PROCEEDINGS

Southern African Computer Symposium 1991

6th de Rekenaarsimposium van Suider Afrika 1991

DE OVERBERGER HOTEL, CALEDON

2 - 3 JULY 1991

SPONSORED BY

ISM

FRD

GENMIN

EDITED BY

M H Linck

DEPARTMENT OF COMPUTER SCIENCE • UNIVERSITY OF CAPE TOWN

PROCEEDINGS / KONGRESOPSOMMINGS

6th
SOUTHERN AFRICAN COMPUTER SYMPOSIUM

6de
SUIDELIKE-AFRIKAANSE REKENAARSIMPOSIUM

De Overberger Hotel, Caledon

2 - 3 JULY 1991

SPONSORED by

ISM
FRD
GENMIN

EDITED by

M H LINCK

Department of Computer Science

University of Cape Town

TABLE OF CONTENTS

FOREWORD

The 6th Computer Symposium, organised under the auspices of SAICS, carries on the tradition of providing an opportunity for the South African scientific computing community to present research material to their peers.

It was heartening that 31 papers were offered for consideration. As before all these papers were refereed. Thereafter a selection committee chose 21 for presentation at the Symposium.

Several new dimensions are present in the 1991 symposium:

* The Symposium has been arranged for the day immediately after the SACLA conference.

* It is being run over only 1 day in contrast to the 2-3 days of previous symposia.

* I believe that it is first time that a Symposium has been held outside of the Transvaal.

* Over 85 people will be attending. Nearly all will have attended both events.

* A Sponsorship package for both SACLA and the Research Symposium was obtained. (This led to reduced hotel costs compared to previous symposia)

A major expense is the production of the Proceedings of the Symposium. To ensure financial soundness authors have had to pay the page charge of R20 per page.

A thought for the future would be consideration of a poster session at the Symposium. This could provide an alternative approach to presenting ideas or work.

I would sincerely hope that the twinning of SACLA and the Research Symposium is considered successful enough for this combination survive. As to whether a Research Symposium should be run each year after SACLA, or only every second year, is a matter of need and taste.

A challenge for the future is to encourage an even greater number of MSc & PhD students to attend the Symposium. Unlike this year, I would recommend that they be accommodated at the same cost as everyone else. Only if it is financially necessary should the sponsored number of students be limited.

I would like to thank the other members of the organising committee and my colleagues at UCT for all the help that they have given me. A special word of thanks goes to Prof. Pieter Kritzinger who has provided me with invaluable help and ideas throughout the organisation of this 6th Research Symposium.

M H Linck
Symposium Chairman

SYMPOSIUM CHAIRMAN

M H Linck, University of Cape Town

ORGANISING COMMITTEE

D Kourie, Pretoria University.

P S Kritzinger, University of Cape Town.

M H Linck, University of Capè Town.

SPONSORS

ISM

GENMIN

FRD

LIST OF REFEREES FOR 6th RESEARCH SYMPOSIUM

NAME	INSTITUTION
Barnard, E	Pretoria
Becker, Ronnie	UCT
Berman S	UCT
Bishop, Judy	Wits
Berman, Sonia	UCT
Brink, Chris	UCT
Bodde, Ryn	Networks Systems
Bornman, Chris	UNISA
Bruwer, Piet	UOFS
Cherenack, Paul	UCT
Cook Donald	UCT
de Jaeger, Gerhard	UCT
de Villiers, Pieter	Stellenbosch
Ehlers, Elize	RAU
Eloff, Jan	RAU
Finnie, Gavin	Natal
Gaynor, N	AECI
Hutchinson, Andrew	UCT
Jourdan, D	Pretoria
Kourie Derrick	Pretoria
Kritzinger, Pieter	UCT
Krzesinski, Tony	Stellenbosch
Laing, Doug	ISM
Labuschagne, Willem	UNISA
Levy, Dave	Natal

MacGregor, Ken	UCT
Machanick, Philip	Wits
Mattison Keith	UCT
Messerschmidt, Hans	UOFS
Mutch, Laurie	Shell
Neishlos, N	Wits
Oosthuizen, Deon	Pretoria
Peters Joseph	Simon Fraser
Ram, V	Natal, Pmb.
Postma, Stef	Natal, Pmb
Rennhackkamp, Martin	Stellenbosch
Shochot, John	Wits
Silverberg, Roger	Council for Mineral Technology
Smit, Riël	UCT
Smith, Dereck	UCT
Terry, Pat	Rhodes
van den Heever, Roelf	UP
van Zijl, Lynette	Stellenbosch
Venter, Herman	Fort Hare
Victor, Herna	Stellenbosch
von Solms, Basie	RAU
Wagenaar, M	UPE
Wentworth, Peter	Rhodes
Wheeler, Graham	UCT
Wood, Peter	UCT

6TH RESEARCH SYMPOSIUM - 1991

FINAL PROGRAM

TUESDAY 2nd July 1991

10h00 - 13h00 **Registration**

13h00 - 13h50 **PUB LUNCH**

14h00 - 15h30 SESSION 1A

Venue: Hassner

Chairman: Prof Basie von Solms

14h00 - 14h30
"A value can belong to many types. "
B H Venter, University of Fort Hare

14h30 - 15h00
"A Transputer Based Embedded Controller Development System"
M R Webster, R G Harley, D C Levy & D R Woodward, University of Natal

15h00 - 15h30
"Improving a Control and Sequencing Language"
G Smit and C Fair, University of Cape Town

SESSION 1B

Venue: Hassner C

Chairman:Prof Roelf v d Heever

14h00 - 14h30
"Design of an Object Orientated Framework for Optimistic Parallel Simulation on Shared-Memory Computers" P Machanick, University of Witwatersrand

14h30 - 15h00
"Using Statecharts to Design and Specify the GMA Direct-Manipulation User Interface" L van Zijl & D Mitton, University of Stellenbosch

15h00 - 15h30
"Product Form Solutions for Multiserver Centres with Heirarchical Classes of Customers" A Krzesinski, University of Stellenbosch and R Schassberger, Technische Universität Braunschweig

15h30 - 16h00 **TEA**

16h00 - 17h30 SESSION 2A

Venue: Hassner

Chairman: Prof Derrick Kourie

16h00 - 16h30
"A Reusable Kernel for the Development of Control Software" W Fouché and P de Villiers, University of Stellenbosch

16h30 - 17h00
"An Implementation of Linda Tuple Space under the Helios Operating System" P G Clayton, E P Wentworth, G C Wells and F de-Heer-Menlah, Rhodes University

17h00 - 17h30
"The Design and Analysis of Distributed Virtual Memory Consistency Protocols in an Object Orientated Operating System" K MacGregor, University of Cape Town & R Campbell, University of Illinois at Urbana-Champaign

19h30 **PRE-DINNER DRINKS**

20h00 **GALA CAPE DINNER**
(Men: Jackets & ties)

WEDNESDAY 3rd July 1991

7h00 - 8h15 BREAKFAST

8h15 - 9h45 SESSION 3A

Venue: Hassner

Chairman: Assoc Prof P Wood

8h15 - 8h45
"Concurrency Control Mechanisms for Multidatabase Systems" A Deacon, University of Stellenbosch

8h45 - 9h15
"Extending Local Recovery Techniques for Distributed Databases" H L Victor & M H Rennhackkamp, University of Stellenbosch

