

205  
A TOOL FOR USE IN  
ON-LINE DATA DICTIONARY CREATION

by

CHINHO LAI

B. S., Soochow University, 1975

---

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY  
Manhattan, Kansas

1983

Approved by:

  
Major Professor

LD  
2668  
R4  
1983  
L34  
C.2

111202 244805

TABLE OF CONTENTS

PAGE

I. INTRODUCTION . . . . .	1
1. Problems . . . . .	1
2. The data dictionary . . . . .	6
3. Summary . . . . .	7
II. DATA DICTIONARY . . . . .	9
1. Definition . . . . .	9
2. Advantage of data dictionary . . . . .	12
III. TOOLS FOR USE IN AN ON-LINE DATA DICTIONARY . . . . .	20
1. User-friendly display device . . . . .	24
2. Transaction processing support . . . . .	28
3. Spelling checker. . . . .	29
4. Transaction formatter . . . . .	31
IV. IMPLEMENTATION . . . . .	36
1. Data dictionary data base . . . . .	38
2. SHADOW II . . . . .	40
3. Data dictionary processor . . . . .	46
4. User interface . . . . .	48
5. System design consideration . . . . .	53
V. CONCLUSIONS . . . . .	55
1. Conclusions . . . . .	55
2. Future research . . . . .	56
APPENDIX	
A. File description with/without data dictionary . . . . .	61
B. Data dictionary data base . . . . .	64
C. Maps definition . . . . .	75
D. Design screen . . . . .	80
E. Punch file output for design screen . . . . .	86
F. Source program . . . . .	115

	PAGE
2.1 Data Dictionary Interface in Data Base Environemnt . .	11
3.1 Basic On-Line System Flow Diagram . . . . .	23
3.2 Display Interface Logic Flow . . . . .	27
4.1 Data Dictionary System Flow . . . . .	37
4.2 On-Line Mapping Environment . . . . .	43
4.3 Algorithms For Processess . . . . .	51
4.4 Structure of Screen Invocations . . . . .	52

## ACKNOWLEDGEMENTS

I would like to thank Dr. Elizabeth Unger for her constant guidance, suggestions, and tireless editing. I would like to thank Drs. David Gustafson and Rod Bates for serving as committee members. Thanks also to my fiancée, Esther, for her encouragement.

## CHAPTER 1. INTRODUCTION

As the eighties are upon us, one fact stands out clearly: past practices and procedures in the data processing (DP) field are inadequate to meet the challenges of the future. There is a need to increase the productivity of an enterprise, to stabilize software costs, and to plan and manage the most critical resource of the eighties, the information resource, in a more effective fashion. In this report, we address a suggested partial solution to these needs, the data dictionary. The focus is the creation of a dictionary of data about data, i.e., metadata, by the ultimate user of the data.

### 1.1 Problems

Conventional data definition has been inadequate. The manner in which data has been defined in most of today's application systems has been a function of (1) the programming language used, (2) the characteristics of the programming language used, and (3) the habits of the individual programmers who have written the programs. Some of today's programming languages have the characteristics that data definitions can be embedded in the executable statements. If data definitions are only used in one program, no data administration problem is raised. An advance toward disciplined programming occurred when the data division was defined in the COBOL language. In this division, all of the data definitions used by a program

are stated explicitly. This made the task of retrieving all data definitions much easier than in languages which allow definitions to be scattered throughout the programs. COBOL did not fully solve the problem of controlled data definitions. If the same data file is used by five programs, each program would include the data definitions for that file. If changes are made in the data definitions for file, it is necessary that all programs that use the file be identified and their data divisions changed.

Many user organizations have ended up with multiple data files in which the same data item has been defined differently and or with a different name, different field length, different updating cycle, etc. (semantic redundancy). Additionally, different data items have ended up with the same name (syntactic redundancy). Sometimes the same data item has been defined "well" in certain instances and "poorly" in others, to the point where the same data item in two files cannot be compared. Other difficulties from conflicting data definitions have included (1) the need for special programs to convert 'division' data to conform with the 'corporate' data definitions, (2) loss of data integrity, due to mistakes because of the multiple definitions of the same data item, (3) confusion in interpretation, with consequent mistakes in decision making, due to the multiple definitions, (4) loss of opportunity for using the data for additional purposes, because