2004

Recent Trends in Software Engineering Research As Seen Through Its Publications

Terry L. Smith
University of North Florida

RECENT TRENDS IN SOFTWARE ENGINEERING RESEARCH AS SEEN THROUGH ITS PUBLICATIONS

by

Terry L. Smith

A thesis submitted to the
Department of Computer and Information Sciences
in partial fulfillment of the requirements for the
degree of

Master of Science in Computer and Information Sciences

UNIVERSITY OF NORTH FLORIDA
DEPARTMENT OF COMPUTER AND INFORMATION SCIENCES

April 2004

The thesis "Recent Trends in Software Engineering Research as Seen through Its Publications" submitted by Terry L. Smith in partial fulfillment of the requirements for the degree of Master of Science of Computer and Information Sciences has been

Approved by the thesis committee: Date

Signature deleted

4/14/04

Dr. Neal S. Coulter
Thesis Adviser and Committee Chairperson

Signature deleted

4/16/04

Dr. Charles Winton

Signature deleted

4/16/04

Dr. Yap S. Chua

Accepted for the Department of Computer and Information Sciences:

Signature deleted

4/22/04

Dr. Judith Solano
Chairperson of the Department

Accepted for the College of Computing, Engineering, and Construction:

Signature deleted

4/22/04

Dr. Neal S. Coulter
Dean of the College

Accepted for the University:

Signature deleted

5/7/04

Dr. Tom Serwatka
Dean of Graduate Studies

ACKNOWLEDGEMENT

It would be impossible to list everyone whose guidance and encouragement made this endeavor possible, but of special mention are Ms. Carrol Reilly, Dr. Neal Coulter, Dr. Yap Chua, and Dr. Charles Winton.

For my family, especially Daniel, please accept my most heartfelt gratitude for your continued faith and encouragement.

I wish to offer special thanks to Dr. Suresh Konda. Dr. Konda's pioneering work in the development of the CAIR system, its port to Linux, and making CAIR available to the University of North Florida has made this study possible.

CONTENTS

FIGURES

TABLES

ABSTRACT

This study provides some insight into the field of software engineering through analysis of its recent research publications. Data for this study are taken from the ACM's Guide to Computing Literature (GUIDE). They include both the professionally assigned Computing Classification System (CCS) descriptors and the title text of each software engineering publication reviewed by the GUIDE from 1998 through 2001.

The first part of this study provides a snapshot of software engineering by applying co-word analysis techniques to the data. This snapshot indicates recent themes or areas of interest, which, when compared with the results from earlier studies, reveal current trends in software engineering.

Software engineering continues to have no central focus. Concepts like software development, process improvement, applications, parallelism, and user interfaces are persistent and, thus, help define the field, but they provide little guidance for researchers or developers of academic curricula.

Of more interest and use are the specific themes illuminated by this study, which provide a clearer indication of the current interests of the field. Two prominent themes are the related issues of programming-in-the-large and best practices.

Programming-in-the-large is the term often applied to large-scale and long-term software development, where project and people management, code reusability, performance measures, documentation, and software maintenance issues take on special importance. These issues began emerging in earlier periods, but seem to have risen to prominence during the current period.

Another important discovery is the trend in software development toward using networking and the Internet. Many network- and Internet-related descriptors were added to the CCS in 1998. The prominent appearance and immediate use of these descriptors during this period indicate that this is a real trend and not just an aberration caused by their recent addition.

The titles of the period reflect the prominent themes and trends. In addition to corroborating the keyword analysis, the title text confirms the relevance of the CCS and its most recent revision.

By revealing current themes and trends in software engineering, this study provides some guidance to the developers of academic curricula and indicates directions for further research and study.

Chapter 1

INTRODUCTION

This study uses content analysis techniques to examine a large volume of software engineering research publications to determine themes and trends both in the specific discipline of software engineering and in the general field of computer science. It is believed that an understanding of these themes and trends would be a useful and effective guide for curriculum, research, and application.

The data for this empirical study are taken from the Association for Computing Machinery's (ACM) Guide to Computing Literature (GUIDE). The GUIDE reviews and indexes a wide range of computing literature, including individual articles, journals, trade magazines, book chapters, whole books, and other published materials. The GUIDE is carefully indexed

by professionals using the ACM Computing Classification System (CCS), which provides a standard method for categorizing publications included in the GUIDE by assigning descriptors (or keywords) to each publication.

A variety of content analysis techniques exist to aid in the study of textual data. Similar to co-citation analysis [see SMALL73] and bibliographic coupling [see KESSLER63], this study examines the co-occurrence of textual phrases within the data set of indexed publications related to the field of software engineering.

This study follows up and expands on an earlier study [COULTER98B] that applies co-word analysis techniques in the examination of GUIDE classifications of publications from 1982 through 1994. This study continues this analysis for publications from 1998 through mid-2001. The choice of the period, 1998 - 2001, is a natural one, as the data set contains

relatively current data and allows for an examination of the GUIDE since the last update to the CCS. A comparison of the results of the analysis with that of the earlier study provides an excellent opportunity to discover patterns and trends in software engineering research.

In addition to analyzing the GUIDE classifications of the publications in the 1998 - 2001 time period, this study also examines the title text. It is believed that such an examination reveals general terms that help define the field of software engineering. Additionally, the title data analysis may offer corroboration of the results of the descriptor (keyword) data analysis.

Chapter 2

THE DATA

The ACM's Guide to Computing Literature (GUIDE) provides an enormous repository of data for this study. Publications indexed by the GUIDE include individual articles, journals, trade magazines, book chapters, books, conference proceedings, and other items of computing literature. This study examines a portion of the GUIDE data from 1998 through mid-2001.

Key to indexing in the GUIDE is the ACM's Computing Classification System (CCS). The CCS is a "carefully designed and maintained taxonomy" [COULTER98B, page 1207] used to categorize publications and provide keywords for sorting and searching.

Professional indexers assign publications to one or more CCS categories, taking into consideration that

publications may span multiple subjects. As part of the category assignment, proper subject descriptors (or keywords) and implicit subject descriptors (mostly proper nouns, like "C++" and "Grace Murray Hopper") are associated with each publication. Both types of descriptors provide the textual data to which co-word analysis techniques are applied in this study.

Variations in the application of the CCS are averaged out in this study by including a large volume of publications. This study uses those publications indexed by the GUIDE from 1998 through the first half of 2001 that include at least one descriptor from the "Software Engineering" category (D.2) of the CCS.

2.1 The CCS

The current version of the ACM Computing Classification System (CCS) is based on the framework established in 1982 when it was published as the "Computing Reviews Classification System" [see SAMMET82]. It has been revised four times since, in 1983 [SAMMET83], 1987 [SAMMET87], 1991 [COULTER91], and 1998 [COULTER98A].

The CCS provides a fixed system of descriptors (or keywords), which imposes a common nomenclature across all computing literature. Professional indexers assure that this system is applied to the computing literature as homogeneously as humanly possible. Considerable research continues to be done on the effectiveness of automating this process [see BORKO63, WONG96, and SEBASTIANI02].

The CCS is a hierarchal structure with "11 top-level nodes and a maximum of four levels of nodes" [COULTER98A, p. 111]. Appendix A lists the top two levels of the CCS classification tree. The first level provides very broad categories designated by letters (A through K). This is followed by more specific levels, which are designated by numbers or letters. For example, "D" designates the "Software" category, "D.2" designates "Software Engineering," and "D.2.8" designates "Metrics."

Indexers associate descriptors with the publications they review for the GUIDE. Descriptors (or keywords) come from three sources: category names (such as "Metrics"), explicit subject descriptors, and implicit subject descriptors. Explicit subject descriptors are text associated with most leaf nodes of the CCS tree and are published as part of the CCS. For example, the D.2.8 explicit subject descriptors are "Complexity measures," "Performance measures," "Process metrics," "Product metrics," and "Software science."

The names of people, systems, languages, and such are not included as part of the published CCS. However, indexers may choose from select proper nouns, called implicit subject descriptors, which can be used to further specify the subject of a given publication. Some implicit descriptors are "Alan Turing," "C++," "DARPA," "IBM," "QuickBASIC," "UNIX," and "World Wide Web (WWW)."

In addition to the text already discussed, indexers may specify general terms that are not associated with any specific CCS category but which may apply to any category. Table 1 lists the general terms that can appear in the data of this study.

Algorithms	Management
Design	Measurement
Documentation	Performance
Economics	Reliability
Experimentation	Security
Human Factors	Standardization
Languages	Theory
Legal Aspects	Verification

Table 1: CCS General Terms

The data for this research include publications indexed with at least one descriptor from the D.2 Software Engineering category of the CCS. Table 2 lists the level-three descriptors for this category. Since the documents of this study may be assigned descriptors from other CCS categories in addition to D.2 categories, one may learn something of the interactions between software engineering and other computing fields by examining the co-occurrences of these descriptors.

D.2.0 General
D.2.1 Requirements/Specifications
D.2.2 Design Tools and Techniques
D.2.3 Coding Tools and Techniques
D.2.4 Software/Program Verification
D.2.5 Testing and Debugging
D.2.6 Programming Environments
D.2.7 Distribution, Maintenance, and Enhancement
D.2.8 Metrics
D.2.9 Management
D.2.10 Design
D.2.11 Software Architectures
D.2.12 Interoperability
D.2.13 Reusable Software
D.2.m Miscellaneous

Table 2: Software Engineering Descriptors

2.2 SGML Data Set

The D.2 Software Engineering portion of the ACM GUIDE database is delivered for this study as several files in Standard Generalized Markup Language (SGML). Each SGML file contains a wealth of information about publications that were added to the GUIDE during a specific year. Depending on the type of publication,

a record may contain the title, authors or editors, publication year, journal name, abstract, category codes, and keywords. A sample record for a single publication (in this case, a journal article) is reproduced in Appendix B.

As a markup language, SGML provides a method for specifying data in human-readable plain-text. For example, the title of a publication in this study is specified by placing the title text between <TITLE> and </TITLE> tags. <TITLE> and </TITLE> are referred to herein as the TITLE tag-pair. Table 3 provides descriptions for some of the tag-pairs found in the data of this study.

Tag-Pair	Delimits …
STARTREC	Record for a single publication.
TITLE	Title text.
SUB	Subtitle text.
AUTHEDIT	Name of an author, editor, chairperson, or translator.
AUTHTYPE	AUTHEDIT type for the name specified in the preceding AUTHEDIT field, which may be AUTHOR, EDITOR, CHAIRPERSON, or TRANSLATOR.
PUBTYPE	Publication type, which may be BOOK CHAPTER, DIVISIBLE BOOK, DOCTORAL THESIS, JOURNAL ARTICLE, MASTER'S THESIS, PROCEEDINGS PAPER, REPORT, WHOLE BOOK, WHOLE JOURNAL, or WHOLE PROCEEDINGS.
JRNLNAME	Name of the journal, if applicable.
GENTERM	A general term assigned to the publication by an indexer.
PRICATDESC	Primary subject descriptors associated with the PRICATCODE that follows.
PRICATCODE	Primary CCS category code, such as D.2.2.
DESCRIPTOR	Subject descriptors associated with the CATCODE that follows.
CATCODE	CCS category code, such as F.3.1.
ABSTRACT	Abstract for the publication.
REVWTEXT	Text of the review of the publication.

Table 3: Some SGML Tag-pairs.

Some tag-pairs may appear multiple times in a given record and some tag-pairs must always appear together with other tag-pairs. For instance, AUTHEDIT may appear for each author, editor, chairperson, or translator listed for a given publication. DESCRIPTOR and CATCODE may also appear multiple times, but they must always appear together.

This study makes use of the text of the TITLE, PRICATDESC, PRICATCODE, DESCRIPTOR, and CATCODE fields.

2.3 Initial Examination

The data, as delivered, are in the form of a number of SGML files, each labeled with a year. For this study, the 1998, 1999, 2000, and 2001 data files are used. Before proceeding to parse and format the data, some idea is needed of what data are actually available in these files. The simplest approach is to perform some

counts. This can be accomplished with some basic commands found in many UNIX and UNIX-like operating systems.

Table 4 lists the number of records in each data file. These numbers may be obtained by issuing the following command at the system prompt:

```
cat yeardata.sgml | grep -c "<STARTREC>"
```

where "yeardata.sgml" represents the SGML data file for a given year.