9h15 - 9h45
"Analysing Routing Strategies in Sporadic Networks" S Melville, University of Natal

SESSION 3B

Venue: Hassner C

Chairman: Prof G Finnie

8h15 - 8h45
The Design of a Speech Synthesis System for Afrikaans" M J Wagener, University of Port Elizabeth

8h45 - 9h15
"Expert Systems for Management Control: A Multiexpert Architecture" V Ram, University of Natal

9h15 - 9h45
"Integrating Simularity-Based and Explanation-Based Learning" G D Oosthuizen and C Avenant, University of Pretoria

9h45 - 10h15 TEA

10h15 - 11h00 SESSION 4

Venue: Hassner

Chairman: Prof P S Kritzinger
Invited paper: E Coffman

11h00 - 11h10 BREAK

11h10 - 12h40 SESSION 5A

Venue: Hassner

Chairman: Prof C Bornman

11h10 - 11h40
"Efficient Evaluation of Regular Path Programs"
P Wood, University of Cape Town

11h40 - 12h10
"Object Orientation in Relational Databases"
M Rennhackkamp, University of Stellenbosch

12h10 - 12h40
"Building a secure database using self-protecting objects" M Olivier and S H von Solms, Rand Afrikaans University

SESSION 5B

Venue: Hassner C

Chairman: Prof A Krzesinski

11h10 - 11h40
"Modelling the Algebra of Weakest Preconditions"
C Brink & I Rewitsky, University of Cape Town

11h40 - 12h10
"A Model Checker for Transition Systems"
P de Villiers, University of Stellenbosch

12h10 - 12h40
"A New Algorithm for Finding an Upper Bound of the Genus of a Graph"
D I Carson and O R Oellermann, University of Natal

12h45-12h55 GENERAL MEETING of RESEARCH SYMPOSIUM ATTENDEES

Venue: Hassner

Chairman: Dr M H Linck

13h00 - 14h00 LUNCH

FINIS 6th COMPUTER SYMPOSIUM

PAPERS

of the

6TH RESEARCH SYMPOSIUM

The Design of a Speech Synthesis System for Afrikaans

M.J.Wagener
Department of Computer Science, University of Port Elizabeth,
PO Box 1600, Port Elizabeth, 6000

ABSTRACT

The design of a speech synthesis system for the generation of speech from text is presented. The system accepts any Afrikaans sentence and then derives segmental and suprasegmental information through various linguistic processes. The resultant information is used to generate control parameters for a formant synthesizer.

Relevant features of the most well-known experimental speech synthesis systems are given, followed by a specification of the design objectives of the system. The system design is explained in terms of data modelling and data flow analysis. Further explanation is provided for the implementation of the data structure, the data structure manager and a typical linguistic process.

1 Introduction

The synthesis of speech by a computer can be done in different ways, ranging from the simple reproduction of previously recorded speech to the synthesis of speech from text. Methods based on prerecorded speech are easy to implement and provide high quality sound but are restrictive in terms of what speech can be reproduced. Speech synthesis from text on the other hand, requires complex implementation algorithms with a lower speech quality but is completely flexible in its application. The system presented here falls in the latter category. It takes unrestricted text and uses linguistic knowledge to synthesize the speech. The process consists of two main phases;

- the translation of text to a phonetic representation and
- the generation of a speech signal from the phonetic representation.

2 Overview of speech synthesis systems

The earliest speech synthesis systems concentrated on the production of a speech signal from a given phonetic representation. Various types of synthesizers were developed which can be used to generate phonetic sounds. These synthesizers fall into two broad classes namely articulatory and formant synthesizers. The operation of articulatory synthesizers is based on the physical human articulation movements whereas formant synthesizers attempt to reproduce the acoustic signal associated with speech. This system and most other experimental speech synthesis systems make use of a formant synthesizer.

During the 1970's research was directed towards the automatic translation of text to a phonetic representation. In order to derive sufficient information from text, linguistic processes like syllabification and stress assignment must be applied. Carlson and Granström [Car76] developed the first system which separates the linguistic knowledge from the logic of the system. They developed a special programming language that allows the specification of linguistic knowledge in terms of rules. Other systems like the Klattalk system developed by D.H. Klatt [Kla82], also followed this approach but went further in modularizing the system in terms of separate linguistic processes. This can also be seen in the SRS system developed by S.R. Hertz [Her82]. The SRS system uses three different rule sets to perform specific parts of the translation process.

A major shortcoming of these systems is that they all operate on a linear data structure which is semantically overloaded. This has a direct and restrictive effect on the expressive power of the linguistic rules. A further shortcoming is that

the systems are geared for the synthesis of a specific language and fall short as general synthesis environments.

These two shortcomings were addressed in the development of the Delta system [Her85]. The Delta system is hailed as the ultimate synthesis system. It uses a delta (an hierarchically interconnected structure) as the central data structure and a powerful programming language to manipulate it. The user is required to do all data typing and structuring and manipulate the delta structure by following pointers and coding loops and procedures. The expertise required by a user is certainly on par with that of a third generation programmer. The Delta system thus addresses relevant issues concerned with the engineering of a speech synthesis system, but in trying to be a general speech synthesis environment, it has turned out to be another specialized third generation language.

3 Design objectives

The above discussion of existing speech synthesis systems gives an overview of their historic development and identifies the major design issues that were addressed in these systems. With this overview in mind, the following objectives were identified as important in the development of a speech synthesis system for Afrikaans. The objectives are given with reference to the existing systems.

- To provide a framework whereby the linguistic theory for the generation of Afrikaans speech can be practically implemented and tested. The emphasis is on designing a speech synthesis environment specifically for Afrikaans. It differs in this aspect from the Delta system which caters for different languages. Although the system is aimed at a restricted application domain, it must still be general and flexible enough to be used as an experimental environment for synthesising Afrikaans speech.

- To provide a system in which the linguistic knowledge is clearly separated from the logic of the system. This objective was already achieved by Carlson and Granström and also in the later systems, but the Delta system has gone backwards by combining the specification of linguistic knowledge and the logic of traversing the data structure.

- To provide a friendly and familiar interface to the linguistic rule writer. The system is aimed at linguists with none or very little programming experience and must thus provide an interface that suits the expertise and needs of this type of user. The Delta system requires knowledge of programming concepts like data typing, data structuring, loops and modularization, which makes it unsuitable for use by pure linguists.

- To employ an internal data structure that closely models the real world and provides manipulative power. The use of a linear data structure in the earlier systems like the Klattalk and SRS systems, was one of the major drawbacks of these systems. This was addressed in the Delta system by introducing the so-called delta. The critism against the Delta system is that it does not provide any abstract data model based on the delta, but leaves it to the rule writer to do the semantic data modelling. This level of expertise is not expected from rule writers using this system.

- To provide a functional decomposition of the system that is organized into levels of abstraction and semantics with clear interfaces between these levels. This objective is to a certain extent already a solution to some of the problems raised earlier and fundamental to the design of modern software systems.

Throughout this paper, special emphasis is placed on explaining how these objectives influenced the design of the synthesis system.

4 Designing the data structure

It was specified that the internal data structure used in the system should represent the user's perspective of the data. A top-down approach is followed whereby firstly the user's view of the data is determined by an abstract data modelling

technique. The abstract data model is then implemented in a physical data structure.

