A generic approach for developing dedicated database systems for professional applications

L.D.J.F. Nanayakkara and S. Ganesapiragas
Dept. of Mechanical Engineering, University of Moratuwa, Sri Lanka.
A generic approach for developing dedicated database systems for professional applications

L.D.J.F. Nanayakkara and S. Ganesapiragas
Dept. of Mechanical Engineering, University of Moratuwa, Sri Lanka.

Abstract:
There are many benefits to be reaped by organizations of small and medium size by encouraging and exploiting the motivations non-IT engineering professionals possess for programming applications for day-to-day use. Potential shortcomings that are potentially there with such work by non-IT professionals could well be avoided by training them on a generic methodology for rationally developing and testing such systems. This will promote low-cost applications of even somewhat so-called 'technologically less up-to-date' software and hardware versions commonly available in Sri Lanka. The paper discusses specific case studies of resource scheduling using three separate application development platforms and argues that the approach is applicable to many different contexts in industry in production and process planning.

Introduction

The most dynamic changes that are taking place in the world today are found in the field of computer and information technologies. New generations of software and compatible hardware to go with these are being innovated and marketed at an unprecedented rate. Most of the objective users question as to whether this rate of innovation is always driven by general customer needs (technology driven by the demand) or whether it is a technology push by those who are in the computer trade.

Professional users keep wondering limits for freezing their software and hardware applications economically so that the application development and stabilization in industry does not get impaired by the drive for constantly upgrading the computer systems. In Sri Lanka the tendency to upgrade constantly is being aggravated by two factors: one is the offering of the latest hardware at attractive prices and the other the access to pirate copies of expensive software at remarkably low price. It is doubtful as to how long the latter condition could legally prevail in the country. Thus what is most costly in the long-run would be the impairment of the initiative to in-house development of IT-wise modest but professionally advanced and very useful software by non-IT professionals such as engineers. This is an extremely important factor in a country such as Sri Lanka where such contribution to value addition by the non-IT professionals may potentially lead to tremendous improvements in productivity.

In Sri Lanka where 86% of the industries are small and medium enterprises employing less than 250 persons (small employing less than 50), where entrepreneurs find it impossible to keep pace with applications built on fast changing software and hardware technologies. They have three options, each of which has certain disadvantages in the long-run application. One is the installation (either by outright purchase or licensing) of ready-made standard software. These are known to come with many relatively redundant features which make them unnecessarily 'heavy' and contain features which are incompatible with the organizational structure and culture of a particular organization.
The second option is the customization of standard systems which leads to problems with upgrading when necessity arises for using newer versions later on. The third option is to employ IT personnel for developing custom-made systems. Such IT personnel lack professional know-how and intimacy in the particular field of application. This leads to a gap in the communication of the problem at hand which may have professionally advanced and complex decision requirements which may get overlooked for simplicity. Alternatively, when non-IT professionals of the organization get engaged in developing professional support systems (some small and medium industries attempt this) there are advantages due to their sound understanding of the true nature and complexity of the professional problem area, and self-motivation arising from being able to effect continuous improvement of their own work system. There will be increased objectivity in the development exercise, which promotes further professional revision and continuous improvement of systems economically for a long period of time without being forced to quit the old system for unnecessarily faster, and colourful computer gimmicks. Most professionals in engineering and management every now and then venture into productively developing professional applications in-house. In small industries this is partly due to a lack of investment capital for acquiring professional IT applications other than those used mainly for word-processing and AutoCAD.

The authors propose to train non-IT professionals such as engineers to be systematically trained in a generic approach to developing computer modest but professionally advanced software applications. This will eliminate the intrinsic weaknesses such as confusion in data handling and programming, poor back-up procedures, poor presentation and poor user friendliness and deficiencies in robustness due to inadequate testing. These defects have been found in the attempts made so far by non-IT professionals. Affordable and familiar development platforms such as spreadsheets and visual languages, database management systems and common AI languages particularly those in the engineering applications such as (Visual) Auto LISP offer prospects limited only by one’s imagination. The paper takes a step in this direction and presents a common professional application of resource allocation using three different application software development platforms, viz., Excel & Visual Basic, Prolog and AutoLISP. As this programming work is performed by non-IT professionals there will remain the need for the authors themselves to be further trained on a generic approach to professional software development for professional use.