Year	No. of Records
1998	1590
1999	1194
2000	1379
2001	810

Table 4: SGML Record Counts

There are 4973 records in the SGML data of these four year files. Before an accurate count of the number of actual publications for each year can be obtained, it is necessary to ensure that the data files contain records for only documents published in the specified year and that the intersection of the data files is empty.

Since each record contains a PUBYEAR field, it is relatively easy to obtain a list of the publication years contained in each data file. The following command can be issued to obtain this list:

```
cat yeardata.sgml | grep "PUBYEAR" | sort -u.
```

The results for the 1998 SGML data file, for example, include PUBYEAR values of 1996, 1997, 1998, 1999, and 2000. This means that the SGML data files contain publications for more than the specified year, raising the possibility of duplicate records.

Chapter 3

PREPARING THE DATA

This study will use the Context Analysis and Information Retrieval (CAIR) system, produced at the Carnegie Mellon University Software Engineering Institute, to perform co-word analysis and generate graphical networks for publications between the years 1998 and 2001. To accomplish this, considerable manipulation of the raw SGML data is required before they may be fed into the CAIR system.

3.1 CAIR-Prep

It is a daunting task to manually select publication records for a given year, ensure their uniqueness, and reformat them for the CAIR system. Fortunately, a software solution already exists to accomplish much of

this. CAIR-Prep is a program designed by Hammond, et al. [see HAMMOND99] to clean up the ACM SGML data files and prepare them for analysis by the CAIR system.

CAIR-Prep takes as input an SGML data file, the current CCS specification, and a list of valid implicit subject descriptors. For each publication year found in the SGML data file, CAIR-Prep generates two text files: one containing the publications' subject descriptors and one containing their titles. CAIR-Prep also generates an error file that provides a list of invalid descriptors found in the SGML data.

Fortunately, the "invalid descriptors" in the SGML data of this study are minor and easily corrected. The most common error involves the inclusion or exclusion of text used to clarify particular descriptors. For example, the D.2.1 category includes the descriptor, "Methodologies," which may include the additional text, "(e.g., object-oriented,

structured)." If such additional text is missing from the SGML data, CAIR-Prep would list the descriptor as being invalid. Likewise, the SGML data may include example text not found in the version of the CCS specification used by CAIR-Prep and, so, that descriptor would also be listed as invalid.

The simplest solution to this problem involves the removal of the additional text from both the CCS specification used by CAIR-Prep and from the SGML data. These deletions do not impact the validity of this data set, as the additional text does not change the assignment of the keywords (CCS descriptors).

After correcting the "invalid descriptors" and re-running CAIR-Prep for each SGML data file, a series of new data files are generated. A sample of the generated keyword and title files are reproduced in Appendices C and D.

Both files follow the basic format presented in Figure 1. CAIR-Prep keeps a running count of the number of valid publication records it discovers, which is used to generate the document_number for each record in the output file. The "1998" seen in the sample records shown in Appendices C and D refers to the CCS revision year, not the year of publication.

```
\*
\#
document_number
\#
\!
document_text
\!
\*
```

Figure 1: CAIR-Prep Results File Format

The document_text for the title file is simply the title text. For the keyword file, however, it includes descriptor text concatenated with the associated CCS category code in the format, "-1 (descriptorcode) () 0." The descriptor text included here is not the main category descriptors, but, rather, the leaf-node descriptors actually assigned by the indexer. Hence, "assertion checkersd.2.4" may appear as a keyword even when the D.2.4 category name, "Software/Program Verification," does not. This may seem odd and, possibly, a loss of valuable data. But, it should be remembered that the leaf nodes are more specific than the category names and, thus, provide a much better indication of the subject of a publication.

3.2 Final Preparations

CAIR-Prep generates a separate file for each publication year discovered in the SGML data. So, for each SGML data file, several "year" files are generated. For example, the 1999 SGML data file spawns 1986, 1998, 1999, and 2000 keyword and title files. One reason for this seemingly strange occurrence is that the SGML data files may be divided into year of insertion into the GUIDE database, not the publication date. Another source of such records is late publication of papers originally presented at conferences in years past.

One of the concerns with the original SGML data is the possibility of duplicate records. Despite the convenient separation of records into publication year, elimination of duplicates and inclusion of records from earlier and later insertion years is still a tedious, manual process. For this study, 4063

unique records from 1998 through mid-2001 are, finally, available for analysis.

For the final data preparation, it must be noted that the CAIR system has undergone additional revision since the development of CAIR-Prep and its input data format has changed. The new format uses a SGML style, replacing the earlier *, \#, and \! delimiters with DOC, DOCNO, and TEXT tag-pairs, as shown in Figure 2. It is a simple matter to use a text processor to replace the old-style delimiters with the new SGML-style tags. A sample of the keyword data in the new format is shown in Appendix E.

```
<DOC>
<DOCNO>
  document_number
</DOCNO>
<TEXT>
document_text
</TEXT>
</DOC>
```

Figure 2: CAIR SGML Format

Chapter 4

CO-WORD ANALYSIS

Co-word analysis allows one to reduce a large space of related descriptors to smaller, inter-related spaces that, hopefully, are easier to understand. From the networks generated in this study, various levels of analysis can be performed: (1) as the relationships apparent within networks, (2) as relationships that become obvious from the interaction of networks, and (3) as the transformation of these structures over time [COULTER98B].

4.1 The Metric

In order to form networks (also referred to as leximaps or, simply, maps), there must be a metric (or measurement) used to distinguish between related and

unrelated nodes and also to establish how related any two nodes are. There has been extensive research on metrics for co-word analysis [see CALLON86, COURTIAL89, WHITAKER89, CALLON91, LAW92].

Two descriptors are said to co-occur if they are used together to classify a single document. Consider a corpus of N documents, each indexed by a set of unique descriptors. Let c_k be the number of times descriptor k is used for indexing documents in the corpus. Let c_{ij} be the number of documents in which descriptor i and descriptor j are used together for indexing.

As in the 1998 study by Coulter et al. [COULTER98B], the metric chosen for this study is the strength of the association between descriptor i and descriptor j, S_{ij}. This strength is defined by the expression shown in Figure 3.

$$S_{ij} = \frac{c_{ij}^2}{c_i \cdot c_j}, 0 \le S \le 1$$

Figure 3: Strength of Association

This metric provides an intuitive measure of the symmetrical relationship between the descriptors [CALLON91]. It is also the default metric used by the CAIR system.

4.2 The Algorithm

The co-word analysis algorithm employed in this study uses the strength metric to build networks of related descriptors. This is accomplished with two passes through the data. The first pass, Pass-1, builds the primary associations between descriptors. Descriptors identified during this pass are referred to as "internal nodes" and the links between them are

"internal links." These internal links identify areas of strong association.

Pass-2 identifies links between Pass-1 nodes in one network with Pass-1 nodes in other networks, thus forming the associations between networks. Pass-2 nodes may appear in several networks, where they are referred to as "external nodes," but each one must appear as a Pass-1 node in exactly one network. "External links" highlight associations between the networks produced in Pass-1, and, thus, may indicate more pervasive issues.

Constraints are placed on the network-building process in order to prevent dominance by common pairs of descriptors and also to help break up large networks into more manageable sizes. Consider what would happen if two terms occur infrequently but, when they do occur, they always occur together. Their strength value would be quite large, but the meaning of that strength would have little significance for the study.

Take, for instance, the occurrence of "petri" and "net." These words almost always occur together in titles as "Petri nets," but they may occur in only a handful of documents. Thus, one of the constraints used in this study is to require a minimum co-occurrence value, c_{ij}, before a link can be generated.

Networks can also become cluttered with legitimate nodes and links. One can prevent this cluttering by forcing the generation of a new network when a maximum number of nodes or links is reached. Both node and link constraints are used here. This may seem like a very artificial and arbitrary means of breaking up networks, but a better understanding of the algorithm employed in this study helps to alleviate such concerns.

Pass-1 of the algorithm begins with the link of highest strength. The nodes of this link become starting points for the first network. Additional links and their corresponding nodes are determined

breadth-first and are added to the existing network until one of the constraints (co-occurrence minimum, link maximum, or node maximum) is reached. Once a link and its nodes have been included in a Pass-1 network, they are removed from inclusion in subsequent Pass-1 networks. The next Pass-1 network always begins with the remaining link of highest strength.

Once all the links and nodes have been placed into networks, Pass-2 begins by restoring all Pass-1 nodes to the list of available nodes. Starting with the first Pass-1 network, Pass-2 then builds links between the Pass-1 nodes to Pass-1 nodes in other networks that meet a minimum co-occurrence value and in order of descending strength. After all the Pass-1 nodes in the first network are exhausted, Pass-2 repeats the process for the second Pass-1 network, and so on until all Pass-1 networks have been completed.

Occasionally, some of the links generated in Pass-2 are between Pass-1 nodes within the same network. Such a link is sometimes referred to as a Pass-3 link.

Choosing appropriate constraints can be tricky. Consider the co-occurrence minimum, which, if too high, produces too few links and, if too low, produces an excessive number of links. In the former case, the networks are not granular enough to show important details. In the latter case, the networks may be so complex as to hide important themes.

As with the 1998 study [COULTER98B], parameters in this study are chosen somewhat arbitrarily, and considerable experimentation is done to determine which constraint parameters produce the most useful (i.e., detailed, yet coherent) networks from the current data. Of principal concern is the minimum co-occurrence value, as its effect on the number and complexity of networks produced is less easily determined than node and link count maxima.

Time Period	Documents	Descriptors	Descriptor / Document Ratio
1982 - 1986	1646	5645	3.43
1987 - 1990	7650	28471	3.72
1991 - 1994	7395	23611	3.19
1998 - 2001	4063	15883	3.91

Table 5: Documents and Descriptors per Time Period

The 1998 study examines descriptors for documents from three time periods: 1982 - 1986, 1987 - 1990, and 1991 - 1994. Both the number of documents and the number of descriptors are varied, and, in the case of the earliest period, these numbers are considerably different. Table 5 reproduces these values from both the 1998 study as well as this study. The computed value of the descriptor to document ratio is included, as it may provide some additional insight.

In terms of number of documents, number of descriptors, and descriptor/document ratio, the data of the current period are not significantly different from that of earlier periods. This should mean that this study will see similar effects for changes in minimum co-occurrence value to what was seen in the earlier study.

The 1998 study notes that decreasing the minimum co-occurrence value results in an increase in the number of networks produced. A similar relationship is also seen with the current data set, as shown in Table 6. However, the correlation is not quite linear. Perhaps a future study will determine the mathematical relationships, if there are any, between descriptor-to-document ratio, minimum co-occurrence value, and the number of maps produced.

Min. Co-occurrence	No. of Networks
15	8
10	10
7	15
5	15
3	18

Table 6: Co-occurrence and Number of Keyword Networks

For the portion of this study dealing with the CCS descriptors (keywords) assigned to publications from the 1998 - 2001 time period, a minimum co-occurrence value of seven (7) is chosen. This produces a total of 15 networks.

For the portion of this study dealing with words found in the title text of publications from 1998 - 2001, a minimum co-occurrence value of five (5) produces 16 useable networks, while a value of three (3) increases the number of networks to 24. Hence, a minimum co-occurrence level of five (5) is chosen for the study of titles.

4.3 The CAIR System

The Context Analysis and Information Retrieval (CAIR) system is a series of programs to assist in the analysis of large scale text corpora developed at the Software Engineering Institute, Carnegie Mellon University. The principal developers of this system are Suresh Konda and Ira Monarch.

The CAIR system implements the two-pass algorithm used in this study and provides a graphical user interface with which the produced networks can be manipulated. CAIR also includes tools for analyzing the "internal strengths" and the strengths of the interactions between networks with graphical representations.

4.3.1 CAIR Command-Line Tools

The majority of the CAIR processing takes place at the command-line through the execution of a sequence of programs (outlined in Appendix F). This command-line portion of CAIR processes the input data and produces leximap (LM) output files, which can then be used with the CAIR graphical user interface to generate the graphical network maps that are analyzed in this study.

The lm2 program is the last step before entering the graphical portion of the CAIR system. It is with this program that the network constraints are set, including minimum co-occurrence (c), maximum number of nodes per network (n), maximum number of links per network (l), and maximum number of maps (m). For this study, the number of maps generated is never greater than 30, so setting (m) to a high value

(say, 100) simply has the effect of not excluding any generated maps.

For this study, the (n) and (l) parameters are set to 10 and 12, respectively. Because there are two passes of the algorithm, this has the effect of allowing a maximum of 20 nodes and 24 links per network. These values are chosen to match those of the 1998 study [COULTER98B] and seem to produce maps of reasonable complexity.