The sentence is taken as the synthesis unit because larger units like paragraphs, do not contain more information that is derivable by the system. The structural components of a sentence which are of interest for this study are words, syllables and sound segments. The sound segments further consists of parameter frames which contain the parameter values for the formant synthesizer. An abstract data model that represents all the required data elements and the relationships between them, is constructed by using entity-relationship modelling [How83]. The entity-relationship model is shown graphically in an entity-relationship diagram in figure 1.

All the structural components of a sentence namely words, syllables, sound segments and parameter frames are all represented in the model by the SEGMENT entity. This is possible because they are all described by the same attributes. Consequences of this generalisation are firstly that the model is very simple and therefore easy to manipulate and secondly that the model is flexible in that other types of components, e.g. clauses and morphemes, could also be accommodated in the SEGMENT entity. Each occurence of the SEGMENT entity can be associated with certain features. All the features of interest to the system are combined into one entity, called the FEATURE entity. This again puts no restrictions on the different types of features that can be accommodated.

The relationship CONSISTS-OF represents the internal structure of the data e.g. one word consists of many syllables. The HAS relationship shows that a segment can have many features but also that it has one value for that associated feature.

5 Data flow analysis

Figure 2 shows a data flow context diagram of the system. The rule writer provides linguistic rules which are used to generate control parameters for a synthesiser from a sentence entered by the user.

The flow of data inside the synthesis system is shown in Figure 3. All processes operate on the data structure through the data structure manager. The data structure is initialized with information from the input text, expanded by various participating linguistic processes and finally used to generate control parameters for the synthesizer.

The processes are modularized according to their linguistic function in the system. Processes 2, 3 and 4 work together to generate the segmental information for the speech signal. Segmental information applies to individual sound segments. The function and existence of these processes are justified by the theory of natural generative phonology [Com87]. The transformation of a phonological sound segment to a phonetic sound segment as depicted in the design, stands central to this theory. Processes 5, 6 and 7 generate suprasegmental or prosodic information. Prosodic information has a global effect on speech and applies to larger segments like sentences and words.

The different processes in the design are now explained in more detail.

5.1 Text normalization

This process serves as a filter of the text to convert the input to a standard form, e.g. expand abbreviations and numerals like "i.p.v." to "in plaas van" and "1" to "een". The standard form accepted by the system consists of only words and punctuation symbols combined into sentences. Any other special characters which are not handled in the normalization rules will be filtered out. The user can thus customize this module to suit his own needs.

5.2 Generating sound segments

This process takes as input the ortographic characters from the data structure and generates corresponding phonological representations. It also assigns initial feature information to the phonological sound segments. The process closely corresponds with what is known as letter-to-sound rules in other systems [Wag87], except that the output is an abstract phonological sound segment on which further phonological rules must be applied before a phonetic representation is acquired.

5.3 Syllabification

This process groups the sound segments into syllables and assigns feature information to each syllable. A syllable typically consists of vowel which is optionally surrounded by consonants. Since phonological rules can change the syllable boundaries, the syllabification rules must be applied again after the phonological rules.

5.4 Application of phonological rules

It is the task of this process to convert the underlying phonological representations to phonetic representations. This is achieved by different types of phonological rules that insert, delete and replace sound segments and also expand the sound segment feature information. It uses as input the existing sound segments and syllables as well as the existing feature information of these entities.

5.5 Partial syntax analysis

The current aim of the partial syntax analysis of a sentence is to determine the syntactic role of each word in the sentence. Since a natural language parser is beyond the scope of this study, this is achieved by maintaining a dictionary of words with their parts of speech. The dictionary is searched for each word in the sentence and the part of speech is added to the feature information of the word.

5.6 Sentence stress assignment

The syntactic role of words and punctuation symbols are used to stress certain words in the sentence and insert different pauses in appropriate places in the sentence.

5.7 Word stress assignment

This process uses the syllabic information in the data structure to apply a stress pattern within a word. Syllables within a word are given contrasting degrees of stress. Word stress and sentence stress information together is used in the next process to generate fundamental frequency, duration and amplitude parameter values.

5.8 Parameter generation and interpolation

The function of this process is to use the segmental and suprasegmental information generated by the other linguistic processes and determine control parameters for the synthesizer. Parameter values of neighbouring sound segments are interpolated to provide smooth transitions. The parameters are targeted at a software formant synthesizer [Kla80] that is used to construct a digital representation of the speech signal.

6 Implementing the data structure

The physical implementation of the abstract data model has a file for the SEGMENT entity and a file representing the HAS relationship between SEGMENT and FEATURE. Each segment is represented by a record in the SEGMENT file and each feature of a specific segment is stored as one record in the SF file. (Note that these files can also be seen as a sequence of records stored in memory.) The relationships between these records (eg. one segment has many features) are implemented by keeping pointer fields in the records. The record formats of the two files are:

```
SEGMENT(information,parent_segment,
        first_segment,last_segment,
        left_segment,right_segment,
        first_sf)

SF(name,value,parent_segment,
   right_sf)
```

In the SEGMENT file the number field is represented by the physical record number and is therefore excluded from the record description. The CONSISTS-OF relationship is implemented by three types of pointers; a bidirectional child link, a child-parent link and a parent-to-first-and-last-child link. The bidirectional child link connects segments which are all part of the same larger segment together and allows traversal of

these segments in both directions. The child-parent link connects each segment to the segment in which it is contained, and the other two pointers connect a segment with the first and last segments of its list of subsegments. The relationship between segments and features is implemented by a pointer chain connecting firstly a segment to its first feature (the first_sf field in SEGMENT) and then each feature with the next (the right_sf field in SF). Each feature is also connected to its parent segment by the parent_segment field in SF. An example of part of the physical data structure for the sentence "Ek eet lekker" is given in figure 4. A feature list is shown for the word "ek".

7 Data structure manager

The data structure manager plays a very important role as interface between the data structure and the rest of the system. It provides an abstract view of the physical data structures and hides any physical changes that might occur in the data structures, from the rest of the system.

The abstract view provided by the data structure manager can be described as a hierarchical structure of segments in which each segment can have a number of features. The interface is formally defined in terms of the data structure operations provided by the data structure manager. The abstract data structure and defined operations closely follows the approach followed in traditional network database management systems as proposed by the Data Base Task Group [Dat86]. It hides all the intricacies of pointer management from other components in the system and provides complete and powerful operations on the abstract data structure.

Each segment and feature has a pointer or address associated with it. The interface allows access to these pointers which can then be used to navigate through the data structure by means of the operations. The example shown below serves as an illustration of typical operations defined in the data structure manager. In this example the feature "byw 1" is added to the last word in the sentence in figure 4. Assume access to the address of the sentence segment is obtained prior to this code and stored in PARENT. CHILD is another pointer variable and FOUND a boolean variable.

```
FOUND := find_first_segm(CHILD,PARENT)
while FOUND do
  FOUND := find_right_segm(CHILD,CHILD)
insert_first_feat(CHILD,'byw',1)
```

The data structure manager provides the other components in the system with data independence. If any changes in the physical storage of the data are made, it would not affect the other components. These changes are absorbed by the interface. It should also be noted that the interface does not enforce the semantics of speech synthesis but provides a framework that is ideal for such an implementation. The actual structuring of a sentence into words, syllables, sound segments and parameter frames is semantics enforced by other higher level components. Thus the data structure manager in turn is protected from changes in these semantics.