Dedicated Database Management Systems for Resource scheduling

Resource scheduling is a typical problem which arises in many different professional contexts. There are at least two important levels at which the problem is handled. They are long or medium term planning and short term schedule control. Common scheduling contexts include transport fleet scheduling, job-shop scheduling, production line scheduling and personnel scheduling. All these essentially require databases with tools for database management and a dialogue system. Decision models range from the simple search through heuristic rules to full optimization. It is well known that the commonest in many industrial organizations as well as in commercial software systems marketed today belong to the first category. The practice of heuristic decision rules is indirectly applied
in the system by letting a professionally experienced person to operate the scheduling system rather than building in prescriptive optimization algorithms into the system. The scope for the application of AI is clearly seen but less is being done in this direction on a commercial scale. Taking the resource allocation problem as one of heuristic search authors wish to present the prospects for developing professional decision support systems in-house on the following three development platforms using time-tableing data at the University for testing:

**Excel and Visual Basic a development platform**

**Scope:** Excel and Visual Basic reside in the Windows operating system. This enables the programmer to exploit the benefits of object oriented programming (OOP). As OOP's terminology and approach are novel and different from others, some familiarization is required for the non-IT professionals to exploit its elegant ways of handling objects. Keeping in mind the basic fact that any computer programming problem is essentially the same in that it is an exercise to translate an instruction to perform a desired task so that the computer system (that is hardware, compiler or interpreter for the language working in the operating environment) will understand and obediently execute in order to produce the desired output. However, the intrinsic differences of OOP have been understood and made use of in this project.

**Data handling:** The number of time tables (different activity time slots corresponding to various jobs or projects to be done) that could be stored as data is limited by the normal Excel worksheet limitation; this is well above 200. One of the major problems in creating databases on spreadsheets is that the database becomes developer friendly at the expense of being end-user friendly. This is a major obstacle when the system is to be operated routinely by users other than the developer himself. With Excel it is possible to write functions to receive data from any user for storage. Alternatively, Visual Basic (VBA) could be made use of. VBA treats Excel sheet, AutoCAD drawing etc. as objects for manipulation and these are natural partners with Windows. This is an added advantage for professionals who are already familiar with AutoCAD.

**Initialization:** When a VB program runs it comes out with an object whereby by clicking a button an event is initiated. This is conveniently possible because VB programming is mainly event driven. Form-load function in VB is made use of to create the desired object (form) and contains the necessary instructions to initialize various objects in the forms. There are an adequate number of labels (around 6x8x2 = 96) for holding days and times and the contents of the time slots together with class-room (facility) information such as location, capacity etc. Three comb boxes (combos) are used for day of the week, time slot and staff name. From these combos desired selections are made. For initializing using the combo, visual basic opens the excel file, analyzes and fills up a list as required.

**Outputs on screen:** There is a text box to receive a database number, which is incidentally the same as the spreadsheet number. By typing the required number and clicking an event is started and the corresponding output flushes the timetable on to the screen.
**Command buttons:** View next flashes the next sheet. After selecting day and time, for example, from the combo the button will trigger searching and finally come out with the result which is recorded to a text file as well as flashed on to the screen in the region having a list box.

An added advantage of Excel VB combination is that if the program fails, the end user can manage to survive with Excel alone and this is a factor of safety in management. Even though the number of Excel worksheets was limited to 30 in testing, the search time was slow. The user can generally tolerate this slowness considering the usefulness of the application.

The authors intend to supply the program codes in full to those interested as a gesture to induce researchers to investigate and extend the concept to other custom-made applications. Two other development platforms described in the next sections prove how the same functionality is obtained on an entirely different programming environment using implementation algorithms appropriate for the programming capabilities of each environment. This is expected to prove our hypothesis that in developing useful professional applications in industry one need not be too concerned of what to use, but should be more concerned on how to use effectively what one is already familiar with for productive application, rather than be over-concerned about what may be the latest or what better systems could possibly be there unknown to one. The next two cases show the prospects for extending the in-house software efforts in the direction of AI.

