Case Study: A Course Advisor Expert System

Ovidiu Noran

Griffith University, Brisbane QLD (Australia)
noran@cit.gu.edu.au
http://www.cit.gu.edu.au/~noran

Abstract. This article presents the development of a knowledge-based expert system for a non-trivial application domain, i.e. selecting a customised postgraduate study program from existing- and targeted sets of skills. Following a customised spiral development paradigm, the paper describes the problem domain, followed by the concepts and requirements involved in the expert system design. The main design constraints are stated and the design decisions are justified in light of the specific task. Attention is paid to the knowledge representation / acquisition and rule base design processes. Implementation and deployment are explained, together with a sample run and result evaluation. Finally, further work is detailed in light of existing features and limitations of the prototype.

Keywords: Constraints, Expert Systems, Knowledge Acquisition, Knowledge Representation.

1 Introduction

The changes brought about by the Information and Communication Technology (ICT) revolution on the human society have also had an impact on the required set of skills of the workforce. While for the recently qualified professionals basic ICT knowledge has been built into the education process, the more mature workforce has to acquire this knowledge separately and in 'backwards compatibility' mode with their existing skills.

Particularly at higher levels of education, individuals may already possess some of the required ICT skills. These abilities may exist in the form of tacit knowledge or formal/informal awareness acquired during past undergraduate study[1] or current professional activity. Typically, this knowledge (although not structured in a consistent way) has the potential to be reused in the process of acquiring and formalizing the necessary ICT skills of the individual. Potential hurdles in the reuse of such existing knowledge are:

- unambiguously eliciting and self-assessing existing knowledge, which assumes the owner's awareness of his/her skills (not always trivial);
- devising a flexible learning process to incorporate the identified skills;
- matching a study program to a particular set of existing- and targeted skills.

[1] e.g. most engineers have undertaken at least one course of structured programming during their undergraduate studies.

T.D. Gedeon and L.C.C. Fung (Eds.): AI 2003, LNAI 2903, pp. 1014–1027, 2003.

This paper illustrates the problem through a case study describing the use of an expert system to customise the necessary study program for particular sets of existing- vs. targeted knowledge. The case study is limited to an expert system *prototype*, dealing with postgraduate students with a non-IT background who wish to acquire ICT formal education in an accredited University environment. For the purpose of this paper, the study program will be called a *Conversion Course*[2]. The system will offer guidance to potential students and to course designers about the necessary structure of a particular study program.

2 The Problem Domain

The existing Web-based University course description system presents a number of limitations. Occasionally, erroneous or stale information is provided, which may lead to misleading students in their enrolments / exams. In addition, the present system does not verify that a student may enrol in subjects in a way contradicting the University policies. Finally, the present subject and course descriptions may be confusing for prospective students[3]. Such limitations could adversely affect the image of the teaching institution.

The advisory expert system is intended to provide preliminary assistance and advice for prospective postgraduates, extracting and inferring the necessary information from a current knowledge base within a consultation session. This will provide a customized, realistic assessment of the alternatives, requirements and expectations for a particular student, and thus help prevent enrolment errors.

In the current teaching system, subjects are considered to be an indivisible entity, non-customizable, with a fixed number of points awarded on completion. Thus, prior knowledge covering part of a subject does not benefit a student, which still has to enrol in the *whole* subject and be awarded *all* the subject points if successful. To enable and support the proposed expert system, the teaching system should allow subject *modularity*, with points allocated to separately assessed course components.

3 Developing the Expert System

In developing the expert system it is desirable to start with a *prototype* of the complete expert system. A prototype can assess the feasibility of a project without full financial or resource commitment and may then be submitted for evaluation to users / stakeholders to obtain their feedback and commitment. User and host institution acceptance is a multidimensional aspect [15], which ultimately decides the usefulness of the entire system development effort. The development

[2] In this paper it is assumed that *subject* equals a single study unit (e.g. 'Programming 2') while *course* is a collection of *subjects* leading to the awarding of a degree (e.g. 'Master of Information Technology').

[3] e.g. compulsory/elective subjects, or a particular subject enrolment order for a course

of the prototype could not be left to the knowledge engineer alone[4], as it needs the knowledge elicitation from at least one domain expert [6] (in this case, a subject convener).