8 A typical linguistic process

The internal decomposition of each of the linguistic processes shown in Figure 3 is the same. It consists of a rule compiler and an inference engine. The interdependence of these components is shown in figure 5. The user enters linguistic rules into an ASCII file using a text editor. The rules are analysed for syntactic errors and then compiled into object code (which is Pascal). The inference engine uses the output from the rule compiler and manipulates the data structure via the data structure manager.

Further attention is firstly given to the format of the linguistic rules which will give an indication of the interface provided to the rule writer, and secondly to the operation and reasons for using an inference engine.

8.1 Rule format

The format of the linguistic rules is consistent with the format of phonological rules used in the linguistic literature [Com87].

A rule has a target level in the hierarchical data structure associated with it. The target level depends on the linguistic process in which the rule is used. The phonological rules for instance will operate on the sound segments. The general function of a rule is to identify a specific segment in the data structure and then to modify the information and features of the segment. A segment is identified by specifying the information content and/or feature information of the segment. Depending on the specific linguistic process, the changes specified by the rule is applied to the same or next lower level of the hierarchy. A change can be an insertion, deletion or update of the current segment information. A segment can also be identified by specifying a context in which it must appear. The context can include one or more segments to the left and/or right of the target segment. Matching any segment in the rule can also be preceded and/or followed by one or more statements. These statements are primarily used to assign the result of Pascal expressions to predefined string, real and integer variables.

8.2 Inference engine

Each of the different linguistic processes in the system uses an inference engine to apply linguistic rules to the data structure. The algorithms used in these inference engines only differ in terms of the specific level of the data structure that they operate on. The main task of the inference engine is to traverse through the appropriate level of the data structure, choose a rule to apply and provide the procedures which are called in the object code.

The inherent nature of the inference engines enforces a specific semantic model on the data structure. The inference engines use the data structure manager to built a data representation that reflects the linguistic structure of the input data. By operating on a specific level of the data structure, an inference engine adds a specific semantic interpretation to that level. The parser for instance, takes its input from the first level and generates segments for the second level of the data structure. This implies that the first level represents the sentence and the second level the different words from the sentence. If the synthesis model is changed, the changes will be absorbed by the inference engines and it will not affect the data structure manager.

Another important reason for using an inference engine is to separate the linguistic knowledge, which is contained in the rule sets, from the logic of the system.

9 Conclusions

In conclusion the system design is discussed in terms of the original design objectives.

The system caters for all the relevant linguistic processes involved in the translation of text to speech and is therefore suitable for the practical evaluation of these processes. It will thus be possible to determine if processes like syllabification aids in the synthesis of intelligible speech.

A clear distinction is made in the system between the specification and application of linguistic knowledge. All the linguistic information is represented in the rule sets and modifying these rules does not affect the logic of applying them to the data structure.

It is further maintained that the system provides a friendly and familiar user interface because

- it uses a rule format that is consistent with the rule format used in the linguistic theory.
- for each rule set the rule writer operates on a data model that is appropriate to the specific linguistic process and equivalent to the data model perceived in the linguistic theory.

The data structure closely models the real world and can thus be easily manipulated. A major advantage of the implementation of the data structure is the two levels of semantics associated with it, namely the physical level provided by the data structure manager and the logical level provided by the inference engines. This provides the necessary abstraction in terms of the user interface but at the same time provides a flexible system which can be adapted for different synthesis strategies and the synthesis of other languages.

A final comment regarding the design is that it lends itself ideally to a parallel implementation. The different linguistic processes can be executed in parallel with communication amongst

them handled through the central data structure manager. This will be pursued in a later version of the system.

References

[All76] Allen, J. 1976. 'Synthesis of Speech from Unrestricted Text.' Proceedings of the IEEE, 64(4), April, pp433-442.

[Car76] Carlson, R., Granström, B. 1976. 'A Text-to-Speech System based entirely on rules.' IEEE CASSP, pp686-688.

[Com87] Combrink, J.G.H., De Stadler, L.G. 1987. 'Afrikaanse Fonologie.' Macmillan Suid-Afrika .

[Dat86] Date, C.J. 1986. 'An Introduction to Database Systems.' Volume 1, Addison-Wesley.

[Her82] Hertz, S.R. 1982. 'From Text to Speech with SRS.' J. Acoust. Soc. Am., 72(4), October, pp1155-1170.

[Her85] Hertz, S.R., Kadin, J., Karplus, K.J. 1985. 'The Delta Rule Development System for Speech Synthesis from Text.' Proceedings of the IEEE, 73(11), November, pp1589-1601.

[How83] Howe, D.R. 1983. 'Data Analysis for Data Base Design.' Edward Arnold.

[Kla80] Klatt, D.H. 1980. 'Software for a cascade/parallel formant synthesizer.' J. Acoust. Soc. Am., 67(3), March, pp971-995.

[Kla82] Klatt, D.H. 1982. 'The Klattalk text-to-speech conversion system.' IEEE CASSP, pp1589-1592.

[Wag87] Wagener, M.J. 1987. 'Rekenaar Spraaksintese: Die Omskakeling van Teks na Klank.' Quaestiones Informaticae, Vol 5, No 2, October, pp1-6.