**Turbo Prolog as a development platform**

This is a logic based programming language which offers sophisticated matching functions and relational database facilities. Its inference engine uses a backward chaining approach. The disadvantages of built-in inference engine leading to a combinatorial explosion during search can be overcome to some extent by controlling inference by a frame system built on top of Prolog. The basis of Prolog is predicate calculus. A predicate has a name associated with arguments. The arguments can also be variables. There are expert process planning systems written in Prolog.

The salient differences found in developing or translating coding, for the same resource allocation problem which is referred to in this paper, to Prolog environment are as follows:

**Data Handling:** This is done by employing predicates which are prominent elements in logic programming. A predicate with data becomes a clause. Turbo Prolog allows the programmer to place the desired and relevant clauses in text files and Prolog consults these files when it is necessary. Clauses are either added or deleted to modify individual timetables.

One type of clause holds individual names, designation, database number etc. and another type of clause holds time-slot details such as day and time, subject, class room and
database number. Here the database numbers really connect two clauses and form a coherent time-table structure. See appendix 2.

**Searching and Sorting:** Unlike in other languages, there is no need to cast functions to search and sort in an explicit manner. Prolog has power in its special feature of inference mechanism. This is profitably used to perform the task of bringing out selected items such as individual timetables from the database. Even in list processing (AI) languages such as AutoLISP, this task is achieved only by writing an explicit function. However, time-slot search as well as room search have been adequately dealt with in the project in both Prolog as well as LISP platforms. A text file contains search results.

**Data Security:** In Prolog (as well as in LISP) the storage of data takes place as text files and the data security problem remain unattended. This is a management issue because when there is no absolute need for security, there is no need to invest on this.

The authors have been able to use Turbo Prolog version 2.0, which is a ten years old package. This proves our argument that if the software serves the intended purpose there is no need to discard it as obsolete. It was possible to obtain display in the familiar table formats. This is obtained in the screen graphic mode. Nevertheless, it offers the valuable advantage of present-day Windows environment as it could be pasted on Word documents without any distortion. See appendix 1.

**AutoLISP as a development platform**

AutoLISP is a programming language used to write instructions carried out by AutoCAD with which most engineers are familiar. AutoLISP is a subset of LISP Processing language LISP, the most popular Artificial Intelligence (AI) language that has several dialects. There have been attempts to standardize of these dialects under the name COMMON LISP. LISP treats program and the data in the same way. LISP can treat any segment of a program as data, and thus is able to modify it. This enables it to modify knowledge base. The smallest element in LISP is called an atom. An atom can be a variable name, a symbol or a number. A list can be formed by placing a number of items between parentheses. An internally linked structure stores the list. There are many functions available in AutoLISP to manipulate data structure, lists, symbols, vectors and arrays; perform arithmetic operations handle input/outputs, interface with operating system functions and define new functions, help debugging and control information flow. It is an interpreted language, which continuously interacts with the user. It is possible to program dialogue boxes with AutoLISP. Authors are aware that the current AutoCAD versions come with Visual LISP. In addition to its capability to modify program segments, a special feature of LISP is that it keeps track of instructions that have been executed and in what order they have been executed. This explanation facility can be exploited for extending the basic program to an expert system (1).
What is described as basic approach under Prolog is also applicable in many ways here. The following description aims at highlighting the main differences that are prominent in AutoLISP when translating the problem to its environment.

**Data Handling:** List structure was employed to hold individual time tables. These are fed to the computer main memory as variables with sequential database number tied to the variable name in an appropriate manner so that it could be recalled for display, modification and addition of further time slots. See Appendix 3.

**Display of Output:** Display has unlimited scope for presenting information ergonomically to ease the perception by human beings as AutoLISP works with AutoCAD. Each time table was assigned to a distinct layer available in AutoCAD. Viewing individual time tables is effected by just switching to the particular layer pertaining to any required database number. All database numbers are kept in a list and they are flashed on the screen one by one in succession. See Appendix 3&4.

**Data Security:** The data is stored in a manner, which is a peculiar feature of AutoLISP. That is, data as well as the program takes the same form in the list structure. Data also is saved as a LISP file with the same .LSP extension. LISP files are text files. No attempt is made by the authors to introduce security. However, when the situation demands this, the operating system facilities can be made use of profitably for security. Alternatively, confidential coding and decoding (encryption) may also help.

