A TOOL FOR BUILDING SECOND GENERATION EXHCRT SYSTEMS

Xuejun Tong, Zhijun He and Ruizao Yu

Artificial Intelligence Laboratory
Computer Science and Engineering Department
Zhejiang University, Hangzhou, P.R.China

ABSTRACTS

The demand for second generation expert systems will soon bring forward the demand for tools to build them. ZDEST-2 is built to test the authors' ideas on what characteristics such tools should have and how to reflect them consistently within a system. New concepts such as three-level control scheme and 3-D explanation are introduced;multiple representation methods and inference engines are offered and integrated. The most distinguished characteristics of ZDEST-2 are: (1) It is a general tool suitable for building various domain oriented expert systems(ESs). (2) The ESs built by it can fully reflect the nature of applicated domains. (3) Well-developed environment is provided for the building and using of ESs. (4) A user friendly interface is emphasized, explanations can be tailored to the elaboration needed, and, external procedure accesses are allowed. This paper gives a quite detailed description about how ZDEST-2 gains such characteristics.

I INTRODUCTION

During the past decade, ES technology has been developing rapidly, attracting a great number of researchers and cooperations. But still, it is felt that the first generation ESs neglect the importance of simulating experts' problem-solving trains and the importance of meeting the needs of terminal users. Thus begins the research on second generation ES(SGES). To conform to this tendency, the problem challenging tool designers is how to build tools which are able to assist the procedure of building and using SGESs. Several important characteristics a SGES will have are discussed in C13. Now as to the tools, what are their major characteristics? First of all, in order to have some background knowledge, let us examine what prior tools lack. Prior tools pay little attention to how to reflect the nature of problem domain. Many tools are derived from domain-oriented ESs, it is inevitable that they carry with them more or less features peculiar to application domain. Other tools such as HEARSAY-"1 C63 and AGEC73, tend to be general aids, sacrifice efficiency to gain generality, so they are impractical to use. Prior tools lack the intention to organize separate modules into a closely related tool set to provide life long service for ESs. Domain experts try hard to find tools providing good assistance to the use of ESs, only to find those tools are not suitable for building their own ESs. So we feel, future tools should have the following characteristics: firstly, they should have great flexibility in combining various different inference engines; secondly, ESs built by them should fully reflect the nature of application domain, the expert's reasoning trains and the ways expert giving explanation; thirdly,they should be able to record rich-structured bookkeeping information for knowledge in order to support the maintaining and updating of SGESs and to provide user friendly interface; and forthly, as the SGES may be a sort of distributed problem-solving system, the tools should have the potentiality to build distributed systems. ZDEST-2 is built to explore the potentiality of having these characteristics within a system, in order to offer SGESs life long service. It results in a fair success. It has been used to build four ESs from different domains: one for detecting blood defects (ZCLOT); one for repairing superheterodyne transistor radios (ZRAD); one for prospecting mineral resources and estimating the quantity (ZGPE); and one for making up the best cotton composition for cotton mills (ZCOT).

I OVERVIEW OF ZDEST-2

According to the classification of C23, tools can be divided into several catalogs, and ZDEST-2 can be seen as a general system aid with well-developed supporting facilities. ZDEST-2 has two distinct functions. One is talking with experts interactively to extract domain knowledge and domain problem-solving procedures so as to build domain oriented ESs. This process is called ES building process. The other one is providing various supporting facilities including

explanation based on 3-D truth maintenance structure and meta-knowledge based knowledge base management to assist the using and maintaining of ESs.

During ES building process, by talking with domain experts interactively, ZDEST-2 gains knowledge about the conceptual structure of the domain, and builds upon that ES components such as task, meta-ks and body-ks used by the ES to be built.

Task, meta-ks(MKS), body-ks(BKS) and inference engine are four main ES component types, called ES items. There is a comment frame associated with each item, which records control information and bookkeeping information relevant to the item. Information in the comment frames is useful in scheduling the item or explaining the item's behavior. The comment frame is filled in under the guidance of the domain expert, and so the expert has the power of deciding the actual internal form of the ES under the same three-level control architecture.