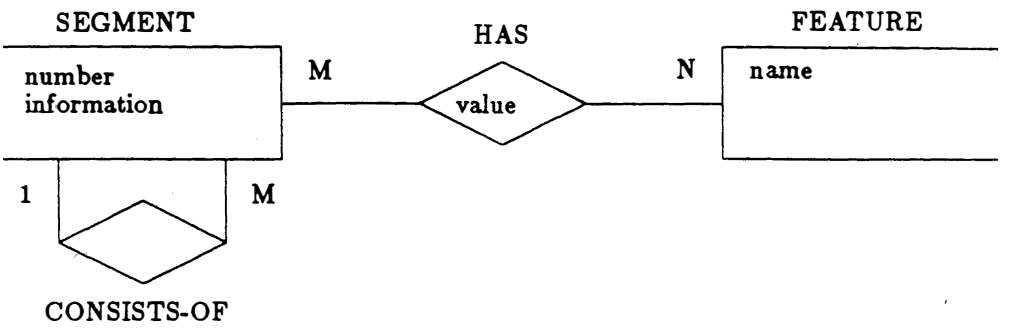

Figure 1: ER Diagram

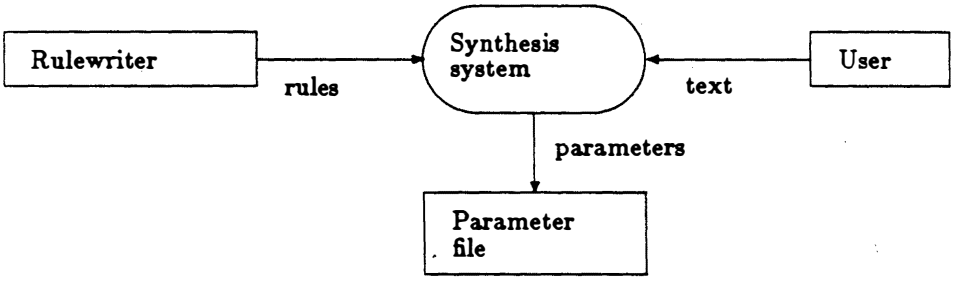

Figure 2: DFD Context diagram

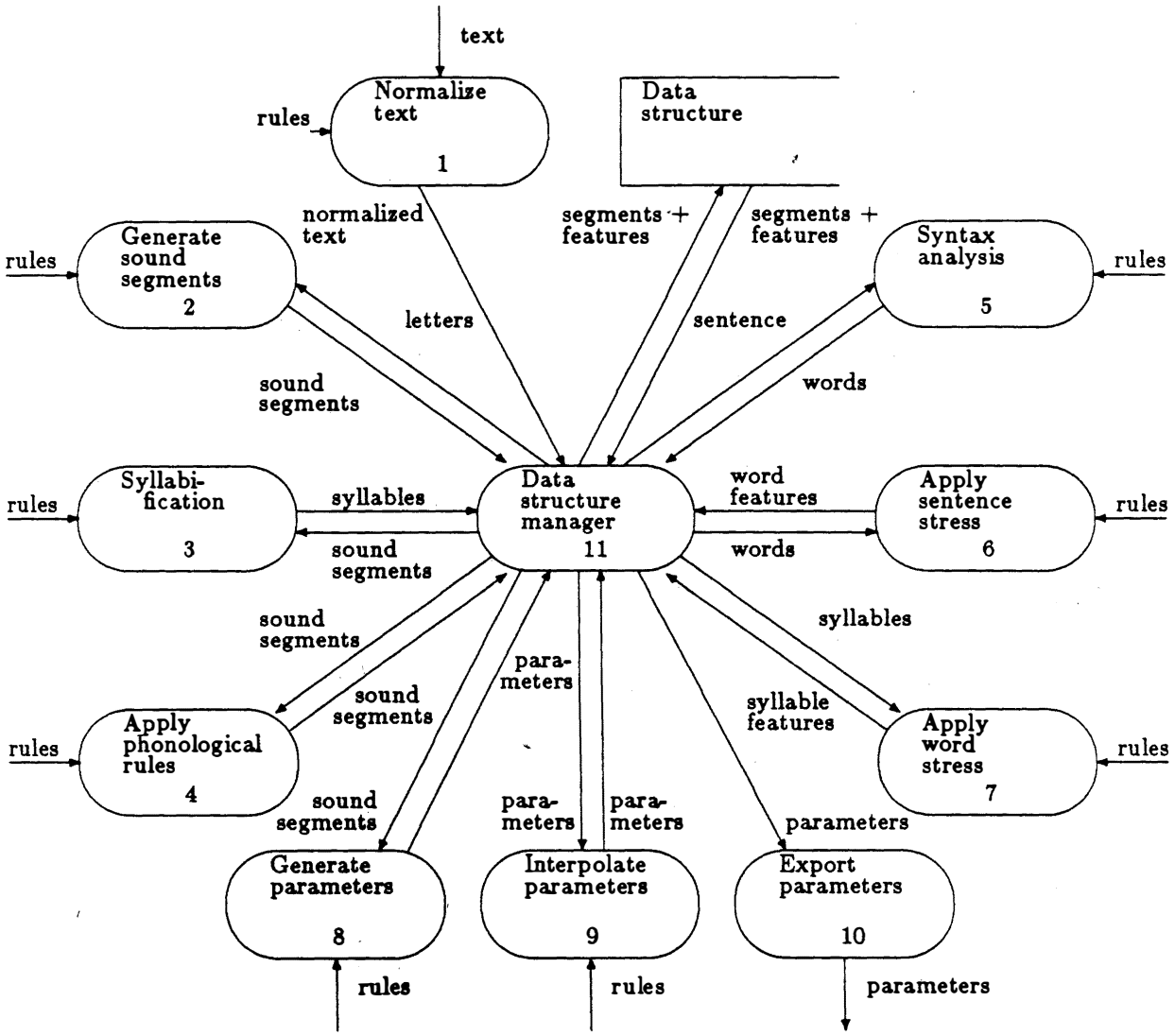

Figure 3: DFD Overview diagram

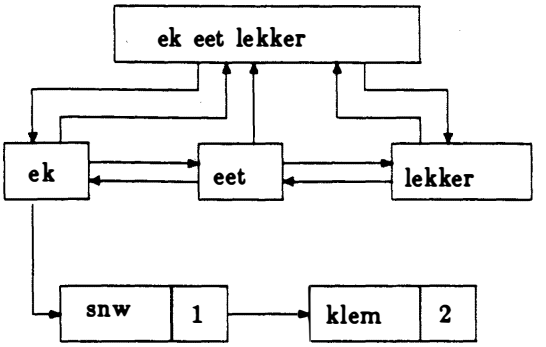

Figure 4: Example of physical data structure

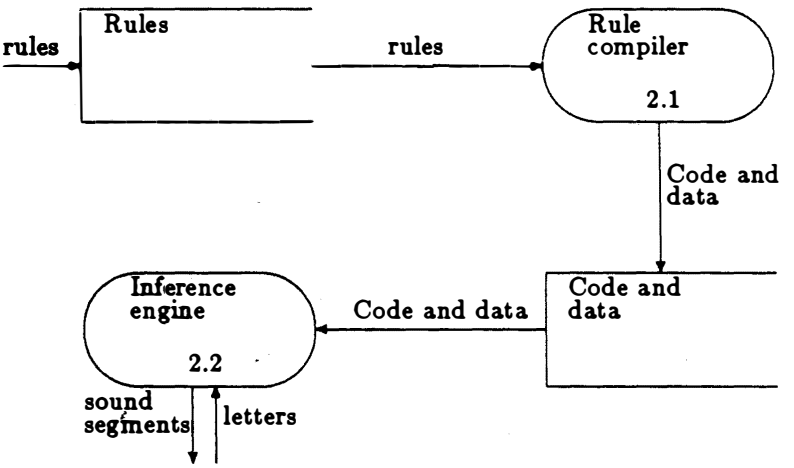

Figure 5: DFD Level 1 diagram

EXPERT SYSTEMS FOR MANAGEMENT CONTROL: A MULTIEXPERT ARCHITECTURE

VEVEK RAM
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF NATAL
PIETERMARITZBURG

ABSTRACT

The use of Expert Systems technology in management decision making domains is increasing rapidly as business environments worldwide grow more turbulent and as the cost of development tools decrease. Research effort in this field however, is concentrated largely on confined areas such as market analysis, financial diagnosis and production scheduling. The development of an Expert System to support a wider management area presents problems of both size and complexity since such a system would require a large monolithic knowledge base which would exhibit the associated problems of maintainability, consistency and reduction in inference speed.

This paper describes a blackboard based Multiexpert architecture that is capable of integrating the problem solving capabilities of a range of confined expert systems in order to provide problem solving support for a wide area such as management control at the strategic level. The system consists of several dedicated expert modules in the area of marketing, finance, production and so on as well as a control module that handles problem decomposition, task allocation and dynamic scheduling. A prototype version of such a system has been successfully implemented in Prolog.

INTRODUCTION

Many organizations engage in Long Term or Strategic Planning in order to match their internal capabilities with the opportunities and threats that exist in their operating environments. Such a match is characterised by the commitment of the organization's resources to achieve a desired objective and is often referred to as the organization's Strategic Posture. Management Control is the process whereby the organization continually re-assesses the appropriateness of the match and re-aligns its Strategic Posture to accommodate changes in the environment. The rate at which the organization is able to respond to changes in the environment is known as the Strategic Response Rate.