4.3.2 CAIR LM File

The CAIR LM files provide a wealth of information about the results of the co-word analysis and the generated maps. The first part of the LM file lists the run parameters, such as the minimum co-occurrence, maximum numbers of links and nodes, and the resulting number of maps. The rest of the file is devoted to describing each of the generated leximaps.

Each leximap description has four parts: header, node list, link list, and summary. The header consists of three numbers: the map number, the number of nodes, and the number of links. For example, if the header is "2 20 24," it means that this is Map-2, which has 20 nodes and 24 links.

Following the header are the nodes that make up the map. Each node and its characteristics appear on a single line. Consider,

^javad.3.2^ 170 3 4 1 2.

In this typical example, the node text (javad.3.2) is delimited by carets. The numbers that follow the node give, respectively, the number of documents in which the node text appears (170), the number of maps in which the node appears (3), the number of links involving the node in the current map (4). The penultimate number (1) tells whether the node is generated during Pass-1 (a '1') or Pass-2 (a '2').

The final number (2) provides the number of the map in which the node is generated during Pass-1.

The next section contains information about the links that make up the leximap with each link starting on a new line. This includes the two linked nodes (delimited by carets), the number of times the nodes appear together, the strength of the link between the nodes, and the pass during which the link was generated (1, 2, or 3). Pass-3 links are just Pass-2 links between Pass-1 nodes in the same map. The final value depends on the pass number of the link; for Pass-1 or Pass-3 links, the final number is 0; and for Pass-2 links, the final number is the map number of the Pass-2 node.

Consider a link description of "^metricsd.2.8^ ^software developmentk.6.3^ 15 0.003805 2 9." In this example, the nodes, "metricsd.2.8" and "software developmentk.6.3," occur together 15 times; the strength of the link between these nodes is 0.003805;

the link is generated during Pass-2; and the Pass-2 node is generated as a Pass-1 node on Map-9.

The fourth section of each leximap description consists of a single line and contains some useful, computed values. From left to right, these values are: cohesion (a measure of the internal strength of the network), the sum of the Pass-2 strengths, and the sum of the squares of the Pass-2 strengths.

4.3.3 CAIR GUI

The next step in using the CAIR system involves entering the graphical user interface component of the system (a program named, "gui"). The CAIR GUI permits the user to view, manipulate, and print the individual leximaps. The CAIR GUI produces two additional graphical outputs: a coupling-cohesion distribution plot and a representation of the supernetworks.

There is something of an art to displaying the maps produced by the CAIR system. Often, the maps are a tangled web of nodes and links. This can make analysis quite difficult. Fortunately, the CAIR system includes a tool to help untangle these webs, called "kamada." Kamada makes a best attempt to reposition nodes to eliminate overlapping links. Some manual repositioning of nodes is still often necessary. Once the maps have been untangled, they may be printed for more detailed analysis.

Two metrics used in the analysis of these networks are cohesion and coupling. Cohesion (also called density) is a measure of the internal strength of a network; it is how strongly the nodes within a network are linked with each other. Cohesion is formally defined as the mean of the Pass-1 link strengths. Coupling (also called centrality) is a measure of how strongly a given network interacts with other networks; it is defined as the square root of the sum of the squares of Pass-2 strengths. Coupling, thus, is a "composite

measure of a network's intersection with all other networks" [COULTER98B].

The CAIR system produces a coupling versus cohesion plot. In this plot, the horizontal axis represents coupling and the vertical axis represents cohesion, with the median values at the origin. Each map appears in this plot as a circle inscribed with its map number, and it is positioned according to its coupling and cohesion values.

Some general comments can be made based on the positions of maps in the coupling-cohesion plot. It is helpful to divide the plot into quadrants, starting with Quadrant-I above and to the right of the axes, and then numbering the quadrants counter-clockwise. Maps in Quadrant-I are characterized by having both strong internal and external interactions. Quadrant-II, above and to the left, is characterized by having strong internal interactions but weak external interactions. Quadrant-III maps are loosely

interactive internally and externally. Quadrant-IV maps are loosely bound internally but strongly interact with other maps.

Quadrant-I maps represent more unitary concepts as well as concepts that interface with many other concepts. This makes Quadrant-I maps especially important in identifying central concepts.

4.4 Naming Networks

The CAIR system numbers the networks it produces, but no other distinguishing notations are provided. Thus, it is useful to assign descriptive names to networks that aid in their correct recognition and in the interpretation of their interactions with other networks.

Shah defines five criteria that can be used to name networks and provides algorithms to simplify the

network naming task [SHAH97]. A less formal application of these algorithms is used for this study. Principally, networks are named in this study by using the one to three nodes with the highest number of Pass-1 links. Exceptions to this rule are allowed when: (1) there is an especially strong link between a chosen node and another Pass-1 node or (2) a Pass-1 node has at least as many Pass-1 and Pass-2 links as a chosen node.

Chapter 5

KEYWORD ANALYSIS

Fifteen networks are generated from the descriptor data using a minimum co-occurrence of seven (7). The CAIR LM file for keywords is reproduced in Appendix G and the resulting leximaps (graphical representations of the networks, often referred to simply as "maps") are provided in Appendix H.

The first step in the analysis is to name the maps. As stated, the name for each map is formed from the text of its prominent node or nodes. For example, Map-4's prominent nodes are "user interfacesd.2.2" and "documentationd.2.7." Thus, the name assigned to Map-4 is "User interfaces / documentation." The names chosen for the maps generated in this study are listed in Table 7.

No.	Assigned Map Name
1	Logic and constraint programming
2	Software development / object-oriented programming
3	Applications / Petri nets / computer-aided engineering
4	User interfaces / documentation
5	Web-based services
6	Distributed systems
7	Performance measures / parallel programming
8	Design tools and techniques
9	Management / metrics
10	Compilers / optimization
11	Software maintenance
12	Language constructs and features
13	Real-time and embedded systems
14	Performance of systems / network protocols
15	Requirements-specifications / testing and debugging

Table 7: Assigned Names for Keyword Maps

5.1 Review of Keyword Maps

Several of the resulting keyword maps might be classified as obvious, redundant, or simply uninteresting. For example, Map-1, named "Logic and constraint programming," contains two nodes: "logic programmingi.2.3" and "logic and constraint programmingf.4.1." The link strength is 0.606811, which is fairly high and indicates that publications classified with one of these descriptors are, more often than not, classified with the other. Such maps, thus, do not provide much useful information.

Some other maps that might be classified as "obvious" include: Map-5 ("Web-based services"), Map-6 ("Distributed systems"), and Map-13 ("Real-time and embedded systems"). The fact that these maps exist is an indication that research in these areas is taking place, but they do not interact much or at all with other areas of software engineering.

Not all such poorly interacting maps are without interest. Often, they serve to highlight important concerns of a given area. Consider Map-7 ("Parallel programming / performance measures") and Map-10 ("Compilers / optimization"). These maps clearly illustrate that performance measures are important in the study of parallel programming and that optimization is still a big concern of compiler design. Similarly, Map-11 ("Software maintenance") shows that restructuring, reverse engineering, and re-engineering are important parts of software maintenance and software development.

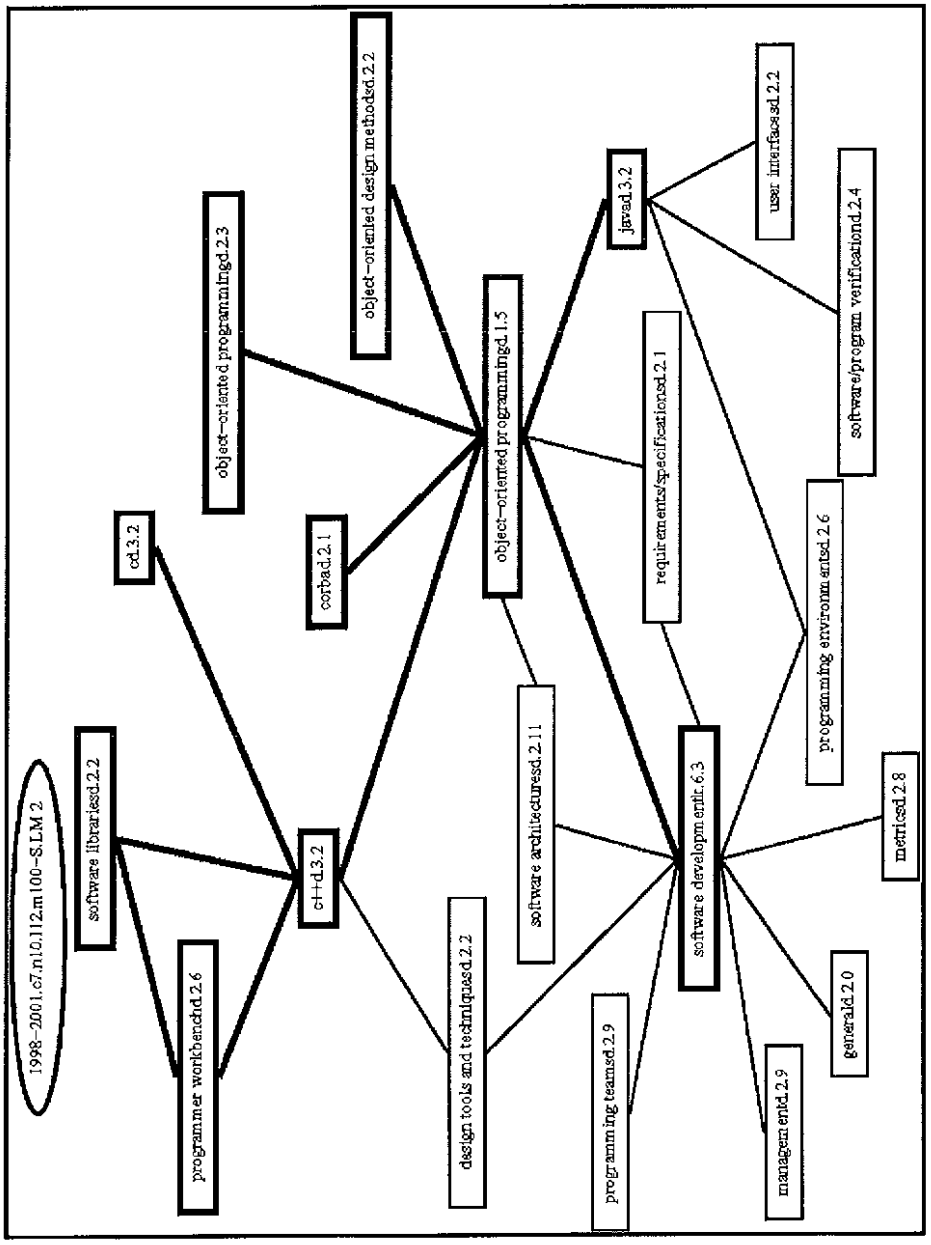


Figure 4: Map-2: "Software development / OOP"

With the largest number of Pass-1 links, the "object-oriented programmingd.1.5" node is clearly the prominent node of Map-2 (see Figure 3). The "software developmentk.6.3" node has a strong link with "object-oriented programmingd.1.5" and has the largest total number of links (Pass-1 and Pass-2). Hence, the name of this map is "Software development / object-oriented programming." Structured programming does not appear as a node in this map, showing the continued prominence of object-oriented programming noted in the earlier study [COULTER98B].

Some other noteworthy observations can be drawn from Map-2. First, the major tools and environments of software development are C++, Java, and CORBA. Second, some basic areas of software development continue to appear in the literature, namely software architectures, requirements and specifications, design tools and techniques, programming environments, metrics, and management. In this case, "management"

may refer to more than just code management, as evidenced by the Pass-2 node, "programming teamsd.2.9." Object-oriented programming techniques naturally lend themselves to team projects.

Map-3 is about computer-aided engineering and manufacturing. Petri nets continue to make an appearance, as they did in the latter of the three periods studied in 1998 [COULTER98B]. Petri nets have "become particularly important in the modeling of automated manufacturing systems" [CHAPMAN97].

Map-4 ("User interfaces and documentation") shows that user interfaces continue to be a focus of research, as they were during the 1987 - 1990 and 1991 - 1994 periods. The appearance of documentation, Java, and parallel programming indicate their importance in the area of user interfaces and human-computer interaction.

The spoke-like pattern of Map-8 centers about "design tools and techniquesd.2.2" and highlights fundamentals as well as some of the prominent, related concerns. The fundamentals of design, such as programming environments, requirements and specifications, testing and debugging, and management are expected to appear in such a map. The concentration on parallel and concurrent programming during this period is interesting to note as is the appearance of engineering and the physical sciences.