3.1 Life Cycle Models

Several life cycle paradigms have been considered, such as waterfall, incremental, linear, spiral, etc [6, 12, 13]. A modified *spiral* paradigm has been successfully adopted, observing some guidelines in adding new facts and rules, such as: maintain integrity (do not contradict the existing facts and rules), avoid redundancy (do not represent knowledge already existent in the rule base), prevent scattering the knowledge over an excessive amount of rules / facts, etc. Thus, a balance must be struck between the facts' and rules' complexity, expressive power and number[5].

3.2 Concepts of the Expert System

The expert system is based on several concepts aiming to improve the subject selection and education processes. These concepts are *modularity* of subjects[6], *prerequisites* and *outcomes* for subject modules and *credit* for previous studies.

Hence, the aim is to establish *modules* within the subjects, having their own prerequisites, outcomes and credit points awarded on completion. The granularity of (i.e. number of modules within-) a subject must maintain a balance between flexibility and processing / development time required[7].

3.3 User Requirements for the Expert System Prototype

The user will provide a preferred type of occupation (targeted set of skills) and previous knowledge (potentially usable to satisfy some of the prerequisites for the modules composing the study program) as requested by the system. The system will provide a study course to achieve the targeted occupation and may also suggest corrections if the user skills (as stated) are too limited or too high.

3.4 Design of the Expert System Prototype

System Requirements The system requirements represent a translation of the user requirements into the system domain[8]. For this particular case they may take the form:

[4] unless the knowledge engineer was also a domain expert, in which case some method of triangulation should be employed, e.g. member checking.

[5] i.e., a large number of simpler rules vs. fewer, more complex- and expressive rules.

[6] subject modules may be inferred e.g. from its syllabus (which may also assist subject decomposition)

[7] a very small number of modules defeats the purpose of decomposing the subjects, while a large number of modules may require too many resources.

[8] the accuracy of this translation should be subsequently *validated* with the end user.

- the expert system must rely on the user's tacit knowledge[9];
- modules are seen as objects, having interfaces (in fact, its prerequisites and outcomes) contained in special lists, further contained within module facts;
- a module prerequisite may be satisfied by at most one outcome, (either of another module, or declared as 'known' by the user);
- the consultation starts with a set of initial facts, asserted at run-time according to the user's answers to 'job domain' and 'job type' queries. These facts provide the initial list of unsatisfied prerequisites;
- the system attempts to match these unsatisfied prerequisites (first against outcomes declared 'known' by the user, and if unsuccessful, against the 'outcomes' lists of the other module facts in the knowledge base;
- when a matching outcome is found for a prerequisite, the module owning the outcome is marked as 'needed' and its prerequisites list is then in its turn scanned for matches with other outcomes. The prerequisite for which the match has been found is marked 'done';
- any prerequisites found not to be either declared 'known' by the user, or not already satisfied by other modules' outcomes will trigger additional questions for the user;
- according to the user's answers, prerequisites are either marked 'known' (user already has the required skill), or added to the unsatisfied prerequisites list;
- the process is repeated until all prerequisites are satisfied - either by other modules' outcomes ('done') or by previous skills of the user ('known');
- the list of all modules marked as 'needed' (that the user will need to enrol in) is printed out[10]. Note that some special modules (e.g. project) are automatically added, regardless of the user's prior knowledge.

System functionality is shown diagrammatically in Fig. 1.

Knowledge Representation and Method of Inference Design decisions had to be made regarding the formalism to be used in representing the knowledge contained in the teaching system. The type of knowledge to be represented (namely components of the various courses composing current study programs) matched the form of a collection of isolated facts, rather than a structured set of knowledge or a concise theory. Therefore, rules / assertions were preferred to frames or algorithms in a first decision round [16]. Production rules [21] have been chosen in preference to logic and semantic nets / frames [19] for being more suitable for solving design and planning problems [17].

Bottom-up reasoning / inference was decided upon, whereby a starting fact set (provided by the user) is matched against the conditions of the production rules in the rule base (constructed upon the policies governing the University study programs). Bottom-up inference has led to forward chaining as a preferred model of conflict resolution, with possible prioritisation by assigning production rules appropriate weights [17].