The intensification of global competition has emphasised the importance of a rapid Strategic Response Rate as only those organizations that are able to seize environmental opportunities early can compete effectively. Unfortunately, shifts in Strategic Posture involve the whole organization and the effects of it have to be assessed in various areas before changes can be implemented. This generates a lag in the response. For most organizations, posture shifts involve a reorganization of marketing, financial, production, research and human resources plans. Since these areas are separate in most organizations, the response lag can be attributed to the actual delay in assessment in each area and also the delay that can arise due to the communication and co-ordination between these departments. Computer-based systems in the form of Decision Support Systems and Expert Systems have to a large extent provided assistance

in reducing the problem in the individual areas. Descriptions of such systems can be found in King and Rodriques (1977), King and Dutta (1980), Klein and Newman (1980), Bouwman (1983), Smith et al (1985), Cooper (1986), Chandrasekaran and Ramesh (1987,1988), Goul (1987), Lee and Lee (1987) and Biswas (1988). Although the problem of communication and co-ordination can be solved by developing a single system that is representative of the collective activities of the various organizational areas, such a system would require a large and complex knowledge base. Large knowledge bases exhibit problems of maintainability and consistency (Prerau et al 1990). Also, there is a considerable reduction in inference speed and efficiency as the size of a knowledge base increases. The other alternative is to build a system that can integrate the functions of various individual systems by enabling them co-operate to solve a common problem while at the same time retaining their individual status. This approach is used extensively in the area of Distributed Artificial Intelligence and many useful techniques have been developed as a result. Specifically, the blackboard architecture (Erman et al (1980), Nii (1986) and Hayes-Roth, (1988)) and the centralised multiagent framework (Cammarata, (1983)) is most readily applicable. The blackboard architecture is based on a shared global data structure called the blackboard. The blackboard is divided into levels of varying abstraction depending on the application. Independent knowledge sources may read from and write to one or more levels of the blackboard. Multiagent frameworks use a single agent or knowledge source or a group of knowledge sources to form a coherent plan for solving a multiagent problem. Dependencies and potential conflicts among the agents are identified in advance. In centralised multiagent frameworks, one agent acts as the controller and coordinator for the whole network of agents. A combination of the centralised multiagent framework and the blackboard architecture facilitates the integration of discrete knowledge sources and enables the power of their collective knowledge to be used as a single large knowledge base without the associated problems. The remainder of this paper describes the architecture and operation of such a distributed system for use in the management control area. Construction details for some of the more important aspects of the system is also included.

ARCHITECTURE

The distributed management control system consists of a control module, a scanning module and several functional modules as shown in Figure 1. A brief description of the knowledge sources or modules and a discussion of their major roles in the distributed network follows.

The Control Module.

The control module acts as the strategic management expert and also as the manager of the network. As the strategic management expert, it controls the direction and format of the network problem solving process. It contains knowledge about the strategic management process and it also contains meta-knowledge, which is knowledge about how the rest of the system's knowledge is distributed throughout the network. This meta-knowledge allows the control module to decide that interest rates concern the financial expert, product cost concerns the production expert and so on. As the network controller, the control module controls the execution of individual modules as well as the management of the status of the blackboard.

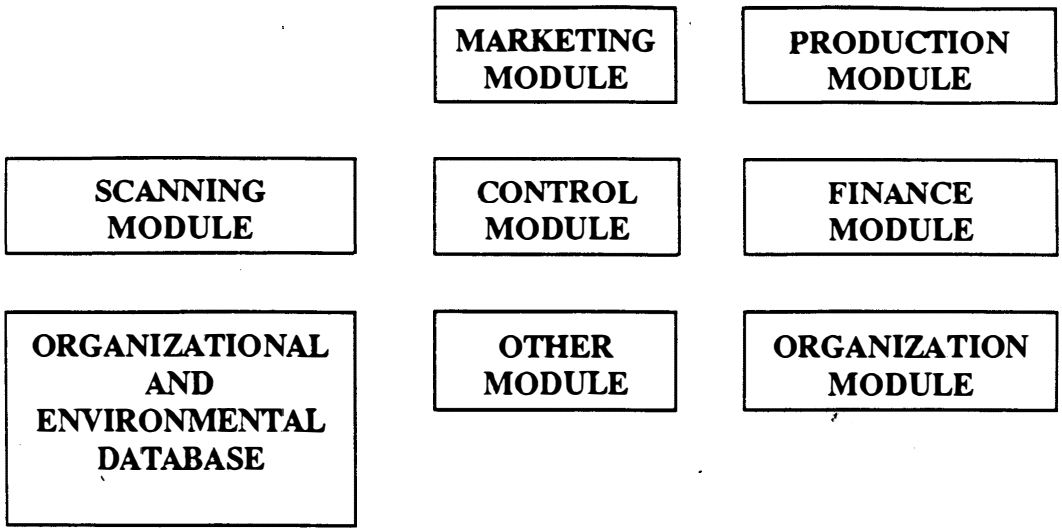

Fig 1. *The Distributed System Architecture*

The Scanning Module.
The scanning sub-system acts as the machine interface between the network and the organization. The scanning module monitors a set of strategic factors and reports all variances to the control module via the blackboard. It performs a simple but nevertheless important role in the network. The module is non-intelligent in that it reports all variances. The control module decides on the severity of an occurred variance.

The Functional Knowledge modules.
There are currently four functional modules in the prototype network. Each module contains conceptual knowledge of the domain area in general and also of specific policies of the organization in that area. The domain areas are Marketing, Finance, Production and Organizational. Each module has sufficient domain and control knowledge in order to function as a stand alone expert or knowledge-based system. Additional modules for other functional areas such as Research and Development or Distribution can be added when required.

OPERATIONAL OVERVIEW

Each knowledge source is responsible for maintaining a set of strategic variables in its own domain. Variables are categorised as either internal or external depending on whether the entity that a variable relates to is changed from within or outside of the organisation's boundary. As long as the values of these variables remain within predefined limits, there is a balance between the organisational ability, environmental pressure and a chosen strategy. Values for the internal variables are held in the organisational database which is constantly updated through the organisation's information system. External variables are updated through manual input on a regular basis. All variables are monitored by the scanning subsystem. When the value of a variable changes, the scanning subsystem communicates this change to the control module. The control module decides on the degree of severity of the variance (and others which may occur simultaneously), assigns priorities and then decides on which

modules need to be called in order to resolve the problem. It then posts a request with parameters describing the nature of the variance on the blackboard and activates the appropriate expert module or knowledge source. The individual knowledge source assesses the impact of the change in relation to the present strategic posture and communicates the result back to the control module via the blackboard. If the result concerns other knowledge sources, these are then activated by the control module. The process continues until a final result is obtained that is consistent with all the experts individual results. If two or more experts put forward recommendations that are conflicting, the control module can resolve the conflict by choosing the recommendation with the highest utility value or by modifying and reposting variables on the blackboard so that the individual experts reassess their respective recommendations and in so doing resolve the conflict themselves after a number of cycles.

COMMUNICATION AND CONTROL

The System Blackboard.

The blackboard does not exist as a physical entity in the system but rather as a communication mechanism through which the knowledge sources communicate both their requests and their findings. All individual knowledge source activity is initiated from the blackboard and all conclusions or results from knowledge sources are directed to the blackboard. In this application, the blackboard is in the control of the strategy formulation and control knowledge source. The system blackboard is divided into three main areas as shown in Figure 2.