In order to gain the problem-solving efficiency and the capability of characterizing application domain and to be generally applicable, we proposed the three-level control scheme.It is a general control architecture composed of planning, sketchy reasoning and concrete reasoning. It is a problem-solving architecture of ES suitable for various application domains. A blackboard acts as the general data communication area, where ES items record and look for needed data.

ZDEST-2 is currently running on VAX-11/785 and Honeywell DPS8/51, using FRANZ LISP and LISP/66 respectively. Its source code is about 9000 lines long (not including the comment lines). ESs built by it can be transferred to run on IBM PC.

III ES BUILDING PROCESS

The ES building process is shown in Fig.l (the single-line arrow indicates data flow, while the double-line arrow indicates control flow).At any time during ES building process, ZDEST-2 is always in one of the following four ES building stages: (1)conceptual structure description, (2) ES component description, (3) knowledge acquisition, (4) ES refinement and assembly. The semantic and syntactic examination is performed from beginning to end for the purpose of eliminating errors occurred in ES building process.

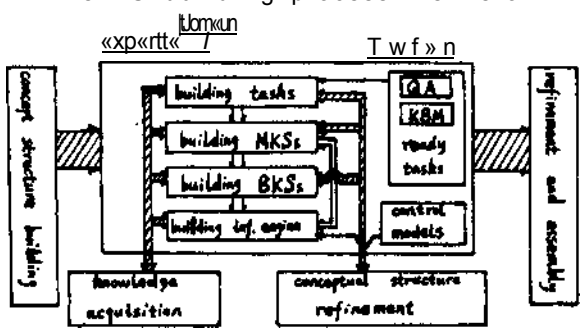


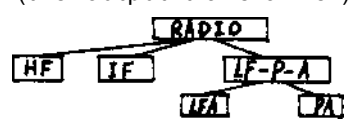
Fig.1. Tht ES bttiUma process

A. Conceptual Structure Description

Conceptual structure is organized around concepts and context types, indicating the property inheritance between context types. Concepts stand for basic objects of the application domain, they are used to define problems, rules, goals and messages mentioned in ES. Context types describe the region within which certain concepts are applicable. Context types themselves can be regarded as concepts which are applicable within certain regions, in other words, they are higher level concepts. For any application domain, the organization of conceptual structure may not be unique, it is influenced by the way experts intend to organize ES components.

The properties of context types and concepts are recorded in the property list associated with them.Through the dialogue, expert enters the properties of each context type and concept.

Part of the conceptual structure of ZRAD is shown in Fig.2, where IF (intermediate frequency), HF (high frequency) and LFA (low frequency amplifier) are context types, that is, they are higher level concepts. The concepts which can be applied in LFA are TR4VC (the voltage of TR4 collector),ITR(the input transformer), OTR (the output transformer) and so on.



F19. 2. *?o#t of tk* c*notft«*l structan j MAP*

B. ES Component Description

Tasks, MKSs, BKSs and Inference engines are ES components, they are combined to form an executable ES. The specification of them is done by filling in the comment frames associated with them and acquiring relevant knowledge pieces used by them. The comment frames have slots for specifying the precondition of task

execution, relation between tasks, the meta rules used by MKS, the peculiar inference engine used by KS, the domain rules used by KS, and so on (Table 1, 2).

Table 1. Comment frame of a BKS of ZCLQT

```
namei bcu
knowledge form: production rules
inference engine: goal-directed-inf
context types: patient
concepts included: finaldef ....
rules: rule-0, rule-7, rule-9
father ks: mcu
```

Table 2. Comment frync of an inf. engine

```
name: goal-directed-inf
glob parameters: (cntxt ddata qua-numb)
cfcombine: cf-combine
cfadjust: nil
usedby: (bcu)
description: This is a goal-directed in-
             ference engine, MYCIN like
             certainty measure is used.
```

C. Knowledge Acquisition

During the ES component description period, the related meta-knowledge and domain knowledge are acquired when mentioned. For the convenience of representation, three kinds of representation methods are offered by ZDEST-2.