[9] i.e. basic knowledge assumed by a prospective postgraduate student (e.g. Maths) - as in [5]. Not to be confused with the 'previous knowledge' stated to the system.

[10] depending on the granularity of the subject decomposition, some 'needed' modules may have 'bonus' outcomes (i.e. which do not satisfy any prerequisites).

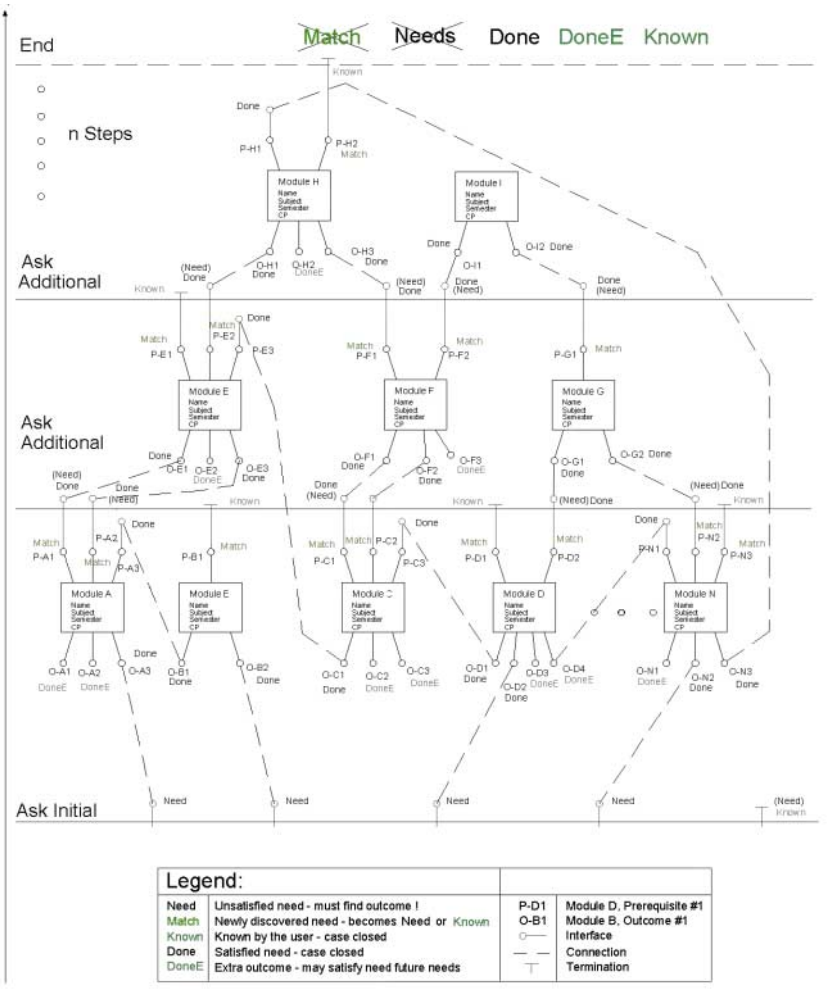

Fig. 1. Expert system prototype functionality

Knowledge Acquisition The knowledge acquisition methods chosen for the prototype have been the questionnaire and structured interview. This decision owes to several factors, such as prototype size, nature of the problem domain and availability of domain experts. Questionnaires are well suited to future automated processing, which benefit the knowledge acquisition process [2]. The questionnaire and interview designs have acknowledged the gap between the descriptions of domain specialists (subject conveners) and the resulting compu-

tational models [4, 18] and the social issues underlying knowledge creation [10][11]. The interview design has loosely followed the COMPASS procedure [23].