**Modification:** To add and modify timetables, AutoLISP functions are used. This will find a member of a list or associate a particular type of sub-list with a known first item. As the data is added or modified, the text file containing the data is updated by appending the file. The size of the file is reduced by rewriting the file with the data from the memory.

**Program Help:** By running ‘qhp’ (a help command) the user gets information necessary to run the program in the intended manner.

**Practical Observations**

The authors have established by this case study that for non-IT engineering and management professionals it is productively possible to develop user friendly professional applications in programming environments (such as AutoCAD) with which they are already familiar with. It is possible to translate implementation algorithms from one development platform to another without losing user friendliness and functionality. One should not be over concerned about the software used being outdated. The temptation to switch to later versions every now and then should be resisted so long as the old software serves the ultimate purpose. Switching to later versions seems to offer only marginal benefits.

There is a considerable amount of opportunity for developing and continuously improving advanced and dedicated decision making systems for professional use, by
exploiting the so far untapped potential of AutoCAD development languages based on LISP which is an AI language. Prolog is less commonly used in the professional work environment. Visual Basic and Excel are more common in application.

The algorithms implemented in this study are search procedures, which form a major element in AI. The models used in the project have descriptive optimization possibility based on what-if simulation. As timetabling as well as job shop scheduling problems have been historically known as the hardest type of permutations problems, this type of descriptive approach seems to serve the practical purposes satisfactorily and economically. Dr. Thornton and Prof. Du Boulay (2) state that, “Almost all artificial intelligence programs can be said to be doing some form of problem solving whether it be interpreting a visual scene, parsing a sentence or planning a sequence of robot actions. Search is one of the central issues in problem-solving systems.” In this context different AI techniques are seen as based on variant generation based on search.

“It is not uncommon to find that quite large organizations have become dependent on a computer application that has been developed using spreadsheet or database package, on a personal computer, by someone who is not an IT professional. Such systems have rarely been developed for professional (IT) standards; for example, testing is usually inadequate and there are no proper change control procedures.” (3). The authors themselves agree with this statement. Among the many dangers of this ad hoc approach to program development are the following:

- Dependence on one person who is the developer who may eventually leave the company leaving behind a poorly user-friendly program.
- The system may be lost through lack of backup procedures. It is not enough to keep all backup copies stored in the same place even.
- Ill-advised changes may make it impossible to get back to a satisfactorily working version of the system
- Both spreadsheets and DBMS make it easy to confuse program and data, making it error prone.
- The program may be doing the wrong things

The following are some remedies that could be suggested:

- Keep a diary of all changes made.
- Ensure that a colleague understands the system.
- Copy discs at least once a week and take copy home or store in a fire-proof safe.
- When using spreadsheet keep primary data in a separate area and highlight it or surround it in a box.
- Do not build explicit constants, such as GST rate into calculations but store them as explicit primary data.
- Get someone else to test the system.

While this list covers the basics, it is essential for non-IT professionals aspiring to engage in application development to be educated and trained systematically on a sound generic approach to this work.
Two other professional applications which were developed by non-IT professionals, which did not take off ground due to some of the above problems, were in automatic generation of part programs for CNC and machine design using spreadsheets.

Conclusions

Whatever progress made by non-IT professionals in developing and using advanced applications for their day-to-day use has been hampered under certain circumstances and some of these are becoming uncontrollable. As a result many opportunities have been lost.

It has been possible to provide a practical method for developing dedicated database management systems using three different development platforms for solving a time-tabling (resource-allocation) problem. Visual Basic with Excel, Turbo Prolog and AutoLISP were employed.

There is novelty in the work done by the authors in that a new type of management application has been developed in the AutoCAD environment. Training in this type of problem solving using programming makes a professional develop competence in dealing with his day-to-day work, by making use of whatever programming language that is at his disposal.

Even though non-IT professionals may make use of whatever programming platform they are accustomed to in their professional work for developing other decision support systems, training and education is required on a generic methodology for the realization of the true benefits of their work.

Curricula for professional education should be aimed at educating non-IT persons on a generic approach to developing simple software solutions in their professional areas rather than emphasizing too much on the programming skills in a particular language or software even though the latter is also a necessary skill.