Map-9 appears to have two prominent nodes, "generald.2.0" and "managementd.2.9." Management has also appeared as a prominent node in networks of the 1982 - 1986, 1987 - 1990, and 1991 - 1994 time periods. Its appearance in this data set is not surprising, nor is the appearance of metrics. This map may indicate interest in formalizing the software management process.

The general category is included in the CCS at first and second levels for two purposes: to classify documents that include broad treatments of a topic and to classify documents that cover several related topics in the same category. As expected, then, the "generald.2.0" node is linked with a number of issues important to software engineering: computer-aided engineering, algorithm design and analysis, software development, user/machine systems, software management, computer science education, and curriculum concerns.

Map-9 also shows links between the general categories of software engineering, computer communication networks, logics and meanings of programs, and legal aspects of computing. The appearance of these general nodes instead of others may indicate the current, prominent research pursuits of software engineering.

Map-15 ("Requirements-specifications / testing and debugging") outlines the software development process, from defining requirements and specifications to algorithm design and analysis to testing and debugging to distribution, maintenance, and enhancement.

Further analysis of the keyword maps is made through an examination of how the maps interact with each other. To aid with this examination, two graphs, a coupling-cohesion plot and a supernetwork plot, are presented in Appendix I.

5.2 Keyword Network Cohesion and Coupling

The coupling-cohesion plot for the keyword data of this study (see Figure 4) holds no real surprises. In the plot, most maps appear on or near the horizontal (coupling) axis, meaning that there is little difference in the internal strengths (cohesion) of the various maps; the obvious exceptions are Map-1 and, to

a lesser extent, Map-5. Also, there is a clear division between the weakly interacting maps (to the left of the vertical axis) and the more strongly interacting maps (to the right).

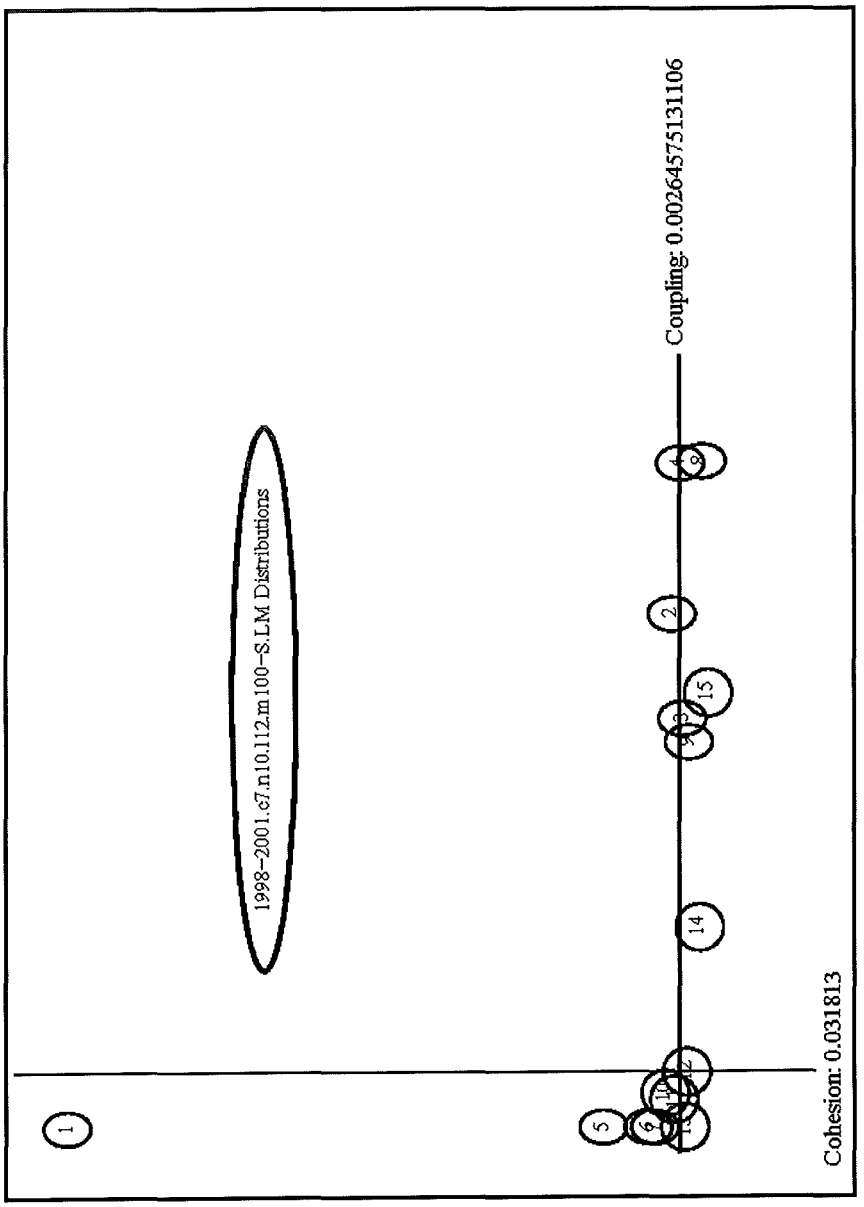


Figure 5: Coupling-Cohesion Plot for Keyword Data

The most interesting networks are the ones that appear in Quadrant-I of the coupling-cohesion plot, as these networks are both tightly bound internally and interact strongly with other networks. Map-2 is the only map to fall within Quadrant-I, which attests to the centrality of software development and object-oriented programming to software engineering research publications during the period of the study. Software development and object-oriented programming appear strongly during the 1991 - 1994 study as well.

Central concepts are often found in strongly interacting maps. A map's coupling value is a measure of its interaction with other maps. Map-4 and Map-8 have the highest coupling values of this study, which is represented by their positions in the coupling-cohesion plot. It is really no surprise that "user interfaces / documentation" and "design tools and techniques" should be central to software engineering.

Also of high centrality are Map-15 ("requirements-specifications / testing and debugging"), Map-3 ("applications / Petri nets / computer-aided engineering"), and Map-9 ("management / metrics"). Again, this is not surprising, but it helps reinforce the correctness of this interpretation.

It is interesting to note the centrality of Map-14 ("performance of systems / network protocols"), which is not as great as, say Map-9, but is still greater than the median. The concepts of Map-14 are not seen in the 1998 study, so this may indicate the growing importance of network protocols and performance of systems to software engineering.

Map-1 ("Logic and constraint programming") appears high in Quadrant-II; this means that it is strongly cohesive but interacts weakly, if at all, with other maps. In fact, Map-1 is completely isolated (its coupling value is zero), which can be confirmed by noting the absence of Pass-2 links. The intuitive

explanation for Map-1's position is that the descriptors, which are the nodes of this map, are so similar that publications indexed with one are almost always indexed with the other. Other than noting the existence of research writing in the area of logic and constraint programming, Map-1 is of little interest.

Map-5 also has a high cohesion value and appears higher in the plot than the majority of the other maps, though not as high as Map-1. Its nodes, "web-based servicesh.3.5" and "web-based interactionh.5.3," clearly have a great similarity and frequently occur together. In addition to noting the existence of web-based services in the literature, Map-5 also shows the rapid incorporation of new descriptors, such as "web-based servicesh.3.5," by indexers. This indicates the importance of regular review and updating of the CCS to maintain its relevance.

Maps-6, 7, 10, 11, 12, and 13 are clustered near the origin of the coupling-cohesion plot. Although these

maps are not tightly bound and do not interact strongly with other maps, they still represent some importance in software engineering; consider the continued importance of compilers and optimization (Map-10).

5.3 Keyword Supernetwork Analysis

Two networks are said to interact with each other when a Pass-1 node in one map appears as a Pass-2 node in another. An indication of the strength between two interacting networks might be the number of such links. Consider, for instance, Map-2, which has three Pass-2 nodes from Map-4, four from Map-8, four from Map-9, and three from Map-15.

Table 8 lists all the connections between the maps generated from the keyword data of this study. From the table, it is clear that Maps-1, 5, 6, 7, and 13 are isolated. Maps-10, 11, and 12 are very weakly

interacting, as they each only have one external link. Map-14 is only slightly more interacting with its two links. This leaves Maps-2, 3, 4, 8, 9, and 15 as significant players in a supernetwork generated from these smaller networks.

Map No.	Connected Maps [Map No.(number of links)]
1	None
2	4(3) 8(4) 9(4) 15(3)
3	4(1) 8(2) 9(1) 14(1) 15(3)
4	2(4) 3(1) 8(6) 9(1) 15(1)
5	None
6	None
7	None
8	2(2) 3(1) 4(4) 9(1) 12(1) 15(4)
9	2(6) 3(1) 4(1) 8(2) 14(1) 15(3)
10	8(1)
11	2(1)
12	8(1)
13	None
14	3(1) 9(1)
15	2(4) 3(3) 4(1) 8(5) 9(3)

Table 8: Connections between Keyword Maps

Figure 5 shows one possible supernetwork based on the data of Table 8. In this case, a threshold of three or more connections is required to show the link. The circles represent maps with the indicated map numbers. Connections between maps are shown with arrows and are labeled with the number of connections. An arrowhead indicates the map in which the link node is Pass-1. Thus, for example, Map-15 contains four (4) Pass-2 nodes that appear as Pass-1 nodes in Map-2.

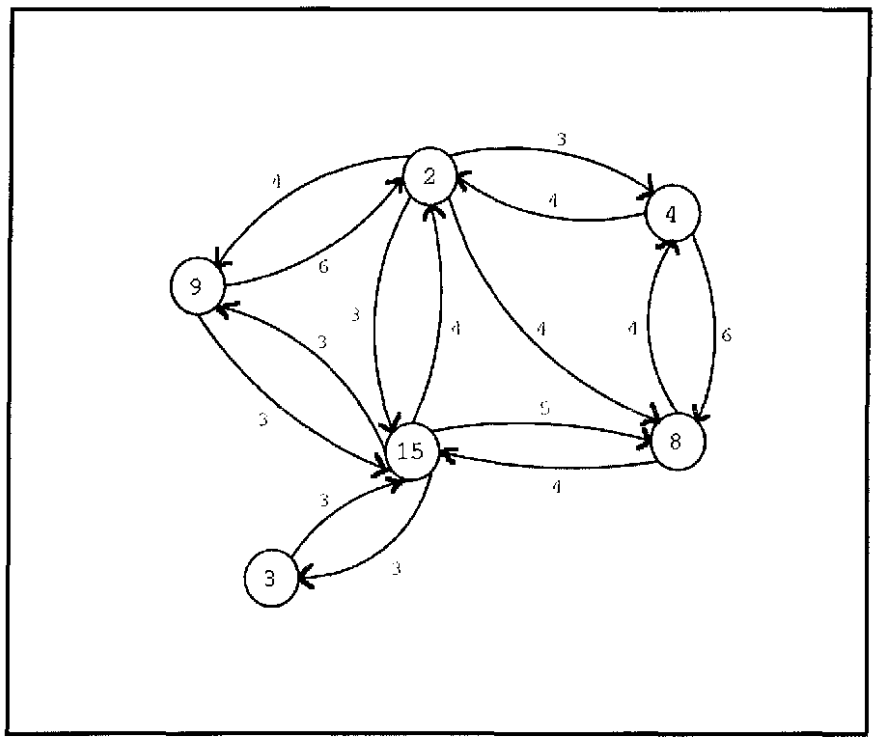


Figure 6: Supernetwork for Keyword Data

There is no single focus to this supernetwork, though Map-2 ("Software development / object-oriented programming") and Map-15 ("Requirements-specifications / testing and debugging") have the highest numbers of connections. This attests to the prominence of these topics in the field of software engineering during the period of this study, and it reinforces the earlier interpretation of the coupling-cohesion plot.

Some note should also be made of Map-3 ("Applications / Petri nets / computer-aided engineering"). Petri nets appear in the 1991 - 1994 period of the 1998 study as an isolated network. In the current study, however, Petri nets have links, directly and indirectly, to "User interfaces / documentation," "Design tools and techniques," "Management / metrics," "Performance of systems / network protocols," and "Requirements-specifications / testing and debugging." Clearly, Petri nets have become more central to software engineering during the 1998 - 2001 period.

Chapter 6

THEMES AND TRENDS

From the preceding analysis, it is clear that software engineering continues to lack a central focus, though there are a number of areas of concentration (or themes). Software engineering continues to evolve as a field: it is incorporating new themes, maintaining others, and dropping still others. Software engineering is defined both by its central (or core) themes as well as its emerging interests.

