-line help:** The help facility provides useful and relevant information about the language and/or environment at any point in time.

## EXPERIENCES OF USING NATURAL/ADABAS

### Programming in the Natural Language

The 4GL, NATURAL has been used over the past eighteen months to write the production planning system described above. The syntax of this very powerful and expressive language has been intentionally chosen to relate closely to the English language. It provides optional syntax for commands and allows relative freedom in programming style. However, danger lurks when a language attempts to be 'all things to all people'. Resulting programmes take on the appearance of different languages when written by different authors. This makes central maintenance of software more difficult, if programming standards are not established by the company.

To enhance the speed of the planning system for both batch and interactive use, it has been necessary to read information from the database in large quantities into main memory using array structures. That is, the 4GL has been used as if it were a procedural 3GL so as to avoid repeated physical database access. Subsequently a high proportion of programme development has moved away from a 4GL programming style and taken on the appearance of an inelegant 3GL.

### The Database Management System - ADABAS

Although ADABAS may be used as a 'relational' database, it does provide additional facilities to structure files in a non-relational format through 'multiple value fields' and 'periodic groups'. These have been found most effective for storing information efficiently when the number of fields needed varies, without losing the benefits of a generally 'relational' structure [2].

Security is only incorporated at the *file* level for controlling the reading and writing of information. This means that where

sensitive data is concerned, it is sometimes necessary to move information to a separate file, adversely affecting access times.

### Data Dictionary

The data dictionary, PREDICT, encourages a top down approach to systems design starting with a conceptual view of information. From the conceptual view, physical database files are formed. Each group of users of the information can then be given a view of the physical file showing only the information of interest. The dictionary's cross-referencing facility has proved extremely useful in maintaining code. Some problems were encountered with data definition standards not being established sufficiently early in the project, probably because the whole company was new to 4GL use.

### Interactive Screen Design

NATURAL provides a facility for designing screens independently from the programmes which use them. This has been very useful as:

- i) It is possible for both the developer and end-user to discuss and design screens together with the aid of a computer tool.
- ii) It becomes easier for a developer to write code once screens have been designed as problems can be visualised.

### Report Generation

When defining a field's characteristics within PREDICT, the designer is also asked to specify default headings for the field, therefore consistent terminology can be maintained in screen and report output.

The ability to produce reports very quickly has aided the acceptance of the planning system. It is now possible to provide further useful information not previously available to other functions within the company.

## GENERAL COMMENTS

Using a 4GL, which is both new to the company and designed for DP work, two O.R./Computing Graduates have written and implemented a fairly complex weekly production planning system (including construction of the required database) in about eighteen months. The task required good co-operation within the company and time for communication to company directors and staff. The first six months were spent on preparing the data and specifying the systems required and the last twelve months using the 4GL to develop and implement the systems.

Change is always difficult to accept especially when it involves the introduction of new computer systems. However, the implementation of the systems proceeded more smoothly than anticipated. The reaction of several key staff was far from favourable at the start of the project but their involvement in the overall development (particularly the design of the user interfaces) has resulted in a very willing acceptance of the system. This has been used weekly for several months with a high degree of satisfaction, discarding the old system completely.

## SOME RECOMMENDATIONS

**Hype:** Do not believe all the hype associated with 4GL development environments. Check that the software is capable of performing the task it will be required to do, if in doubt, try it out!

**Objectives:** Throughout the development of a 4GL system, take time to stand back and ensure that work is being directed towards the objectives.

**Standards:** Ensure that standards are established, communicated and adhered to by everyone. Maintenance becomes difficult with non-standard programmes.

**Design:** Do not use 4GLs as an excuse to miss out the design

stage in developing systems. There is no substitute for good design.

**Prototyping:** Prototyping is an excellent means of experimenting with programming techniques, identifying limitations of a language and communicating with the end-user. However, resist the temptation of endless prototyping with eventual conversion into the final system.