Design Constraints Constraints are necessary in order to enable a finite solution to be produced. Examples:
- the outcomes of a module must be distinct;
- two modules may not produce the same outcome: doing so would produce a redundancy in the teaching structure which needs to be resolved[12];
- all the module prerequisites contained in the knowledge base are satisfied by outcomes of other modules in the base[13].
- nil prerequisites for modules are allowed; however, they still require basic graduate knowledge, such as maths, physics, etc (the tacit knowledge previously mentioned);
- cyclic dependencies between any two modules are disallowed (e.g. if module A has a prerequisite satisfied by module B, module B must not have a prerequisite satisfied only by module A). Should this situation occur, the offending modules must be reassessed with regards to their prerequisites / outcomes[14];

Further constraints may be added in the developing the expert system, e.g.:
- maximum number of year **n** modules: may conflict with the concept of a Conversion Course and limit the flexibility of the expert system;
- maximum number of modules per Semester (the prototype violates this constraint). In real life, subjects tend to be equally distributed in all Semesters;
- balanced number of modules in each Semester: see previous.

The Expert System Conceptual Model The knowledge base should contain *facts* and *rules* referring to the prerequisites and outcomes of modules of the subjects offered in the University[15]. The facts are either 'fixed' (such as the modules information) or run-time asserted (e.g. the user's answers to the expert systems' questions). The inference engine must be chosen to match previous requirements and design decisions. The user interface, preferably graphical and integratable with currently and commonly used operating systems and enabling technology infrastructure (e.g. Internet), would preferably be implemented in the same language as the inference engine.

[11] the questionnaire and the interview structure must take into consideration the *culture*(e.g. shared values, beliefs) and policies of the institution hosting the domain experts and the system.

[12] thus this system may be used to identify and eliminate overlapping areas within different subjects

[13] this constraint does not include 'trivial' prerequisites, i.e. basic knowledge that a graduate is expected to have. Also, the constraint may be relaxed by accepting that students satisfy some prerequisites by enrolling outside the teaching institution.

[14] this should also be disallowed in real life, since there is no way for a student to enrol in either of the modules, *unless* he/she has previous skills covering the prerequisites of one of the offending modules.

[15] supplementary information is also provided, such as semester, credit points, parent subject, etc. Additional information may also be introduced (e.g. module convenor).

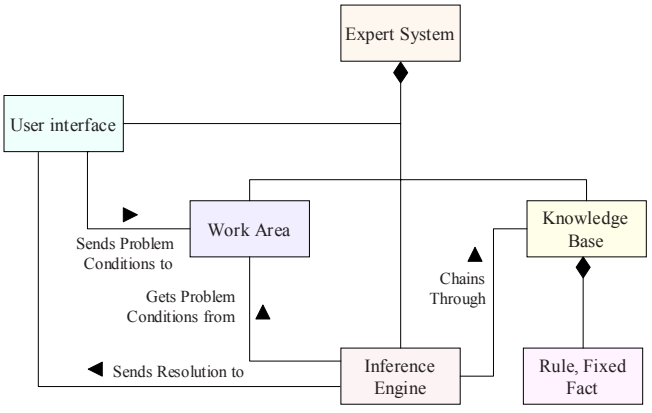

Fig. 2. Object diagram of the expert system (based on [8])

Start::InitialQuery1
ifPart = " job type Database_Admin " thenPart = " ask for Query Optimisation Knowledge & assert the user answer "

Fig. 3. UML rule representation

A Unified Modelling Language (UML, [25]) model of the expert system is presented in Fig. 2. In this figure, the user-expert system interaction occurs through the user interface, which sends the problem conditions (answers to questions) to the work area that holds all the temporary data. The inference engine uses the knowledge base for the rules and fixed facts, and the work area for the dynamic facts (asserted at run-time). The solution is delivered to the user interface. The classes shown in the figure will be further specified in the Implementation section.

The Knowledge Base In an UML representation, rules may be represented as objects, displaying *attributes* and *behaviour* as shown in Fig. 3.

The expressiveness of Fig. 3 may be improved by employing UML extension mechanisms[16], such as presented in Fig. 4. In this version, each rule may also represent the rules that call- and that are called by the rule in question.

3.5 Implementation

The virtual machine hierarchy as described in [10] provides a good guide towards the expert system prototype implementation. Several web-enabled expert system

[16] such custom representations (which in fact create new modelling languages) must be at least expressed by *glossaries* and a consistent *metamodel*, unambiguously describing the structure of the extended modelling language to the intended audience.