1. Production Rules

Production rules can be expressed in simple expression language (SEL) and restricted natural language (RNL).

a. SEL-- SEL uses concepts and their values as operands, predicates as operators to express logical relation between concepts and values. Lisp expressions can be inserted in any place within a SEL expression in case some operations exceed the representation power of SEL. A rule of ZRAD expressed in SEL is show as follows:

```
IF    esc«normal
      40 ma <= ctfc < 70 ma
      type »& Con, off}
THEN  errmsg«cl7t
      (printl MC17 is through")
```

b. RNL-- With the help of predicate description patterns, value function description patterns, acquired properties of concepts and a powerful pattern matcher, the production rules expressed in RNL can be translated without much difficulty.RNL can be extended dynamically by adding new predicate and value function description patterns.

2. Semantic Networks

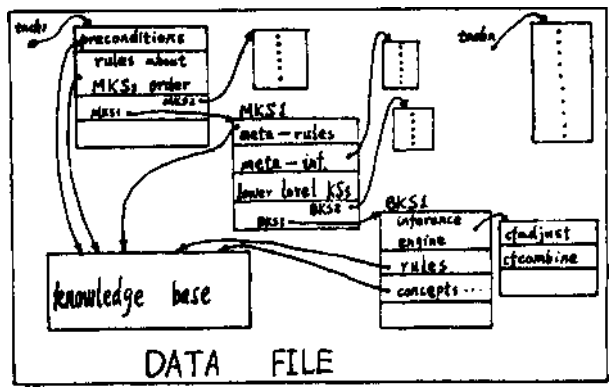
Semantic network representation is also supported by ZDEST-2, a linear partitioned network representation language is provided. The implication relation is represented as a "space"CI03, so in the logical level, the function of implication space is equivalent to that of production rule. This enables the easy transformation between them, makes it possible to allow the co-existence of knowledge represented in different methods. But until now, ZDEST-2 has merely allowed different KSs to use different knowledge representation methods, while knowledge within a particular KS is encoded in the same form.

3. Procedure Representation

ZDEST-2 also allows the knowledge to be represented in procedures written in FRANZ LISP, FORTRAN, PASCAL or C. External access ability is very important to large problem solving activities. By this powerful characteristic, large systems containing tasks such as data processing, table drawing and others can also be built by ZDEST-2.

D. ES Refinement and Assembly

Information and knowledge acquired during conceptual structure description, ES component description and knowledge acquisition are stored in a data file. The data file describes the static organization of an ES loosely. Fig.3 shows the static organization of an example data file.



F*9.3. ES ttwns oryuu^ *in tkt M* jilt*

Because ZDEST-2 is a large tool set, it is composed of a great number of modules. For ES to achieve high performance efficiency, all of the redundant modules and data must be discarded. After that, the newly built system is assembled and compiled into an executable ES. The question answering module and knowledge base management module accompany the ES, in order to support the possible future debugging, updating and maintaining.

Since each KS$_i$ can select its own inference engine and representation method by choice, systems built by ZDEST-2 can be regarded as a combination of a number of the ES cores(by ES core we mean the key components of an ES, that is, knowledge and inference engine), while different cores are different in representation or inference method or others. This distinguished characteristic strengthens the problem-solving capability of ESs, attains great flexibility for ZDEST-2 to build different types of ESs, and also enables the reflection of domain nature.