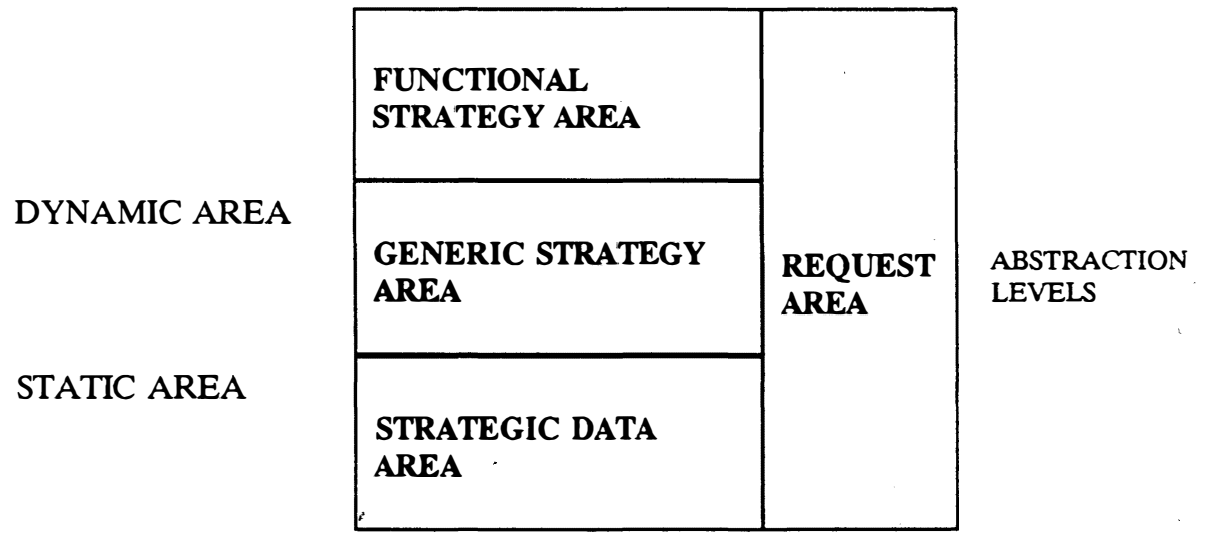

Figure 2. *The System Blackboard*

These three areas are used for static knowledge, dynamic knowledge and control knowledge respectively. Static knowledge is the domain specific knowledge that is relevant to the problem and normally remains relatively stable during the solution process. In the system blackboard, the static knowledge area holds the collection of organizational data that is scanned by the scanning module. Dynamic knowledge is knowledge that is generated during the execution of the system. It consists typically of new facts, hypotheses and suggestions that are made by the knowledge sources. In the

system blackboard, the dynamic knowledge is divided into the generic strategy area and the functional strategy area. The generic strategy area holds heuristic suggestions that are made by the control module and which are used by the functional knowledge sources to restrict their search spaces. The functional strategy area contains all the functional strategies, which are the solutions to a strategic problem, that are generated by the individual sources. Control knowledge is knowledge about the current state of the network itself and also of the status of the problem solving. In the system blackboard, the control knowledge is made up of a set of requests which form a dynamic queue. The requests are either from the control module to a functional module or vice-versa. The control module extracts from this request list a single request which it then converts into a call to an individual module. The responsibility of keeping the blackboard "clean", that is, erasing old or unwanted entries or archiving previous entries rests with the control module. This is an essential activity since the blackboard tends to become cluttered after a reasonable amount of network activity and this can lead to a degradation of the network efficiency.

Scheduling and Control

Network control can be achieved by selecting an individual knowledge source and calling on it to execute inside a problem solving cycle, or it can be achieved by placing knowledge on the blackboard that will cause a knowledge source to execute on its own. The support system uses the strategy formulation knowledge source as the network controller and therefore makes use the former method. The network as a whole makes use of three control mechanisms: Goal-Driven control, which is the control exerted on the network to attain a network-wide or global system goal; Request-Driven control, which is the control exerted on the network by inter knowledge-source requests, and Event-Driven control, which is the control exerted on the network due to the occurrence of certain events. The goal-driven function of control is the classical strategic management function of monitoring and controlling of strategic and functional plans. The goal of the network is to ensure that the implemented strategy adheres to certain performance limits that were used in the formulation of the strategy. Variances that exist obviously affect the strategic posture as a whole and must be accommodated at the strategic level. Variances are translated into network action by the event-driven control mechanism. A variance is regarded by the scanning subsystem as a strategic event that triggers the network into a resolution process. The event-driven control function is to alert the control module into initiating the network. Individual knowledge sources would then attempt to reduce the variance or the effects of it and failing this, to reevaluate the strategy. The request-driven mechanism allows the control module as well as other modules to alter the direction of the problem solving process in a dynamic way by posting requests on the blackboard. These requests may be for further information or for initiating the activation of other modules. There may be many such requests on the blackboard at any one time and in a serial network, that is a network in which the knowledge sources cannot be executed in parallel, these requests need to be serviced in some sort of order. This is accomplished by establishing a schedule of ready-to-be-called modules. This schedule must be flexible enough to be dynamically modified since the execution of one module may cause others that were ready to execute to be no longer required or, the execution of one module may cause others that were not executable, ready to execute. Also, there may be more than one consecutive request for the same module, and the schedule must allow a single activation call with all the

requests consolidated rather than allow more than one consecutive call to the same module.

IMPLEMENTATION

The prototype version of the distributed management control system has been implemented in Prolog on a microcomputer. As the exact constructional details are beyond the scope of this paper, only the more unusual aspects of communication, task decomposition and dynamic scheduling are described. Also, as each individual knowledge module is a conventional production rule and frame based Expert System these will not be examined. Further detail can be found in Ram (1990a, 1990b).

Communication.

The mechanism used to control and facilitate the flow of information in the network is the blackboard. The blackboard is a communication and storage mechanism which is accessible by all the modules and which is divided into different levels. Prolog has an internal database which can be modified during execution and can also be stored and retrieved. This forms a convenient implementation of the blackboard. The different levels can be distinguished by using a separate predicate for each level. At the static level which holds organizational data for example, the predicate has the form;

Factval(FactorName,value)

An example of such a clause is Factval("InterestRate","LOW").

The control level of the blackboard is used to hold control information for the scheduling mechanism. This information is in the form of request clauses that are inserted into the blackboard at the control level by those functional modules that require assistance or additional information. Requests are held in the following format

Request(CallMod,DestMod,Factor,Action,Ref,Status)

CallMod identifies the module issuing the request, DestMod is the module to which the request is directed, Factor and Action are as before, Ref is a request reference number and Status indicates the status of the request. A status value of "U" denotes unresolved and a status value of "OK" or "NotOK" denotes a resolved request. When the control module assembles requests into a queue, it examines the status value in each request and ignores requests that have already been resolved. An example of a request is

Request("MARK",,"PRICE","DEC",1,"U").

This is a request from the marketing module requesting that the control module investigate the possibility of a reduction in product price. Requests in the individual modules are typically invoked by rules which test for the existence of required data. The DestMod slot is left blank since the individual modules do not have knowledge of each others expertise. The control module, through its decomposition procedure, decides on the module to which it can best delegate the resolution of the request and fills the DestMod slot before the delegated module is called. If the request can only be resolved by more than one module, the control module issues as many requests as the decomposition procedure generates.