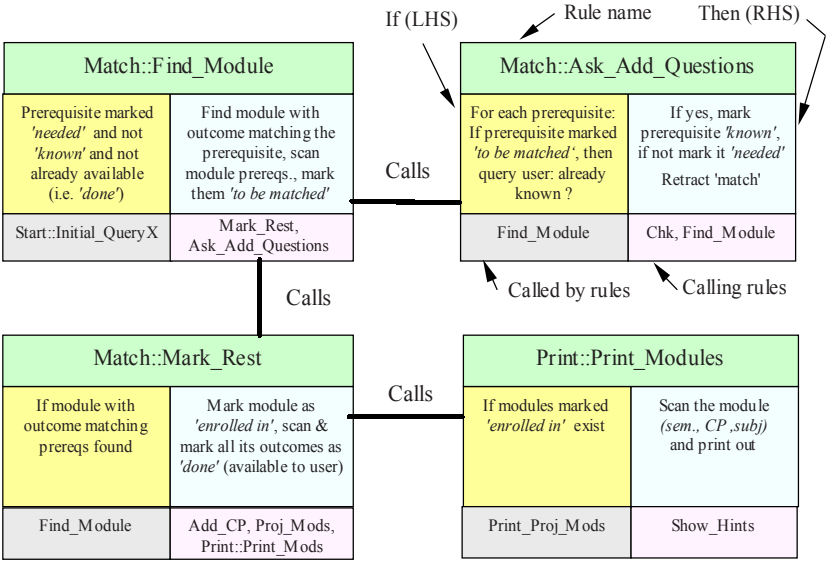

Fig. 4. Possible customised rule representation

shells have been considered for the specific problem domain. The shell has to be matched to the task [14] in order to assist in the knowledge representation exercise. Most shells impose a particular production rule formalism, chaining and structure[17] to the rule set.

Considering the size of the prototype, the resources available and the problem domain, the choice has been an expert system shell written in Java (JESS - The Java Expert System Shell [9]), emulating the CLIPS [11] language, with a simple pre-made graphical user interface. This solution provides the power of the CLIPS language and the flexibility of the Java cross-platform concept. Although the Java AWT (Abstract Window Toolkit) is available in JESS, JConsult [24] has been preferred as an off-the-shelf basic JESS graphical user interface. The inference engine is indirectly provided by CLIPS. The CLIPS language uses forward-chaining and the RETE fast pattern-matching algorithm [7].

Thus, in Fig. 2 the Work Area is the Java applet, the user interface is the applet's window, the inference engine is provided by the Java CLIPS implementation and the knowledge base resides in CLIPS file(s).

The Knowledge Base Implementation JESS, similar to CLIPS (and LISP), uses the *list* construct to hold the data. The facts may be asserted manually (i.e. hardcoded in the knowledge base) or at run-time. They are implemented as lists containing one or more other lists. Example:

(attribute (type job)(value "Database Administrator"))

[17] or lack thereof - refer e.g. EMYCIN [26]

Templates are similar to classes and contain models for the *facts*. Three types of templates have been used: *goal* (a special type of fact, used to initialize the expert system), *attribute* (a general purpose fact consisting of a *type* and a *value*) and *module* (a fact type describing the module information). Example:

```
(deftemplate module ; the module information template, equivalent of a class
    (slot name)
    (slot CP (type INTEGER))
    (slot parent_subject)
    (slot semester (type INTEGER))
    (multislot prerequisites) ; module prerequisites
    (multislot outcomes) ; module outcomes
); end deftemplate
```

A *slot* declares a field within the template - actually, a list in itself. A *multislot* declares a *multifield*, i.e. a list with more than two members. For example:

```
(module ; a particular module - the equivalent of an object
    (name Query_Optimisation_Evaluation)
    (CP 2)
    (parent_subject Database_Management_Systems)
    (semester 1)
    (prerequisites SQL_Data_Definition_Language Data_Storage)
    (outcomes Query_Optimisation Relational_Operators)
); end module
```

This is a *module* object example where the *prerequisites* and *outcomes* list both have more than two members. In this way, a variable number of members may be accomodated within a list without any special requirements.