Small and medium industries which form 86% of Sri Lanka’s industrial base will immensely benefit from IT developments if the development of low-cost software solutions on any development platform which they may possess is encouraged. Freely available facilities in general spreadsheet and visual programming languages and particularly either the AI or non-AI programming features of AutoLISP (or Visual LISP) resident in AutoCAD used in industry have not yet been fully exploited by the respective professionals.

Bibliography

2. Thornton, Dr. Christopher and du Boulay, Benedict, Artificial Intelligence through Search, First Edition,

Note: Full programming code is available from authors on request. This is gesture to further promote similar developments.
Appendix 1  Turbo Prolog Output

Sample portion of Prolog Data file TT2.DAT

timetable1("Mr.Fernando","Snr2","db7")
timetable1("Mr.Ganesh","SnrETA","db5")
timetable1("Dr.A.Nana","Snr1","db4")

slotdetail("db4","tue09-10","n1t","L002")
slotdetail("db7","tue09-10","n1pp","lop")
slotdetail("db7","tue10-11","n1Mech","L001")
slotdetail("db5","mon08-09","p3Math","r30")
slotdetail("db5","thu09-10","p3Math","r30")
slotdetail("db4","mon08-09","n2Math","L001")
slotdetail("db7","tue09-10","n1pp","lop")
slotdetail("db4","wed09-10","n2phy","L002")

The Prolog program consults this data file.
### Appendix 2  Turbo Prolog Output pasted on Word

<table>
<thead>
<tr>
<th>Time</th>
<th>Monday</th>
<th>Tuesday</th>
<th>Wednesday</th>
<th>Thursday</th>
<th>Friday</th>
<th>Saturday</th>
</tr>
</thead>
<tbody>
<tr>
<td>8-9</td>
<td>n2Math</td>
<td>L001</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9-10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10-11</td>
<td>nMech</td>
<td></td>
<td>L001</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11-12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12-13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3-4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4-5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Name:** Dr. A. Nana  
**Designation:** Sr.1  
**DbNumber:** db1

**Enter to view next:**  
**Database System**  
by S. Gianesapiragas.  
**Turbo Prolog 2.0**

---

### Visual Basic Output

**Time-View**

<table>
<thead>
<tr>
<th>Time</th>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thu</th>
<th>Fri</th>
<th>Sat</th>
</tr>
</thead>
<tbody>
<tr>
<td>08:30</td>
<td>n2Math</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>09:30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10:30</td>
<td>n2Therm</td>
<td>AutoRoom</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11:30</td>
<td>n2Therm</td>
<td>AutoRoom</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:30</td>
<td>13:30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13:30</td>
<td>14:30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14:30</td>
<td>15:30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:30</td>
<td>16:30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16:30</td>
<td>17:30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Appendix 3 AutoLISP Output

Sample portion of Lisp Data file Diime.lsp

(set (read "db14") ')
 (ENUMBER "db14") (NAME "Mr.Kahan" "Lect"))
(set (read "db25") ')
((WED9-9 L:N2:CON L001) (MON9-9 L:N2:TDM) (ENUMBER "db25") (NAME "Mr.ganesh" "eta"))
(set (read "db20") ')
((MON8-9 L:N2:MEC L001) (ENUMBER "db20") (NAME "Mr.Rama" "ETA"))
This shows how data is maintained.
Appendix 3 (ctd.) AutoCAD Drawing Output

<table>
<thead>
<tr>
<th>Time</th>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thurs</th>
<th>Fri</th>
</tr>
</thead>
<tbody>
<tr>
<td>8-9</td>
<td>LNK TDM</td>
<td>LNK TDM</td>
<td>LNK TDM</td>
<td>LNK TDM</td>
<td>LNK TDM</td>
</tr>
<tr>
<td>9-10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10-11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11-12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3-4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4-5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Time Table:**
- Mr. Ganesh

(NAME "Mr.ganesh" "eta") (MON8-9 L:N2:TDM)
(NAME "Mr.Rama" "ETA") (MON8-9 L:N2:MEC L001)
(NAME "Mr.Fernando" "Lect") (MON8-9 L:PZ:TDM 240)
(NAME "Mr.Mann" "Snr.L") (MON8-9 L:N2:MEC L002)
(NAME "Mr.Baba" "Snr.L") FREE
(NAME "Dr.A.Nana" "Snr.L") FREE

Output from Time-slot search
Listing of tslot.dat