Problem Decomposition

When an individual functional module encounters a subproblem during its problem solving activity that is outside its domain of expertise, it would issue a request to the control module for assistance. It is the function of the control module to redirect these requests to the appropriate modules. A major problem for the control module in the

execution of this function is "knowing" which module to call for a given request. A simple and effective way to overcome this problem is to maintain a list that links all the relevant organizational data items with the modules responsible for them. Such a list represents Meta-level knowledge since it represents knowledge about the use of the distributed expertise in the most efficient way. When a request that can be resolved by a single module is received, the control module need only scan the list in order to identify the module best suited to resolving the request. A problem arises when a request is received that cannot be resolved by one module alone. Such a request has to be decomposed into subrequests that can be resolved by individual modules. This decomposition process can be implemented by organizing the decomposition relationships into a taxonomy of meta-knowledge frames as follows.

MFrame(Problem,PRef,DecompList,Dmodule)

where:

MFrame is a label distinguishing the Meta-Knowledge frames from other frames in the program;
Problem is the label identifying the problem that this instance of the frame is representing;
PRef acts as a reference number for the problem represented by this frame and is used to establish priorities in the problem solving process;
DecompList is a list of all the subproblems that Problem can be decomposed into;
Dmod is the Domain module responsible for solving Problem and is only present in a frame if the DecompList contains a single element, or if it contains more then one element, then all these elements are the responsibility of the same domain module.

Consider as an example, that the scanning module has picked up a drop in the market share. One option for strategic realignment is to restore the situation by stimulating primary market demand which expands the total market or by stimulating selective demand which increases market share within the existing market. A marketing action plan of reducing product price or increasing advertising can achieve both these. Since product price is outside the domain of the marketing module, it will request the control module to investigate the feasibility of price reduction. The control module has to refer this request to the appropriate module or modules and makes use of the meta-knowledge frame taxonomy search to decide which module or modules are appropriate. The search begins by finding a frame which has price as the label in the problem slot.

MFrame("Price",1,["Cost","Margin"],)

The PRef slot is arbitrarily set to 1 and the domain slot is empty since DecompList contains more than one element. This frame represents the decomposition of the price problem into the two subproblems of cost and margin. The search then continues by finding a frame for each of the elements in the DecomList. These are found as

MFrame("Cost",1,["ProdCost","Ohead"],)
MFrame("Margin",1,["Margin"],FIN)

The first frame further decomposes the cost problem into the two subproblems of production cost and production overhead. The second frame asserts that margin cannot be decomposed further and that it is the responsibility of the FIN or Financial module. The control module continues the search by finding frames with Prodcost and Ohead as labels in the problem slot. This produces the following frames

MFrame("Prodcost",1,["ProdCost"],PROD)

MFrame("OHead",1,["OHead"],FIN)

Since both these frames contain only one element in their respective DecomLists, the search terminates and the control module posts a request to the PROD or production module to investigate the reduction in product cost. The production module contains rules that relate product cost to raw material and labour costs and so is able to function independently in solving this subproblem. The control module also posts a request to the FIN or financial module to investigate the possibility of a reduction in profit margin and production overhead. Both the financial and the production modules communicate the results of their investigations to the control module. Both the requests derived from the decomposition have the same PRef number as the original request and the scheduling mechanism uses this number to keep them in the same logical group.

Control of Dynamic Scheduling

Dynamic Scheduling is accomplished by establishing and managing a queue of Ready-to-be-called modules. The queue is represented by a prolog list and is constructed by examining all the requests held in the control level of the blackboard. The scheduling procedure terminates when the queue is empty which occurs when there are no unresolved requests on the blackboard. Once a queue has been constructed, the control module calls the functional module represented by the first entry in the queue. When the call terminates, that is, when the functional module has completed its task, the control module then reconstructs the queue and the process is repeated. Reconstructing the queue each time a functional module call is terminated, ensures that the scheduling mechanism makes use of the most current problem solving knowledge available. This is necessary since at any stage, a called module may issue a request and suspend its problem solving activity until the request is resolved. The module chosen for investigating this new request must be inserted at the head of the queue and called. On its termination, the original module which is waiting for the response is called and continues its task.

CONCLUSION

A description of a distributed knowledge-based management control system has been presented. A prototype version which has been implemented in Prolog, has generated very favourable results in an important area. It is hoped that the success on this limited scale will encourage further research in other areas. It is considered that the most important aspect of this work is the illustration that complex areas requiring knowledge-based support can be structured into relatively self-contained knowledge modules which can then be integrated into a system which, while addressing the original problem, is easier to build, debug and maintain.

REFERENCES

Biswas, G., Oliff, M. and Sen, A., (1988), An Expert Decision Support System for Production Control, Decision Support Systems, 4, 235-248.

Bouwman, M.J., (1983), Human diagnostic reasoning by Computer: An Illustration from financial analysis, Management Science, 29 (6), 653-672.

Cammarata, S. et al, (1983), Strategies of Cooperation in Distributed Problem Solving, Proceedings of the IJCAI.

Chandrasekar, G. and Ramesh, R., (1987), Microcomputer Based Multiple Criteria Decision Support System for Strategic Planning, Information and Management, 12, 163-172.

Chandrasekar, G. and Ramesh, R., (1988), An Integrated Framework for Decision Support in Corporate Planning, Decision Support Systems, 4, 365-375.

Cooper, P., (1986), Expert Systems in Management Science, in Bertold, T., (ed), Expert Systems and Knowledge Engineering, 61-71, Elsevier, Holland.

Erman, L, Hayes-Roth, F., Lesser, V. and Reddy, D., (1980), The Hearsay II speech understanding system: Integrating knowledge to resolve uncertainty, Computing Surveys, 12 (2), 213-253.

Goul, M., (1987), On Building Expert Systems for Strategic Planners: A Knowledge Engineers experience, Information and Management, 12, 131-141.

Hayes-Roth, B., (1988), A Blackboard Architecture for Control, in Bond, A.H. and Gasser, L., Readings in Distributed Artificial Intelligence, 505-540, California, Morgan Kaufman.

King, W.R. and Dutta, B.P., (1980), A Competitive Scenario Modelling System, Management Science, 26, 261-273.

King, W.R. and Rodriquez, J.I., (1977), Competitive Information Systems, Long Range Planning, 10, 59-64.

Klein, H. and Newman, W., (1980), How to use Spire: A systematic Procedure for Identifying Relevent Environments for Strategic Planning, Journal of Business Strategy, Summer, 32-45.

Lee, J.K. and Lee, H.G., Interaction of Strategic Planning and Short-Term Planning: An Intelligent DSS by the Post-Model Analysis Approach, Decision Support Systems, 3, 141-154.

Nii, H.P., (1986), Blackboard Systems, The Blackboard model of Problem Solving and the evolution of Blackboard Architectures, AI Magazine, 7 (3).

Prerau, D.S., Gunderson, A.S., Reinke, R.E. and Adler, M.R., (1990), Maintainability techniques in Developing Large Expert Systems, IEEE Expert, June, 71-79.

Ram, V., (1990a), A System of Cooperating Experts for Strategic Planning Support in Business, in Mallouppas, A., (Ed), Management Technology: Control Tools for the 90's, London, Peter Perigrinus Ltd.

Ram, V., (1990b), A Distributed Knowledge-Based Support System for Strategic

Management, Unpublished PhD Thesis, University of Natal, Pietermaritzburg.

Smith, L.D., Blodgett, J., Janson, M. and Bartle V., (1985), Decision Support for Marketing Research and Corporate Planning, Information and Management, 8, 133-145.