The rules in the JESS environment take the form *if-part => then-part*:

```
(defrule RuleDB1a
 (attribute (type job)(value "Database Administrator"|"Database Designer"))
  =>
  (printout t "Databases Knowledge." crlf crlf
      "Do you have Database Architectures knowledge ?|explanatory|
      This type of job requires Database Architectures knowledge|yes|no|end")
  (assert (attribute (type Database_Architectures) (value (readline))))
); end RuleDB1a
```

An extensive coverage of the development environment and the user interface is beyond the scope of this document.

3.6 Testing / Verification: Running a Consultation

The system will initially require a target domain and specific employment opportunity (within the chosen domain); this will create the first set of modules needed. The system will then query the user on previous knowledge usable to satisfy the prerequisites of this first set of modules. In addition, the system will seek appropriate modules whose outcomes satisfy the prerequisites in question.

The majority of modules in the knowledge base have non-nil prerequisites. The system will seek to satisfy all the prerequisites of a module before enrolling

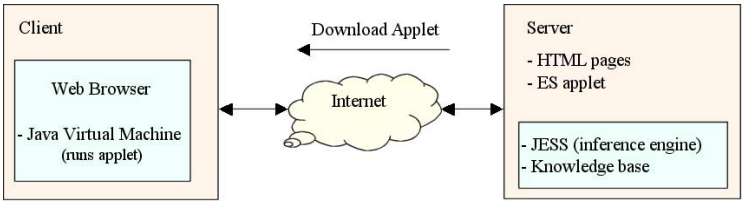

Fig. 5. The Web-based expert system

the user in that module. This process will occur recursively - i.e. a module with **n** prerequisites may require up to **(n-k)** modules (where **k** represents outcomes provided by the user) with outcomes that satisfy those prerequisites. These **(n-k)** module(s) may also have **p** prerequisites that need to be satisfied, and so on. Every time a new prerequisite is discovered, firstly the user is queried whether he/she has the knowledge to satisfy it. If not, a suitable outcome of another module is searched to satisfy the prerequisite. Although this would seem to create a larger pool of prerequisites each time, in fact many prerequisites are satisfied by one module, and per total there can be no unsatisfied prerequisites (meaning that the University caters for all the teaching needs of the student).

At the end of the consultation, the expert system will produce an output containing: Stated (existing) knowledge, Necessary modules with Parent Subject, Semester, Credit Points, Project modules (the Conversion Course must include a 40CP Project) and Total Credit Points necessary for the course[18]. Boundary values are as follows: needed CP $<$ 70 - existing knowledge may be overstated; 70 $<$ CP $<$ 105 - Ok.; 105 $<$ CP $<$ 150 - existing knowledge may have been understated; CP$>$ 150: the envisaged occupation / skills may be unsuitable for that person (previous knowledge too limited).

A sample run of the expert system for the job of *'Artificial Intelligence Researcher'* (with prior knowledge) has produced the output shown partially in Fig. 6. Note that at this stage there is no mechanism to ensure a *balanced* distribution of the modules within the Semesters[19].

3.7 Deployment

Figure 5 shows the current method of deployment of the prototype[20], which was deemed appropriate for the restricted initial scope of the problem domain.

[18] in a customised Course, the number of total credit points depends on the student's prior knowledge.

[19] the algorithm for that kind of function involves careful subject planning and further knowledge elicitation.