IV THREE LEVEL CONTROL ARCHITECTURE

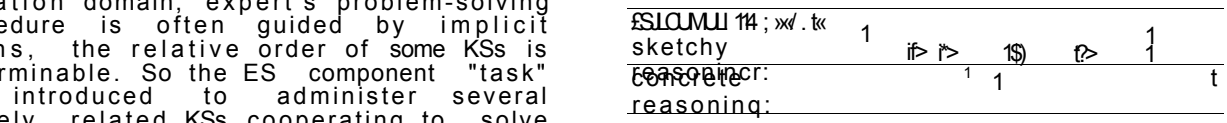
ESs built by ZDEST-2 use three level control scheme, which consists of planning period, sketchy reasoning period and concrete reasoning period, as the problem-solving architecture.

Planning happens on task level. Through dynamic planning, the execution sequence of tasks is dynamically ordered. During planning period, the scheduler controls the execution,suspension of tasks according to the current control information in the blackboard and administers the interaction and message passing between tasks. This process is similar to the scheduling of control KSs and domain KSs by scheduler in blackboard control architectureC33. In blackboard control architecture, the scheduler activates control KSs to determine the importance and necessity of executing certain domain KSs, and then selects one to execute according to the current problem-solving strategy. Such architecture has eight prominent characteristics, but it also has several severe disadvantages: firstly, the time used in scheduling is in proportion to the number of both control KSs and domain KSs; secondly, KSs are relatively independent, isolated components, the execution order of KSs is purely opportunistic. But in application domain, expert's problem-solving procedure is often guided by implicit trains, the relative order of some KSs is determinable. So the ES component "task" is introduced to administer several closely related KSs cooperating to solve certain sub-problem, whose order can be determined when the task runs. The comment frames associated with tasks record a set of production rules used to describe the precondition of task execution or relation between tasks, and a set of production rules to describe the relatively fixed dependency of MKSs administered by the tasks. The function of planning period is to plan dynamically the task execution order in order to reflect the major problem solving steps taking by experts during problem solving.

Each task controls several MKSs. When a task is chosen to run, the sketchy reasoning period starts. The comment frame associated with MKS records the meta rules and meta inference engine. The role of MKS is to make preliminary assessment of the sub-problem based on the clues already occurred on blackboard in order to choose the best suited lower level KSs to run. For instance, it can determine useable rules used by lower level KSs, modify the inference engines used by lower level KSs and enable the execution of lower level KSs. The idea of separating the sketchy reasoning process from the whole problem-solving process is also initiated by the goal of pursuing the similarity of problem-solving between computer and domain experts, because many experts have the intuition of favoring one inference path while discarding others. The "intuition" probably is the result of tradeoff consideration of known clues. The meta knowledge plays various roles. One of the important roles is that it can be used to modify dynamically the comment frames of lower level KSs. Because of this role, when the inference engines used by lower level KSs are so simple that the inference process is nearly determined only by meta rules, the execution of MKS enables the problem-solving process of lower level KSs to be self adaptive to the problem status. We believe the self-adaptive of inference is also an important feature of SGESs.

Concrete reasoning period is responsible for actually solving sub-problems. Since after sketchy reasoning period, everything is tailored to the specific needs, so the concrete reasoning period is quite efficient, especially when the meta knowledge used by MKS is excellent.

The distribution of problem-solving activities at different control level in different time intervals is shown as follows:



where, at time interval:

(l)Choose a task to fire, call it Tl.
(2)0rder the MKSs administered by Tl as Tl-Ml, T1-M2, etc.
(3)Begin the sketchy reasoning of Tl-Ml.
(4)Choose a BKS to run, call it T1-M1-B1.
(5)Preview the rules used by T1-M1-B1.
(6)Adjust inference engine to suit the current state of the blackboard if necessary.
(7)Execute T1-M1-B1.
(8)Repeat (4) to (8), until the sub-problem is solved.
(9)Begin the sketchy reasoning of T1-M2.
(IO)Repeat (1) to (3), choose another

task to fire, until the whole problem is solved.