In this study, the enormous volume of software engineering publications from 1998 through 2001 is reduced to a collection of fifteen networks that represent the themes of the field. Some themes are self-contained and have not yet developed past an emerging interest, such as web-based services and

distributed systems. Others are mature themes that exhibit limited interaction with others, like logic / constraint programming and compilers / optimization. Still other themes are found to interact strongly with many other themes, such as design tools and techniques, user interfaces, and software development.

There is some consistency in the networks generated for this study and those of the 1998 study. Software development, design tools and techniques, and user interfaces, for example, recur in each of the time periods of these studies. This is due in large part to the fixed taxonomy of the CCS, but it also provides some assurance of the correctness of this taxonomy in representing the core themes of software engineering.

The 1998 study [COULTER98B] notes a trend in software development toward large-scale environments. This trend is evidenced in the current study by the prominence of "programming-in-the-large" issues, tools, and techniques, such as object-oriented

programming, project and people management, documentation, and software maintenance.

The incorporation of "relevant supporting tools" into a theme provides some gauge of the "maturity" of a trend [COULTER98B, page 1222]. As a trend matures, specific tools will appear as implicit descriptors. The implicit descriptors that represent specific object-oriented programming tools, such as C++, Java, and CORBA, do appear in the networks of this study. Additionally, the appearance of compilers / optimization and language constructs / features may indicate continued work on incorporating the object-oriented paradigm into the software engineering field.

As one might expect with an increase in programming-in-the-large issues, there is also an apparent increase in interest in best practices and process improvement. This is evidenced by many of the same keywords related to programming-in-the-large, such as

"management," "testing and debugging," "metrics," "reliability," and "program verification."

Some new trends can also be seen. For instance, Petri nets, which appear in the 1991 - 1994 period as an isolated network, have resurfaced in a connected network in the 1998 - 2001 period. Petri nets are commonly used in modeling automated manufacturing systems. As software engineering principles are applied to computer-aided engineering and manufacturing, it is not surprising to see links to other themes of software engineering, such as "requirements and specifications" and "design tools and techniques."

One strong theme in software engineering is the emphasis on parallelism and concurrency. Descriptors related to parallelism and concurrency can be seen in all four periods, but seem fairly ubiquitous in the period of this study. For instance, parallelism-related descriptors appear in Map-3 ("Applications /

Petri nets / computer-aided engineering"), Map-4 ("User interfaces / documentation"), Map-7 ("Performance measures / parallel programming"), Map-8 ("Design tools and techniques"), and Map-15 ("Requirements-specifications / testing and debugging").

The 1998 revision of the CCS includes over 225 new subject descriptors [see COULTER98A]. Many of these new terms are related to distributed and online systems, including the World-Wide Web. It is interesting to note the appearance of these terms in the 1998 - 2001 period, which indicates that the GUIDE's indexers found immediate need for these terms. This is a clear indication that periodic review and revision of the CCS is required for it to remain relevant.

It is also interesting to note the disappearance from the current period of the graphical user interfaces of Windows and X-Windows, which had appeared in the 1991

- 1994 period. Perhaps, this is an additional indication of the trend toward online systems and the use of the web browser as the user-interface of choice.

Chapter 7

TITLE ANALYSIS

Unlike earlier studies, this study has access to the title text for most of the publications in the GUIDE for the period 1998 - 2001. This allows a look at the descriptive text chosen by the authors to represent the topics of their published works. This may provide corroboration of the results of the keyword analysis and offer insight into the relevance and currency of the CCS.

4063 titles are available for this analysis after parsing the original SGML data. Some of these titles are journal names, such as IEEE Transactions on Software Engineering and Journal of Software Maintenance. The incorporation of these titles into this analysis skews the generated maps, simply because these terms occur together more frequently.

Another concern is that there is no fixed taxonomy to limit word choice, and, in some cases, the co-occurrence of related terms may be diluted below the threshold required to produce a link. Thus, important, related terms may not appear in the final maps.

7.1 The Title Data

The CAIR "check" command generates an index of terms parsed from the input text. These terms form the nodes of maps generated in later stages of the CAIR analysis process. The "check" command's "-t" parameter sets a threshold value for clustering. This parameter is set to five, meaning that a word must appear five times to qualify as a term. A higher threshold can reduce the noise of less important words, but there seems to be little to gain from such a reduction in the current data set.

The title terms consist of common nouns, such as "window" and "technique," proper nouns, such as "Java" and "Linux," and compound nouns, such as "software engineering" and "object-oriented programming." The CAIR system parses 485 terms from the title data. In comparison, 366 terms are parsed from the keyword data of the same period. The similarity of these numbers implies that word choice, at least with respect to software engineering titles, is not as unrestricted as it might seem.

Appendix J reproduces a portion of the title index file sorted in order of decreasing frequency. The most common terms ("software," "analysis," and "programming") are expected, considering the subject matter. Some term frequencies may be artificially inflated through their appearance in compound terms. For instance, "software" appears alone and in combination, such as "software engineering," "software development," "object-oriented software," and so on.

Some additional term frequency inflation is due to the repeated appearance of journal titles in the data, such as IEEE Transactions on Software Engineering and Communications of the ACM. Since these journals contain published articles on a wide variety of topics, the inclusion of the journal title for each issue, necessarily, skews analysis results toward the words occurring in these titles.

7.2 CCS General Terms

The CCS includes sixteen General Terms that may be associated with any category. It should be expected that these General Terms are represented in the titles. In fact, most of the General Terms, like "Design" and "Performance," are found verbatim in the index of title terms.

Other General Terms are represented by proxy. For instance, "Experimentation" is represented by a number

of closely related or synonymous terms, like "study," "testing," and "empirical study." Likewise, the General Term, "Economics," does not appear in the title terms, but "business," "cost," and "business process" do.

Table 9 lists the General Terms and their frequencies in the title data. Where appropriate a proxy and its frequency is listed in parentheses. It is interesting that "Legal Aspects" and its potential proxies, such as "law" and "liability," do not appear frequently enough be included in the index file.

119 Design	30 Languages
0 Experimentation (96 Study)	0 Standardization (26 Standard)
86 Performance	25 Measurement
64 Verification	21 Reliability
55 Management	18 Security
40 Documentation	12 Algorithms
0 Economics (35 Business)	10 Theory
0 Human Factors (33 User Interface)	0 Legal Aspects

Table 9: General Terms and Their Frequencies in the Title Data

7.3 Themes from the Title Index

The most frequent terms, such as "software," "analysis," "programming," "design," and "engineering," are those that pervade the software engineering field. These terms are clearly important to the field, but do not tell much about the current emphasis or trends in research.

One theme appearing clearly in the index of title terms involves process improvement and best practices. This is seen in the pervasiveness of terms like "performance," "evaluation," "management," "case study," "practice," "quality," "documentation," "business," "process," "optimization," "debugging," "improvement," and many more.

Proper nouns, like "Java," "C++," and "CORBA," appear with high frequencies, as do other terms, like "object," "object-oriented software," "object-oriented programming," and "software reuse." These terms confirm the emphasis on object-oriented programming (OOP) highlighted by the keyword analysis. Together with the process improvement theme, OOP, hints at another theme revealed by the keyword analysis: large-scale software development.

The trend toward online systems, which the keyword analysis highlights, is also apparent from the titles. Terms, like "communication," "Internet," "hypermedia,"

"network," and "web," appear frequently enough to be added to the index of title terms. The corroboration of this new trend also confirms the usefulness of the new, "online" descriptors added to the CCS in 1998.

7.4 Title Networks

There are considerable differences between the keyword and title data sets, not the least of which is the lack of a fixed taxonomy. Nevertheless, some understanding of the represented publications can be gained by performing an analysis of the CAIR-rendered title maps.

Minimum Co-occurrence (c)	Number of Maps
3	21
5	16
7	8
10	3

Table 10: Co-occurrence and Number of Title Networks

As with the keyword analysis, the choice of parameters for the CAIR system is somewhat arbitrary. If the co-occurrence minimum is too low, then too many links are produced and details are hidden in the complexity of the generated maps. If the co-occurrence minimum is too high, then too few links are produced and important relationships are missed. Table 10 shows the effect on the number of generated maps by the choice of minimum co-occurrence value. A minimum co-occurrence of five (5) produces networks comparable in number to those created for the keyword analysis, so this value is chosen for the analysis.

CAIR generates sixteen maps to represent the title data. The resulting LM file can be found in Appendix K and the maps themselves are reproduced in Appendix L. Coupling-cohesion and supernetwork plots are also generated and can be found in Appendix M.

The title maps are named with the same flexible naming convention used for the keyword maps; that is, with few exceptions, the names are taken from the most prominent, Pass-1 nodes. Table 11 lists the assigned title map names.

No.	Assigned Map Name
1	Interaction - Detection
2	TCL - TK
3	Exception - Handling
4	Client - Server
5	Analysis - Performance
6	Effort - Estimation
7	Software Process - Improvement
8	Software Engineering
9	Report - Experience
10	Software Reliability
11	Project - Management
12	Application - Development
13	Comparison - Technique
14	User Interface
15	Program - Verification
16	Method - Tool

Table 11: Assigned Names for Title Maps

Many of the maps generated from the title data are "obvious." That is, not much in the way of substantial meaning can be derived from them. For instance, Map-2 ("TCL - TK") contains two nodes, "tcl" and "tk," and does not interact with any other maps. The nodes of this map refer to the scripting language, TCL, and its graphical toolkit, Tk. These two software development tools are almost always used together, which explains their link strength of 0.694444.

Maps-1, 3, 4, 6, 7, 9, 10, 13, and 14 also likely would be labeled, "obvious" or "uninteresting." All of these maps are isolated, except Map-13 ("Comparison - Technique"), whose one Pass-2 link associates the nominal nodes with the obviously related node, "analysis." Map-16 ("Method - Tool") has a moderate coupling value, likely only because methods and tools are concerns of many aspects of software development.

The remaining maps, 5, 8, 11, 12, 15, and 16, have high coupling values and may be considered more interesting. Map-5 ("Analysis - Performance") represents primary concerns of software engineering. Notable is the appearance of Petri net, a modeling tool often used in computer-aided manufacturing, which is also seen in the keyword analysis.

Map-8 ("Software Engineering") is clearly skewed by the journal title, IEEE Transactions on Software Engineering. This map has the highest coupling value, which is not unexpected, given the purview of this journal. Map-8 is strongly coupled with Map-11 ("Project - Management") through the "software" node. Map-11 illustrates one of the trends in software development noted in both the 1998 study [COULTER98B] and the keyword analysis of the current study: the trend toward "programming-in-the-large" and the related concern of "best practices."

Map-12 ("Application - Development") has the second highest coupling value and represents another core concern of software engineering. "Internet" and "network" appear in this map, along with real-time systems, hinting at the trend toward online services also noted in the keyword analysis.

Map-15 ("Program - Verification") is not very interesting at first glance. Its high coupling value is clearly due to the pervasive nature of programming in software engineering. The appearance of "2nd ed" reflects the relatively high frequency of second edition programming texts. There were also a small number of third edition works, but not enough to appear in a map.

It is important to note that nothing in the title maps stands out as discordant with the keyword analysis of the same publications. The major themes of large-scale software development, process improvement, and even the trend toward online systems are seen in the

title maps. This lends some credence to the results of the keyword analysis and the relevance of the recent additions to the CCS.

This analysis of the titles provides some corroboration for the keyword analysis, but titles are not necessarily the best indicators of content. The abstracts, review texts, and the texts of the publications themselves would provide a better source of data for analysis.

CONCLUSIONS

Current themes and trends in software engineering can be determined through analysis of its recent research publications. This study applies co-word analysis techniques to publications reviewed in the Association for Computing Machinery's Guide to Computing Literature (GUIDE) for the 1998 - 2001 period with the goal of revealing these themes and trends.

The first part of this study looks at the descriptors (or keywords) assigned to publications by the GUIDE's indexers. Descriptors are taken from the fixed taxonomy of the Computing Classification System (CCS). This analysis extends a 1998 study of the GUIDE descriptor data from the three periods, 1982 - 1986, 1987 - 1990, and 1991 - 1994. The 1998 - 2001 period provides several advantages: it includes the most recently available data, its volume is comparable to

that of the earlier study, and all the data conform to the last CCS revision.