[20] at the time of writing, the expert system prototype is available for evaluation on `http://www.cit.gu.edu.au/~noran`

```
********** Consultation Output Started **********

Job targeted: Artificial Intelligence Researcher .
---------------------------------
Prior (existing) Knowledge stated:
---------------------------------
Java_API .
Information_Systems_Concepts .
Entity_Relationship_Model .
Inference_Nets .
Artificial_Intelligence_Trends .
Knowledge_Representation_Principles .
=====================================================
The following Modules are needed:
---------------------------------
Semester I:
-----------
Programming_I_2, subject Programming_II, Sem. 1, 5 CP.
Advanced_Information_Systems_Development, subject Introduction_to_Information_Systems_Development,
Sem. 1, 5 CP.
Programming_Language_Implementation_1, subject Programming_Language_Implementation, Sem. 1, 5 CP.
Programming_Language_Implementation_2, subject Programming_Language_Implementation, Sem. 1, 5 CP.
Introduction_to_Artificial_Intelligence_1, subject Introduction_to_Artificial_Intelligence, Sem. 1, 5 CP.
Programming_I_1, subject Programming_II, Sem. 1, 5 CP.
Natural_Language_Processing_Basics, subject Natural_Language_Processing, Sem.1, 5 CP.
Natural_Language_Processing_2, subject Natural_Language_Processing, Sem.1, 5 CP.
-----------
Semester II:
-----------
-----------
-----------

-----------------------------------------
The following Project Modules are needed:
-----------------------------------------
Artificial_Intelligence_Project_Part_1, subject Artificial_Intelligence_Project, Sem. 1, 20 CP.
Artificial_Intelligence_Project_Part_2, subject Artificial_Intelligence_Project, Sem. 3, 20 CP.
=====================================================
Total Credit Points = 113 .
*****************************************
WARNING: you seem to require over (the required) 100 Credit Points.
---------
-> It is possible that you haven't stated ALL your existing knowledge.
-> You may also examine various courses of study by ASSUMING you had some of the knowledge.
---------
ADVICE: *Restart* and answer 'yes' to all existing / assumed knowledge.

********** End of Consultation Output **********
```

Fig. 6. Extract from the Expert system output

3.8 Maintenance / Modification

Deployment is not the end of the spiral development paradigm. In fact, another spiral, comprising periodic maintenance and updating will be necessary to keep the knowledge base current. All additions / modifications must preserve the currently applicable knowledge base constraints and be properly validated [1]. Specialised maintenance constraints[21] and querying the reasoning of the expert system must also be available to the knowledge engineer.

[21] e.g. a mechanism to check for unsatisfied prerequisites for modules per total (included in the prototype).

4 Further Work on the Prototype and Beyond

Knowledge acquisition has proved to be the major bottleneck in this case study. Thus, the knowledge elicitation techniques will need to be improved, so as to shorten the knowledge acquisition turn-around time (e.g. via on-line questionnaires, and automated assessment and knowledge base input). Interview techniques should be selected to support this automation (e.g. [20]). Low-level automated knowledge acquisition may also be derived from CLIPS [11].

Substantial improvements may be made in the user interface. Also, an algorithm could be implemented to evenly distribute the modules by the semester they are offered in, together with a mechanism to limit the number of modules per semester (or else extend the study period).

The download time of the current expert system implementation could also be shortened by shifting data / input processing on the server side, using servlets (with user-program interaction occurring either via HTML pages containing e.g. FORM requests, or through applet / servlet interaction).

Although the presented prototype is reusable and scalable to a certain degree, a fully featured expert system may have different needs in terms of the set of development and knowledge acquisition tools [3].

5 Conclusions

This paper has presented the application of an expert system to a non-trivial problem domain that involves producing a customised study program for post-graduate students with previous knowledge. The design and implementation decisions have been highlighted and justified, and sample run results have been provided. The development and testing of the prototype have validated the concept of a knowledge-based advisory expert system, provided that a set of essential guidelines are followed and due attention is paid to the knowledge acquisition process design and implementation. Further details on the design and implementation of this system are presented in [22].

Similar to other rule-based expert systems, (notably EMYCIN [26]), the expert system prototype described in this paper may be reused by (1) establishing that the new problem domain suits the type of design decisions taken for the current system (e.g. pattern matching, forward chaining, Web enabling, etc) and (2) replacing the knowledge base with one specific to the new problem domain.

References

[1] Benbasat, I. and Dhaliwal, J. S. A Framework for the validation of knowledge acquisition. Knowledge Acquisition, 1(2), (1989) 215–233 1024

[2] Bergadano, F., Giordana, A. and Saitta, L. Automated vs. Manual Knowledge Acquisition: A Comparison in Real Domain. In H. Motoda and R. Mizoguchi and J. Boose and B. Gaines (Eds.), Knowledge Acquisition for Knowledge-based Systems. Amsterdam, The Netherlands: IOS Press.(1991) 1018