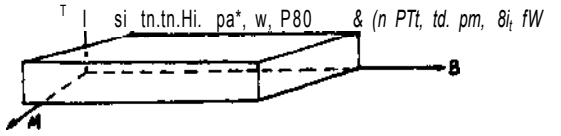
It is not difficult to find out that the three-level control architecture is a general, efficient problem-solving architecture. For example, the moBt often used inference engines: goal-directed and data-directed, can be implemented on the concrete reasoning level; and the more complex mixed-chaining inference engine can be implemented on the sketchy reasoning level. Besides, the symptom based inference engineC43 can be implemented on the planning level. The three level control architecture is an amelioration to the blackboard control architecture. It is more suited to be used in practical systems. Further more, because of the introduction of comment frames, the KSs' structure is loosely organized.Through the fill-in of comment frames, domain experts are free to choose whatever suited inference engines to reflect the exact nature of problem solving.

Because the three level control architecture is a framework capable of characterizing the general problem solving activities of application domain, and because ZDEST-2 provides great flexibility for experts in building an actual system, so ZDEST-2 is a tool worthwhile to use for almost any domain.

V THE SUPPORTING FACILITIES OFFERED

A. Question Answering

The SGES must be able to give appropriate system performance explanations based on user's domain familiarity. ZDEST-2 has made great efforts to provide ESs with a user friendly interface. The ES built by ZDEST-2 encourages user to ask "HOW", "WHY" questions and to demand help from the system during problem-solving process, and also encourages user to use question answering module provided by ZDEST-2 to examine and to understand the evolution of problem-solving process. During the planning, sketchy reasoning and concrete reasoning period, the control transfer information is automatically recorded in a truth maintenance structure (TMS). The TMS is of 3-D structure, indicating control transfer locus between tasks, MKSs and BKSs (Fig.4). According to



Rj. 4. Control twisjer *from Si* to *St.*

user's familiarity, the question can be answered at T, M or B orientation. The locus in any horizontal plane indicates the control transfer within the same task, while the cross plane locus shows control transfer between different tasks. The T level explanation is based on the role of production rules stored in the task's comment frame, it gives the highest level control transfer information. The M level explanation is based on the known clues in the blackboard and the activation record of the production rules stored in the task's comment frame, and it gives the explanation of meta rules' function, indicating the causality concluded by the sketchy reasoning. The B level explanation is based on the activity of each rule, giving the detailed domain rule level explanation. Although the rationale of explanation is based on rule level explanation, the method used is similar to that of C41, but because the explanation could be given at three level, the needs of users with different background can be satisfied. The question types which can be answered by question answering module include the following ones:

- .trace the performance at three level.
- .at T, M or B level investigate the success or fail of rules caused by certain blackboard information.
- .interrogate the conclusions reached at M or B level.
- .investigate how certain conclusions are reached or not reached.

Fig.4 shows the control transfer from SI: task TI, MKS MI, BKS BI to S2: task TI, MKS MI, BKS B2. It is control transfer within same task and MKS. PT1, PM1, PB1 and PB2 point to the sub-explanation structure associated with TI, MI, BI and B2 in place SI, S2 respectively.

B. Knowledge Base Management

ZDEST-2 pays lots attention to knowledge base management in order to give user a we11-developed environment of maintaining ESs. We use meta-knowledge about the semantics of predicates and value functions and meta-knowledge about the logical calculus to check statically the consistency between rules. A knowledge base inquiry language is also provided to assist user browsing through the knowledge base.

VI CONCLUSIONS

From the start of ZDEST-2 project, great emphases have been laid on the four characteristics mentioned above. Now we can say, ZDEST-2 reflects them quite well. The three level control architecture focuses on opportunistic inference at the planning level, focuses on meta level reasoning at the sketchy reasoning level and focuses on solution based inference at the concrete reasoning level. By proper combi-

nation and organization, almost all kinds of application domain can be properly characterized by such three level control architecture. Further more, systems built by ZDEST-2 can reflect the domain nature as much as possible by allowing expert to use particular inference engine and particular representation method to characterize the problem-solving methods peculiar to certain portions of the domain. The maintenance of built systems is also taken cared of by ZDEST-2 since question answering module and knowledge base management module are provided.