The second part of this study applies co-word analysis to the titles of the published works reviewed in the GUIDE during this same period. Examination of the titles reveals the same themes shown by the analysis of descriptors, providing some corroboration of both the results and the analysis techniques.

Software engineering has no central focus, but the themes of software development, process improvement, applications, parallelism, and user interfaces are persistent and help define the field. Trends in the field are more useful as guidance for research and curriculum development. The prominent trends revealed by this study include increased interest in large-scale software development or programming-in-the-large, best practices, and distributed and online computing.

The interest in best practices is a natural consequence of large-scale projects, where planning, management, and review take on special importance. Also reflective of programming-in-the-large is the prominent appearance of object-oriented programming (OOP) and its related tools and techniques. The OOP paradigm naturally lends itself to these large-scale projects, and this may be seen as support for its incorporation into academic curricula.

Distributed and online computing, especially with regard to the Internet and the World-Wide Web, has become a major interest of software engineering. Distributed computing is not new to software engineering, nor is the Internet, but the GUIDE's indexers found immediate use for the newly added Internet-related descriptors. Furthermore, the disappearance from the current data of descriptors related to Windows and X-Windows may indicate a trend toward online software systems that use the web browser as the user-interface of choice.

Many of the descriptors added in the 1998 revision of the CCS found immediate use in classifying recent publications. A clear conclusion from this is that periodic review and revision of the CCS is appropriate, if not required, for it to remain relevant.

This study successfully extends to the current period an earlier analysis of software engineering publications through their assigned CCS descriptors. This study also includes an analysis of the titles of these same publications, providing both the corroboration of the descriptor analysis and some insight into the appropriateness and relevance of the CCS to the current period.

APPENDIX A

The Top Two Levels of the CCS (1998)

- A. General Literature
  - A.0 GENERAL
  - A.1 INTRODUCTORY AND SURVEY
  - A.2 REFERENCE (e.g., dictionaries, encyclopedias, glossaries)
  - A.m MISCELLANEOUS
- B. Hardware
  - B.0 GENERAL
  - B.1 CONTROL STRUCTURES AND MICROPROGRAMMING (D.3.2)
  - B.2 ARITHMETIC AND LOGIC STRUCTURES
  - B.3 MEMORY STRUCTURES
  - B.4 INPUT/OUTPUT AND DATA COMMUNICATIONS
  - B.5 REGISTER-TRANSFER-LEVEL IMPLEMENTATION
  - B.6 LOGIC DESIGN
  - B.7 INTEGRATED CIRCUITS
  - B.8 PERFORMANCE AND RELIABILITY (C.4)
  - B.m MISCELLANEOUS
- C. Computer Systems Organization
  - C.0 GENERAL
  - C.1 PROCESSOR ARCHITECTURES
  - C.2 COMPUTER-COMMUNICATION NETWORKS
  - C.3 SPECIAL-PURPOSE AND APPLICATION-BASED SYSTEMS (J.7)
  - C.4 PERFORMANCE OF SYSTEMS
  - C.5 COMPUTER SYSTEM IMPLEMENTATION
  - C.m MISCELLANEOUS
- D. Software
  - D.0 GENERAL
  - D.1 PROGRAMMING TECHNIQUES (E)
  - D.2 SOFTWARE ENGINEERING (K.6.3)
  - D.3 PROGRAMMING LANGUAGES
  - D.4 OPERATING SYSTEMS (C)
  - D.m MISCELLANEOUS
- E. Data

- E.0 GENERAL
- E.1 DATA STRUCTURES
- E.2 DATA STORAGE REPRESENTATIONS
- E.3 DATA ENCRYPTION
- E.4 CODING AND INFORMATION THEORY (H.1.1)
- E.5 FILES (D.4.3, F.2.2, H.2)
- E.m MISCELLANEOUS

- F. Theory of Computation
  - F.0 GENERAL
  - F.1 COMPUTATION BY ABSTRACT DEVICES
  - F.2 ANALYSIS OF ALGORITHMS AND PROBLEM COMPLEXITY (B.6, B.7, F.1.3)
  - F.3 LOGICS AND MEANINGS OF PROGRAMS
  - F.4 MATHEMATICAL LOGIC AND FORMAL LANGUAGES
  - F.m MISCELLANEOUS
- G. Mathematics of Computing
  - G.0 GENERAL
  - G.1 NUMERICAL ANALYSIS
  - G.2 DISCRETE MATHEMATICS
  - G.3 PROBABILITY AND STATISTICS
  - G.4 MATHEMATICAL SOFTWARE
  - G.m MISCELLANEOUS
- H. Information Systems
  - H.0 GENERAL
  - H.1 MODELS AND PRINCIPLES
  - H.2 DATABASE MANAGEMENT (E.5)
  - H.3 INFORMATION STORAGE AND RETRIEVAL
  - H.4 INFORMATION SYSTEMS APPLICATIONS
  - H.5 INFORMATION INTERFACES AND PRESENTATION (e.g., HCI) (I.7)
  - H.m MISCELLANEOUS
- I. Computing Methodologies
  - I.0 GENERAL
  - I.1 SYMBOLIC AND ALGEBRAIC MANIPULATION
  - I.2 ARTIFICIAL INTELLIGENCE
  - I.3 COMPUTER GRAPHICS
  - I.4 IMAGE PROCESSING AND COMPUTER VISION
  - I.5 PATTERN RECOGNITION
  - I.6 SIMULATION AND MODELING (G.3)
  - I.7 DOCUMENT AND TEXT PROCESSING (H.4, H.5)
  - I.m MISCELLANEOUS

- J. Computer Applications
  - J.0 GENERAL
  - J.1 ADMINISTRATIVE DATA PROCESSING
  - J.2 PHYSICAL SCIENCES AND ENGINEERING
  - J.3 LIFE AND MEDICAL SCIENCES
  - J.4 SOCIAL AND BEHAVIORAL SCIENCES
  - J.5 ARTS AND HUMANITIES
  - J.6 COMPUTER-AIDED ENGINEERING
  - J.7 COMPUTERS IN OTHER SYSTEMS (C.3)
  - J.m MISCELLANEOUS
- K. Computing Milieux
  - K.0 GENERAL
  - K.1 THE COMPUTER INDUSTRY
  - K.2 HISTORY OF COMPUTING
  - K.3 COMPUTERS AND EDUCATION
  - K.4 COMPUTERS AND SOCIETY
  - K.5 LEGAL ASPECTS OF COMPUTING
  - K.6 MANAGEMENT OF COMPUTING AND INFORMATION SYSTEMS
  - K.7 THE COMPUTING PROFESSION
  - K.8 PERSONAL COMPUTING
  - K.m MISCELLANEOUS

APPENDIX B

Sample SGML Data Set

```
<STARTREC>
<PUBTYPE>JOURNAL ARTICLE     </PUBTYPE>
<TITLE>
Toward formalizing structured analysis
</TITLE>
<AUTHEDIT>
Baresi, Luciano
</AUTHEDIT>
<AUTHTYPE>AUTHOR     </AUTHTYPE>
<AUTHEDIT>
Pezz&egrave;, Mauro
</AUTHEDIT>
<AUTHTYPE>AUTHOR     </AUTHTYPE>
<GENTERM>PERFORMANCE                    </GENTERM>
<GENTERM>DOCUMENTATION                  </GENTERM>
<GENTERM>MEASUREMENT                    </GENTERM>
<GENTERM>THEORY                         </GENTERM>
<GENTERM>DESIGN                         </GENTERM>
<KEYWORD>STRUCTURED ANALYSIS/REAL-TIME
</KEYWORD>
<KEYWORD>INFORMAL VERSUS FORMAL SPECIFICATIONS
</KEYWORD>
<KEYWORD>HATLEY AND PIRBHAI'S REQUIREMENTS DEFINITION
NOTATION            </KEYWORD>
<PRICATDESC>
Software,
SOFTWARE ENGINEERING,
Requirements/Specifications,
Methodologies (e.g., object-oriented, structured)
</PRICATDESC>
<PRICATCODE>               D.2.1</PRICATCODE>
<DESCRIPTOR>
Software,
SOFTWARE ENGINEERING,
Coding Tools and Techniques,
Structured programming
```

```
</DESCRIPTOR>
<CATCODE>                D.2.3</CATCODE>
<PUBYEAR>1998</PUBYEAR>
<JRLNAME>
ACM Transactions on Software Engineering and
Methodology
</JRLNAME>
<ABSTRACT>
<par>Real-time extensions to structured analysis
(SA/RT) are popular in industrial practice. Despite
the large industrial experience and the attempts to
formalize the various &ldquo;dialects,&rdquo; SA/RT
notations are still imprecise and ambiguous. This
article tries to identify the semantic problems of the
requirements definition notation defined by Hatley and
Pirbhai, one of the popular SA/RT
&ldquo;dialects,&rdquo; and discusses possible
solutions. As opposed to other articles that give
their own interpretation, this article does not
propose a specific semantics for the notation. This
article identifies imprecisions, i.e., missing or
partial information about features of the notation; it
discusses ambiguities, i.e., elements of the
definition that allow at least two different
(&ldquo;reasonable&rdquo;) interpretations of features
of the notation; and it lists extensions, i.e.,
features not belonging to the notation, but required
by many industrial users and often supported by CASE
tools. This article contributes by clarifying whether
specific interpretations can be given unique semantics
or retain ambiguities of the original definition. The
article allows for the evaluation of formal
definitions by indicating alternatives and
consequences of the specific choices.</par>
</ABSTRACT>
</STARTREC>
```

APPENDIX C

Sample CAIR-Prep Keyword Data

```
\*
\#
1998;1
\#
\!
-1 (petri netsd.2.2) () 0
-1 (assertion checkersd.2.4) () 0
-1 (mechanical verificationf.3.1) () 0
-1 (hypertext/hypermediai.7.2) () 0
-1 (hypertext/hypermediah.5.4) () 0
\!
\*
\*
\#
1998;2
\#
\!
-1 (generalk.3.0) () 0
-1 (generalj.0) () 0
-1 (interoperabilityd.2.12) () 0
\!
\*
\*
\#
1998;3
\#
\!
-1 (standardsk.1) () 0
-1 (standardsd.2.0) () 0
\!
\*
```

APPENDIX D

Sample CAIR-Prep Title Data

```
\*
\#
1998;1
\#
\!
Hyperdocuments as automata: verification of trace-
     based browsing properties by model checking
\!
\*
\*
\#
1998;2
\#
\!
(v.41 n.1) Communications of the ACM
\!
\*
\*
\#
1998;3
\#
\!
Corporate shortcut to standardization
\!
\*
```

APPENDIX E

Sample Keyword Data with SGML-style Tags

```
<DOC>
<DOCNO>
1998;1
</DOCNO>
<TEXT>
-1 (optimizationd.3.4) () 0
-1 (algorithm design and analysisg.4) () 0
-1 (requirements/specificationsd.2.1) () 0
-1 (lambda calculus and related systemsf.4.1) () 0
</TEXT>
</DOC>
<DOC>
<DOCNO>
1998;2
</DOCNO>
<TEXT>
-1 (design tools and techniquesd.2.2) () 0
-1 (language classificationsd.3.2) () 0
-1 (operational semanticsf.3.2) () 0
</TEXT>
</DOC>
<DOC>
<DOCNO>
1998;3
</DOCNO>
<TEXT>
-1 (object-oriented programmingd.1.5) () 0
-1 (reusable softwared.2.13) () 0
-1 (modules and interfacesd.2.2) () 0
-1 (distribution, maintenance, and enhancementd.2.7)
     () 0
</TEXT>
</DOC>
```

APPENDIX F

CAIR Processing Sequence

The CAIR system implements the two-pass co-word analysis algorithm at the command-line. The sequence of commands is illustrated by the steps presented below.

The before_tagger, tagger, and reg_exp_parser are used to prepare free text for co-word analysis. Part of this process involves parsing nouns and noun phrases from the input text. These nouns and noun phrases form the keywords for which co-occurrence metrics are computed. This part of the process is required when analyzing the title text, but the keyword data of this study (see Appendix E) are already in the ".parse" format.

The remaining steps perform counts of terms, compute strengths and co-occurrences, and generate the leximap

(LM) files used by the graphical portion of the CAIR system.