[3] Boose, J. H. A survey of knowledge acquisition techniques and tools. Knowledge Acquisition, 1(1) (1986) 3–37. 1025
[4] Boose, J. H. and Gaines, B. R. (Eds.). Knowledge Acquisition Tools for Expert Systems: Academic Press.(1988) 1019
[5] Collins, H. M., R. H., G. and Draper, R. C. Where's the expertise ? Expert systems as a medium of knowledge transfer. In M. Merry (Ed.), Expert Systems 85: Cambridge University Press.(1985) 1017
[6] Diaper, D. An Organizational Context for Expert System Design. In D. Berry and A. Hart (Eds.), Expert Systems: Human Issues (pp. 214-236). London: Chapman and Hall.(1990) 1016
[7] Forgy, C. RETE: A Fast Algorithm for the Many Pattern / Many Object Pattern Match Problem. Artificial Intelligence, 19 (1985) 17–37. 1021
[8] Fowler, M. and Scott, K. UML Distilled (2^{nd} ed.). Reading, MA: Addison-Wesley.(1999) 1020
[9] Friedman-Hill, E. JESS - The Java Expert System Shell (Report# SAND98-8206). Livermore, CA: DCS, Sandia National Laboratories.(1998) 1021
[10] Gaines, B. and Shaw, M. Foundations of Knowledge Acquisition. In H. Motoda, R. Mizoguchi, J. Boose and B. Gaines (Eds.), Knowledge Acquisition for Knowledge-based Systems. Amsterdam, The Netherlands: IOS Press.(1991) 1019, 1020
[11] Giarratano, J. CLIPS User's Guide (Vol. 1: Rules): NASA.(1992) 1021, 1025
[12] Giarratano, J. and Riley, G. Expert Systems: Principles and Programming. Boston, MA: PWS Publishing Company.(1998) 1016
[13] ISO_JTC1/SC7. ISO/IS 12207: Software Engineering - Life cycle Processes. (2000) 1016
[14] Jackson, P. Introduction to Expert Systems (3^{rd} ed.). Harlow, England: Addison-Wesley.(2000) 1021
[15] Jagodzinski, P., Holmes, S. and Dennis, I. User-Acceptance of a Knowledge-Based System for the Management of Child Abuse Cases. In D. Berry and A. Hart (Eds.), Expert Systems: Human Issues. London: Chapman and Hall.(1990) 1015
[16] Kline, P. and Dolins, S. Designing Expert Systems. N. Y.: John Wiley & Sons.(1989) 1017
[17] Lucas, P. and Van der Gaag, L. Principles of Expert Systems. Wokingham, England: Addison-Wesley.(1991) 1017
[18] Marcus, S. (Ed.). Automating Knowledge Acquisition for Expert Systems. Boston: Kluwer Academic Publishers.(1988) 1019
[19] Minsky, M. A framework for representing knowledge. In P. H. Winston (Ed.), Phychology of Computer Vision. New York: McGraw-Hill.(1975) 1017
[20] Mizoguchi, R., Matsuda, K. and Yasushiro, N. ISAK: Interview System for Acquiring Design Knowledge. In H. Motoda and R. Mizoguchi and J. Boose and B. Gaines (Eds.), Knowledge Acquisition for Knowledge-based Systems. Amsterdam, The Netherlands: IOS Press.(1991) 1025
[21] Newel, A. and Simon, H. A. Human Problem Solving. Englewood Cliffs, NJ: Prentice-Hall.(1972) 1017
[22] Noran, O. A Course Advisor Expert System. School of CIT, Griffith University (`http://www.cit.gu.edu.au/~noran`) (2000) 1025
[23] Prerau, D. S. Developing and Managing Expert Systems. Reading, MA: Addison-Wesley.(1990) 1019
[24] Reichherzer, T. JConsult v1.3. Institute of Human and Machine Cognition, University of West Florida.(2000) 1021
[25] Rumbaugh, J., Jacobson, I. and Booch, G. The Unified Modelling Language Reference Manual. Reading, MA: Addison-Wesley.(1999) 1020

[26] van Melle, W., Scott, A. C., Bennett, J. S. and Peairs, M. The EMYCIN Manual (STAN-CS-81-16): Computer Science Department, Stanford University.(1981) 1021, 1025