**Documenting and Testing:** As with design, systems must be documented and thoroughly tested.

**Complexity:** Beware of adding unnecessary complexity into a system when using a powerful tool. Major benefits can be achieved by getting the basics right.

**Data Management:** Underlying all 4GL systems is data. Remember that data must be centrally managed and controlled like any other company resource.

**Change:** Changes are inevitable, better systems only evolve as the result of change, but the implications of change must be controlled.

#### ACKNOWLEDGEMENTS

The project described in this paper was carried out under the Teaching Company scheme funded by SERC/DTI. The authors wish to thank David Wray and John McCreesh, from Nairn Kingfisher and Ron Adelson and Tony Dorey, from the University of Lancaster, for all their assistance.

#### REFERENCES

1. Abbey SG. (1984), Cobol dumped, Datamation, January 1984, pp.108-114.
2. Alagic S. (1986), Relational Database Technology, Springer-Verlag; N.Y.
3. Aron JD. (1983), The programme development process PartII : The programming team, Addison Wesley;Reading, Mass.

4. Brooks FP. (1975), The mythical man month, Addison-Wesley;Reading, Mass.
5. Chapin N. (1984), Software maintenance with fourth-generation languages, Software Engineering Notes, January 1984, Vol.9 No.1 pp.41-42.
6. Martin J. (1985), Fourth generation languages - Vol.1, Prentice-Hall Inc., Englewood Cliffs, New Jersey.
7. Martin J. (1985), Fourth generation languages - Vol.II, Prentice-Hall Inc., Englewood Cliffs, New Jersey.
8. Metzger PW., (1981), Managing a programming project, Prentice-Hall Inc., Englewood Cliffs, New Jersey.

#### About the Authors

Ian Cannings and Karen Townrow both graduated from Lancaster University in 1987, with OR/Computer Sciences BSc Hons. degrees. A one year placement provided Ian with the opportunity to work for IBM Software Development Division at Hursley and he has now joined a local management consultancy company, Savant. Karen obtained a one year placement with Inland Revenue, OR Department in Bush House, London and since completing her two year teaching company project has been employed by Nairn Kingfisher Limited to continue her work on systems development.

Jean Tunnicliffe-Wilson, Senior Teaching Company Assistant, has spent ten years with Lancaster University's OR Department, researching both the development of simulation techniques and application of simulation in the area of health care, and now teaches part time for both Engineering and OR at Lancaster. Mike Pidd, also at Lancaster University is a Senior Lecturer in OR and Director of the MBA programme. He has consulted for a wide range of companies, most recently the confectionery manufacture and his specialism in simulation has resulted in the publication of two books.

★ INVENTORY REDUCED 77%

★ WIP REDUCED 67%

★ LEAD TIMES REDUCED 78%

★ SALES INCREASED 50%

★ PRODUCTIVITY INCREASED 100%

TIRED OF HEARING THE SAME OLD THING  
FROM CONSULTANTS WHO ALSO SELL HARDWARE AND SOFTWARE ?

**M.T.A. ARE DIFFERENT - WE GET RESULTS!**

AS AN INDEPENDENT, PROFESSIONAL PRACTICE, WE CAN OFFER OBJECTIVE ADVICE ON THE RIGHT COURSE OF ACTION TO ACHIEVE CLIENTS' NEEDS. OUR STRENGTH IS THE ABILITY TO MAKE IT HAPPEN AND HELP CLIENTS INCREASE THE EFFECTIVENESS OF THEIR OPERATIONS.

FOR INFORMATION ON HOW WE CAN HELP YOU, PLEASE CONTACT:

**MICHAEL TAYLOR ASSOCIATES**

**TEL. (0403) 783925**

*Providing Services to Manufacturing Industry*

*Consulting services/Education/I.T./A.M.T./ Temp. Staff/ Executive Selection*

**MTA**  
MICHAEL TAYLOR  
ASSOCIATES LIMITED

11 GORSELANDS, BILLINGSHURST  
WEST SUSSEX RH14 9TT