1. before_tagger < sample.prep > sample.pretag
2. tagger < sample.pretag > sample.tag
3. reg-exp-parser < sample.tag > sample.parse
4. clust1 < sample.parse
5. sort_files
6. clust2
7. check -t 0 -l 5 > sample.index
   ['-l' is a lowercase '-L']
8. lm1 -v < sample.index > sample.LMDB
9. lm2 -c 7 -n 10 -l 12 -m 100 -S < sample.LMDB > sample.c7.n10.l12.m100-S.LM

APPENDIX G

CAIR LM File for Keywords

Run Parameters: Eliminate by Nodes

Pass Two Node Filter: Both nodes. Link Selection: Strength and Max. Nodes

Min. Strength: 0.000000. Min. Co-Occurrence: 7. Max links: 12.

Max maps 100. Max nodes 10. Maps Produced: 15

1 2 1

^logic and constraint programmingf.4.1^ 19 1 1 1 1

^logic programmingi.2.3^ 17 1 1 1 1

^logic and constraint programmingf.4.1^ ^logic programmingi.2.3^ 14 0.606811 1 0

0.606811 0.000000 0.000000 20 14

2 20 24

^software developmentk.6.3^ 384 6 9 1 2

^object-oriented programmingd.1.5^ 287 3 8 1 2

^c++d.3.2^ 112 1 5 1 2

^javad.3.2^ 170 3 4 1 2

^programmer workbenchd.2.6^ 26 1 2 1 2

^requirements/specificationsd.2.1^ 306 4 2 2 15

^design tools and techniquesd.2.2^ 692 8 2 2 8

^programming environmentsd.2.6^ 218 3 2 2 4

^software architecturesd.2.11^ 160 3 2 2 8

^software librariesd.2.2^ 96 1 2 1 2

^user interfacesd.2.2^ 182 2 1 2 4

^cd.3.2^ 33 1 1 1 2

^object-oriented design methodsd.2.2^ 58 1 1 1 2

^object-oriented programmingd.2.3^ 19 1 1 1 2

^corbad.2.1^ 48 1 1 1 2

^managementd.2.9^ 278 4 1 2 9

^programming teamsd.2.9^ 30 2 1 2 9

^generald.2.0^ 498 6 1 2 9

^metricsd.2.8^ 154 2 1 2 9

^software/program verificationd.2.4^ 218 3 1 2 15

^programmer workbenchd.2.6^ ^software librariesd.2.2^ 24 0.230769 1 0

^object-oriented design methodsd.2.2^ ^object-oriented programmingd.1.5^ 29 0.050523 1 0

^c++d.3.2^ ^software librariesd.2.2^ 17 0.026879 1 0

^c++d.3.2^ ^programmer workbenchd.2.6^ 8 0.021978 1 0

```
^object-oriented programmingd.1.5^ ^object-oriented
programmingd.2.3^ 10 0.018339 1 0

^c++d.3.2^ ^cd.3.2^ 7 0.013258 1 0

^javad.3.2^ ^object-oriented programmingd.1.5^ 20
0.008198 1 0

^corbad.2.1^ ^object-oriented programmingd.1.5^ 10
0.007259 1 0

^c++d.3.2^ ^object-oriented programmingd.1.5^ 15
0.007000 1 0

^object-oriented programmingd.1.5^ ^software
developmentk.6.3^ 26 0.006134 1 0

^managementd.2.9^ ^software developmentk.6.3^ 39
0.014248 2 9

^software architecturesd.2.11^ ^software
developmentk.6.3^ 28 0.012760 2 8

^design tools and techniquesd.2.2^ ^software
developmentk.6.3^ 42 0.006638 2 8

^requirements/specificationsd.2.1^ ^software
developmentk.6.3^ 25 0.005319 2 15

^programming teamsd.2.9^ ^software developmentk.6.3^
7 0.004253 2 9

^generald.2.0^ ^software developmentk.6.3^ 28
0.004100 2 9

^metricsd.2.8^ ^software developmentk.6.3^ 15
0.003805 2 9

^javad.3.2^ ^programming environmentsd.2.6^ 11
0.003265 2 4
```

^javad.3.2^ ^software/program verificationd.2.4^ 10
0.002698 2 15

^object-oriented programmingd.1.5^
^requirements/specificationsd.2.1^ 14 0.002232 2 15

^javad.3.2^ ^user interfacesd.2.2^ 8 0.002069 2 4

^c++d.3.2^ ^design tools and techniquesd.2.2^ 12
0.001858 2 8

^object-oriented programmingd.1.5^ ^software
architecturesd.2.11^ 9 0.001764 2 8

^programming environmentsd.2.6^ ^software
developmentk.6.3^ 12 0.001720 2 4

0.039034 0.066729 0.000524 2855 331

3 15 23

^petri netsd.2.2^ 151 4 9 1 3

^applicationsi.6.3^ 140 2 8 1 3

^computer-aided engineeringj.6^ 85 2 6 1 3

^engineeringj.2^ 71 2 5 1 3

^manufacturingj.1^ 42 1 4 1 3

^generalg.2.0^ 94 2 3 1 3

^design tools and techniquesd.2.2^ 692 8 2 2 8

^algorithm design and analysisg.4^ 55 4 2 2 15

^generald.2.0^ 498 6 1 2 9

^stochastic processesg.3^ 19 1 1 1 3

^model validation and analysisi.6.4^ 49 1 1 1 3

^simulation output analysisi.6.6^ 28 1 1 1 3

^parallelism and concurrencyf.1.2^ 102 4 1 2 4

^performance of systemsc.4^ 109 2 1 2 14

^requirements/specificationsd.2.1^ 306 4 1 2 15

^computer-aided engineeringj.6^ ^manufacturingj.1^ 24 0.161345 1 0

^petri netsd.2.2^ ^stochastic processesg.3^ 13 0.058906 1 0

^applicationsi.6.3^ ^petri netsd.2.2^ 27 0.034484 1 0

^computer-aided engineeringj.6^ ^engineeringj.2^ 13 0.028003 1 0

^engineeringj.2^ ^manufacturingj.1^ 8 0.021462 1 0

^applicationsi.6.3^ ^model validation and analysisi.6.4^ 12 0.020991 1 0

^applicationsi.6.3^ ^manufacturingj.1^ 11 0.020578 1 0

^manufacturingj.1^ ^petri netsd.2.2^ 11 0.019079 1 0

^applicationsi.6.3^ ^simulation output analysisi.6.6^ 8 0.016327 1 0

^computer-aided engineeringj.6^ ^petri netsd.2.2^ 12 0.011219 1 0

^applicationsi.6.3^ ^computer-aided engineeringj.6^ 11 0.010168 1 0

^generalg.2.0^ ^petri netsd.2.2^ 12 0.010145 1 0

^applicationsi.6.3^ ^engineeringj.2^ 9 0.008149 3 0

^applicationsi.6.3^ ^generalg.2.0^ 8 0.004863 3 0

^engineeringj.2^ ^petri netsd.2.2^ 7 0.004570 3 0

^algorithm design and analysisg.4^ ^generalg.2.0^ 7 0.009478 2 15

^parallelism and concurrencyf.1.2^ ^petri netsd.2.2^ 12 0.009349 2 4

^performance of systemsc.4^ ^petri netsd.2.2^ 12 0.008749 2 14

^algorithm design and analysisg.4^ ^petri netsd.2.2^ 8 0.007706 2 15

^design tools and techniquesd.2.2^ ^engineeringj.2^ 12 0.002931 2 8

^applicationsi.6.3^ ^requirements/specificationsd.2.1^ 11 0.002824 2 15

^computer-aided engineeringj.6^ ^design tools and techniquesd.2.2^ 10 0.001700 2 8

^computer-aided engineeringj.6^ ^generald.2.0^ 7 0.001158 2 9

0.028686 0.043895 0.000334 1900 139

4 18 24

^programming environmentsd.2.6^ 218 3 6 1 4

^user/machine systemsh.1.2^ 125 2 6 1 4

^user interfacesd.2.2^ 182 2 5 1 4

^parallelism and concurrencyf.1.2^ 102 4 5 1 4

^documentationd.2.7^ 121 1 4 1 4

^user interfacesh.5.2^ 137 1 3 1 4

^visual programmingd.1.7^ 51 1 2 1 4

^software developmentk.6.3^ 384 6 2 2 2

^training, help, and documentationh.5.2^ 32 1 2 1 4

^concurrent programmingd.1.3^ 131 3 2 2 8

^language classificationsd.3.2^ 118 2 2 2 8

^javad.3.2^ 170 3 2 2 2

^design tools and techniquesd.2.2^ 692 8 2 2 8

^electronic publishingi.7.4^ 10 1 1 1 4

^generald.2.0^ 498 6 1 2 9

^human factorsh.1.2^ 37 1 1 1 4

^algorithm design and analysisg.4^ 55 4 1 2 15

^petri netsd.2.2^ 151 4 1 2 3

^documentationd.2.7^ ^training, help, and documentationh.5.2^ 22 0.125000 1 0

^documentationd.2.7^ ^electronic publishingi.7.4^ 9 0.066942 1 0

^user interfacesd.2.2^ ^user interfacesh.5.2^ 31
0.038542 1 0

^documentationd.2.7^ ^user/machine systemsh.1.2^ 24
0.038083 1 0

^training, help, and documentationh.5.2^
^user/machine systemsh.1.2^ 11 0.030250 1 0

^human factorsh.1.2^ ^user interfacesd.2.2^ 10
0.014850 1 0

^user interfacesd.2.2^ ^visual programmingd.1.7^ 10
0.010774 1 0

^user interfacesd.2.2^ ^user/machine systemsh.1.2^
15 0.009890 1 0

^parallelism and concurrencyf.1.2^ ^programming
environmentsd.2.6^ 12 0.006476 1 0

^user interfacesh.5.2^ ^user/machine systemsh.1.2^ 9
0.004730 1 0

^programming environmentsd.2.6^ ^visual
programmingd.1.7^ 7 0.004407 1 0

^concurrent programmingd.1.3^ ^parallelism and
concurrencyf.1.2^ 18 0.024248 2 8

^algorithm design and analysisg.4^ ^parallelism and
concurrencyf.1.2^ 8 0.011408 2 15

^parallelism and concurrencyf.1.2^ ^petri netsd.2.2^
12 0.009349 2 3

^language classificationsd.3.2^ ^parallelism and
concurrencyf.1.2^ 7 0.004071 2 8

^javad.3.2^ ^programming environmentsd.2.6^ 11
0.003265 2 2

^language classificationsd.3.2^ ^programming environmentsd.2.6^ 9 0.003149 2 8

^concurrent programmingd.1.3^ ^programming environmentsd.2.6^ 9 0.002836 2 8

^generald.2.0^ ^user/machine systemsh.1.2^ 13 0.002715 2 9

^javad.3.2^ ^user interfacesd.2.2^ 8 0.002069 2 2

^design tools and techniquesd.2.2^ ^user interfacesh.5.2^ 14 0.002067 2 8

^programming environmentsd.2.6^ ^software developmentk.6.3^ 12 0.001720 2 2

^design tools and techniquesd.2.2^ ^user/machine systemsh.1.2^ 12 0.001665 2 8

^documentationd.2.7^ ^software developmentk.6.3^ 8 0.001377 2 2

0.031813 0.069940 0.000874 2390 220

5 2 1

^web-based servicesh.3.5^ 67 1 1 1 5

^web-based interactionh.5.3^ 52 1 1 1 5

^web-based interactionh.5.3^ ^web-based servicesh.3.5^ 19 0.103617 1 0

0.103617 0.000000 0.000000 71 19

6 2 1

^distributed databasesh.2.4^ 17 1 1 1 6

^distributed systemsc.2.4^ 47 1 1 1 6

^distributed databasesh.2.4^ ^distributed systemsc.2.4^ 7 0.061327 1 0

0.061327 0.000000 0.000000 56 7

7 2 1

^parallel programmingd.1.3^ 39 1 1 1 7

^performance measuresd.2.8^ 47 1 1 1 7

^parallel programmingd.1.3^ ^performance measuresd.2.8^ 10 0.054555 1 0

0.054555 0.000000 0.000000 76 10

8 20 22

^design tools and techniquesd.2.2^ 692 8 13 1 8

^concurrent programmingd.1.3^ 131 3 5 1 8

^language classificationsd.3.2^ 118 2 3 1 8

^software architecturesd.2.11^ 160 3 3 1 8

^software developmentk.6.3^ 384 6 2 2 2

^programming environmentsd.2.6^ 218 3 2 2 4

^interoperabilityd.2.12^ 63 1 2 1 8

^parallelism and concurrencyf.1.2^ 102 4 2 2 4

^managementd.2.9^ 278 4 1 2 9

^requirements/specificationsd.2.1^ 306 4 1 2 15

^testing and debuggingd.2.5^ 271 3 1 2 15

^distribution, maintenance, and enhancementd.2.7^ 82 3 1 2 15

^parallel architecturesc.1.4^ 21 1 1 1 8

^processorsd.3.4^ 50 1 1 1 8

^communications managementd.4.4^ 13 1 1 1 8

^process managementd.4.1^ 24 1 1 1 8

^physical sciences and engineeringj.2^ 21 1 1 1 8

^software/program verificationd.2.4^ 218 3 1 2 15

^engineeringj.2^ 71 2 1 2 3

^language constructs and featuresd.3.3^ 81 2 1 2 12

^interoperabilityd.2.12^ ^parallel architecturesc.1.4^ 8 0.048375 1 0

^design tools and techniquesd.2.2^ ^processorsd.3.4^ 18 0.009364 1 0

^communications managementd.4.4^ ^design tools and techniquesd.2.2^ 9 0.009004 1 0