Compared with tools built in last several years, a number of these tools' designers recognized the importance of providing two or three kinds of representation language to support different ways of representing domain knowledge C2,93. A number of them also recognized the usefulness of making various typical inference engines available for expert's choice £2,83, but few of them explored the possibility of integrating multiple representation methods and inference engines within an ES. Mostly because of this distinct difference, we believe ZDEST-2 is a second generation tool. On the other hand, tools which are general enough to be applied to a group of different domains, such as HEARSAY-^, AGE, seldom provide well-developed environment for the building and maintaining of ES together. ZDEST-2 can provide 3-D explanation for ESs regardless of whatever inference engines are used as far as the needed control transfer information is properly recorded (this can be easily done, for the inference engines supplied by ZDEST-2 have already had such function, if the inference engines are supplied by domain experts, then it is necessary to embed codes to fill in control transfer information at relevant points of inference).

Several experiments have been made to test the abilities of ZDEST-2. Firstly, we chose EMYCIN system. The CLOT system built by EMYCINC63, was re-built by ZDEST-2, called ZCLOT, ZCLOT acts almost exactly the same as CLOT at concrete reasoning level. The building and debugging of ZCLOT takes merely about three hours. Secondly, we chose the KAS system, since KAS was derived from PROSPECTOR and we have already built an ES to prospect mineral resources, called GPE, which is similar to PROSPECTOR in structure and performance. GPE was again re-built by ZDEST-2, called ZGPE. ZGPE takes only one week's time to built. It uses semantic network and procedure as its major representation methods. It has 128 concepts, 140 knowledge pieces, 11 specific procedures and several context types. Usually it takes about two minutes for ZGPE to get a positive conclusion but less than one minute to get a negative result. So, it is certain that ZDEST-2 is more powerful than EMYCIN and KAS. Thirdly, we tried an ES for multi-factor planning. This is an ES used to find the best composition of different kinds of cotton for cotton mills. In this system, about 22 factors of different importance convoluted make the problem-solving difficult. Much work is done on the planning level and the results are satisfactory. Fourthly, we tried mixed-chaining inference by building an trouble shooting system for radio repair. All of these systems are accomplished successfully.

REFERENCES

1. Luc steels, "Second Generation Expert Systems." FGCS, 1:1 (1985) 213-221.

2. D.A.Waterman, "A Guide to Expert Systems." Addison-Wesley Publishing Company, 1985.

3. B.Hayes-Roth, "A Blackboard Architecture for Control." Artificial Intelligence 26:1 (1985) 251-321.

4. B.G. Buchanan, eds., "Rule-Based Expert Systems." Addison-Wesley Publishing Company, 1984.

5. Willam Van Melle, "A Domain-Independent System that Aids in Constructing Knowledge-Based Consultation Programs." Doctoral Dissertation, Computer Science Department, Stanford University, Rept. No. STAN_CS_80_820.

6. F.Hayes-Roth, eds., "Building Expert Systems.", Addison-Wesley Publishing company, 1984.

7. H. Penny Nil, Nelleke Aiello, "AGE(attempt to GEneralize): A Knowledge-Based Program for Building Knowledge Based Programs.", Proc. IJCAI-79, 1979, 645-652.

8. B.Chandrasekaran, "Generic Tasks in Knowledge-Based Reasoning: High-Level Building Blocks for Expert Systems Design.", IEEE expert 1:3 (1986) 23-30.

9. D. Michie, et.al. "RULEMASTER: A Second-Generation Knowledge-Engineering Facility.", The First Conf. on Artificial Intelligence Applications (1984) 591-597.

10. Xuejun Tong, Zhijun He and Ruizao Yu, "ZDEST-1: A Tool for Building Expert Systems.", ICCC/86 (1986) 51-58.