^design tools and techniquesd.2.2^ ^language classificationsd.3.2^ 25 0.007654 1 0

^design tools and techniquesd.2.2^ ^process managementd.4.1^ 11 0.007286 1 0

^design tools and techniquesd.2.2^ ^physical sciences and engineeringj.2^ 10 0.006881 1 0

^interoperabilityd.2.12^ ^software architecturesd.2.11^ 7 0.004861 1 0

^concurrent programmingd.1.3^ ^design tools and techniquesd.2.2^ 20 0.004412 1 0

^design tools and techniquesd.2.2^ ^software architecturesd.2.11^ 22 0.004371 1 0

^concurrent programmingd.1.3^ ^parallelism and concurrencyf.1.2^ 18 0.024248 2 4

^software architecturesd.2.11^ ^software developmentk.6.3^ 28 0.012760 2 2

^design tools and techniquesd.2.2^ ^software developmentk.6.3^ 42 0.006638 2 2

^language classificationsd.3.2^ ^parallelism and concurrencyf.1.2^ 7 0.004071 2 4

^concurrent programmingd.1.3^ ^software/program verificationd.2.4^ 10 0.003502 2 15

^language classificationsd.3.2^ ^programming environmentsd.2.6^ 9 0.003149 2 4

^design tools and techniquesd.2.2^ ^engineeringj.2^ 12 0.002931 2 3

^concurrent programmingd.1.3^ ^programming environmentsd.2.6^ 9 0.002836 2 4

^design tools and techniquesd.2.2^ ^language constructs and featuresd.3.3^ 12 0.002569 2 12

^design tools and techniquesd.2.2^ ^managementd.2.9^ 22 0.002516 2 9

^design tools and techniquesd.2.2^
^requirements/specificationsd.2.1^ 22 0.002286 2 15

^concurrent programmingd.1.3^ ^testing and
debuggingd.2.5^ 9 0.002282 2 15

^design tools and techniquesd.2.2^ ^distribution,
maintenance, and enhancementd.2.7^ 11 0.002132 2 15

0.011357 0.071920 0.000878 2301 277

9 19 24

^generald.2.0^ 498 6 10 1 9

^managementd.2.9^ 278 4 7 1 9

^software managementk.6.3^ 162 2 4 1 9

^software developmentk.6.3^ 384 6 4 2 2

^metricsd.2.8^ 154 2 3 1 9

^programming teamsd.2.9^ 30 2 2 1 9

^project and people managementk.6.1^ 35 1 2 1 9

^computer science educationk.3.2^ 48 1 2 1 9

^curriculumk.3.2^ 77 1 2 1 9

^design tools and techniquesd.2.2^ 692 8 2 2 8

^object-oriented programmingd.1.5^ 287 3 2 2 2

^distribution, maintenance, and enhancementd.2.7^ 82
3 1 2 15

^user/machine systemsh.1.2^ 125 2 1 2 4

^testing and debuggingd.2.5^ 271 3 1 2 15

^generalc.2.0^ 41 2 1 2 14

^algorithm design and analysisg.4^ 55 4 1 2 15

^computer-aided engineeringj.6^ 85 2 1 2 3

^generalf.3.0^ 15 1 1 1 9

^generalk.5.0^ 11 1 1 1 9

^programming teamsd.2.9^ ^project and people managementk.6.1^ 7 0.046667 1 0

^computer science educationk.3.2^ ^generald.2.0^ 33 0.045557 1 0

^computer science educationk.3.2^ ^curriculumk.3.2^ 12 0.038961 1 0

^generald.2.0^ ^software managementk.6.3^ 46 0.026228 1 0

^curriculumk.3.2^ ^generald.2.0^ 30 0.023471 1 0

^managementd.2.9^ ^software managementk.6.3^ 29 0.018674 1 0

^generald.2.0^ ^generalk.5.0^ 7 0.008945 1 0

^generald.2.0^ ^generalf.3.0^ 8 0.008568 1 0

^managementd.2.9^ ^metricsd.2.8^ 19 0.008432 1 0

^managementd.2.9^ ^project and people managementk.6.1^ 8 0.006578 1 0

^managementd.2.9^ ^software developmentk.6.3^ 39 0.014248 2 2

^programming teamsd.2.9^ ^software developmentk.6.3^ 7 0.004253 2 2

^generald.2.0^ ^software developmentk.6.3^ 28 0.004100 2 2

^metricsd.2.8^ ^software developmentk.6.3^ 15 0.003805 2 2

^distribution, maintenance, and enhancementd.2.7^ ^managementd.2.9^ 9 0.003553 2 15

^generald.2.0^ ^user/machine systemsh.1.2^ 13 0.002715 2 4

^design tools and techniquesd.2.2^ ^managementd.2.9^ 22 0.002516 2 8

^generalc.2.0^ ^generald.2.0^ 7 0.002400 2 14

^algorithm design and analysisg.4^ ^generald.2.0^ 7 0.001789 2 15

^managementd.2.9^ ^object-oriented programmingd.1.5^ 11 0.001517 2 2

^design tools and techniquesd.2.2^ ^software managementk.6.3^ 13 0.001508 2 8

^software managementk.6.3^ ^testing and debuggingd.2.5^ 8 0.001458 2 15

^metricsd.2.8^ ^object-oriented programmingd.1.5^ 8 0.001448 2 2

^computer-aided engineeringj.6^ ^generald.2.0^ 7 0.001158 2 3

0.023208 0.046466 0.000298 2383 313

```
10 3 2

^compilersd.3.4^ 70 1 2 1 10

^optimizationd.3.4^ 31 1 1 1 10

^design tools and techniquesd.2.2^ 692 8 1 2 8

^compilersd.3.4^ ^optimizationd.3.4^ 10 0.046083 1 0

^compilersd.3.4^ ^design tools and techniquesd.2.2^
9 0.001672 2 8

0.046083 0.001672 0.000003 764 17



11 3 2

^software maintenancek.6.3^ 82 1 2 1 11

^restructuring, reverse engineering, and
reengineeringd.2.7^ 86 1 1 1 11

^software developmentk.6.3^ 384 6 1 2 2

^restructuring, reverse engineering, and
reengineeringd.2.7^ ^software maintenancek.6.3^ 16
0.036302 1 0

^software developmentk.6.3^ ^software
maintenancek.6.3^ 7 0.001556 2 2

0.036302 0.001556 0.000002 515 23



12 4 3

^language constructs and featuresd.3.3^ 81 2 3 1 12
```

^studies of program constructsf.3.3^ 28 1 1 1 12

^visual basicd.2.2^ 62 1 1 1 12

^design tools and techniquesd.2.2^ 692 8 1 2 8

^language constructs and featuresd.3.3^ ^studies of program constructsf.3.3^ 9 0.035714 1 0

^language constructs and featuresd.3.3^ ^visual basicd.2.2^ 9 0.016129 1 0

^design tools and techniquesd.2.2^ ^language constructs and featuresd.3.3^ 12 0.002569 2 8

0.025922 0.002569 0.000007 808 28

13 2 1

^real-time and embedded systemsc.3^ 50 1 1 1 13

^real-time systems and embedded systemsd.4.7^ 37 1 1 1 13

^real-time and embedded systemsc.3^ ^real-time systems and embedded systemsd.4.7^ 7 0.026486 1 0

0.026486 0.000000 0.000000 68 7

14 5 4

^performance of systemsc.4^ 109 2 3 1 14

^generalc.2.0^ 41 2 2 1 14

^network protocolsc.2.2^ 54 1 1 1 14

^petri netsd.2.2^ 151 4 1 2 3

^generald.2.0^ 498 6 1 2 9

^network protocolsc.2.2^ ^performance of systemsc.4^ 10 0.016989 1 0

^generalc.2.0^ ^performance of systemsc.4^ 7 0.010964 1 0

^performance of systemsc.4^ ^petri netsd.2.2^ 12 0.008749 2 3

^generalc.2.0^ ^generald.2.0^ 7 0.002400 2 9

0.013977 0.011149 0.000082 784 31

15 20 24

^requirements/specificationsd.2.1^ 306 4 9 1 15

^testing and debuggingd.2.5^ 271 3 7 1 15

^algorithm design and analysisg.4^ 55 4 5 1 15

^software/program verificationd.2.4^ 218 3 5 1 15

^distribution, maintenance, and enhancementd.2.7^ 82 3 3 1 15

^software developmentk.6.3^ 384 6 2 2 2

^specifying and verifying and reasoning about programsf.3.1^ 45 1 2 1 15

^concurrent programmingd.1.3^ 131 3 2 2 8

^design tools and techniquesd.2.2^ 692 8 2 2 8

^petri netsd.2.2^ 151 4 1 2 3

^reliabilityd.2.4^ 44 1 1 1 15

^managementd.2.9^ 278 4 1 2 9

^software architecturesd.2.11^ 160 3 1 2 8

^applicationsi.6.3^ 140 2 1 2 3

^javad.3.2^ 170 3 1 2 2

^software managementk.6.3^ 162 2 1 2 9

^object-oriented programmingd.1.5^ 287 3 1 2 2

^generald.2.0^ 498 6 1 2 9

^parallelism and concurrencyf.1.2^ 102 4 1 2 4

^generalg.2.0^ 94 2 1 2 3

^software/program verificationd.2.4^ ^testing and debuggingd.2.5^ 25 0.010579 1 0

^software/program verificationd.2.4^ ^specifying and verifying and reasoning about programsf.3.1^ 10 0.010194 1 0

^requirements/specificationsd.2.1^ ^specifying and verifying and reasoning about programsf.3.1^ 8 0.004648 1 0

^reliabilityd.2.4^ ^testing and debuggingd.2.5^ 7 0.004109 1 0

^algorithm design and analysisg.4^ ^requirements/specificationsd.2.1^ 8 0.003803 1 0

^distribution, maintenance, and enhancementd.2.7^ ^testing and debuggingd.2.5^ 8 0.002880 1 0

^requirements/specificationsd.2.1^ ^software/program verificationd.2.4^ 13 0.002533 1 0

^requirements/specificationsd.2.1^ ^testing and debuggingd.2.5^ 9 0.000977 1 0

^algorithm design and analysisg.4^ ^parallelism and concurrencyf.1.2^ 8 0.011408 2 4

^algorithm design and analysisg.4^ ^generalg.2.0^ 7 0.009478 2 3

^algorithm design and analysisg.4^ ^petri netsd.2.2^ 8 0.007706 2 3

^requirements/specificationsd.2.1^ ^software developmentk.6.3^ 25 0.005319 2 2

^distribution, maintenance, and enhancementd.2.7^ ^managementd.2.9^ 9 0.003553 2 9

^concurrent programmingd.1.3^ ^software/program verificationd.2.4^ 10 0.003502 2 8

^applicationsi.6.3^ ^requirements/specificationsd.2.1^ 11 0.002824 2 3

^javad.3.2^ ^software/program verificationd.2.4^ 10 0.002698 2 2

^design tools and techniquesd.2.2^ ^requirements/specificationsd.2.1^ 22 0.002286 2 8

^concurrent programmingd.1.3^ ^testing and debuggingd.2.5^ 9 0.002282 2 8

^object-oriented programmingd.1.5^ ^requirements/specificationsd.2.1^ 14 0.002232 2 2

^design tools and techniquesd.2.2^ ^distribution, maintenance, and enhancementd.2.7^ 11 0.002132 2 8

^algorithm design and analysisg.4^ ^generald.2.0^ 7
0.001789 2 9

^requirements/specificationsd.2.1^ ^software
architecturesd.2.11^ 9 0.001654 2 8

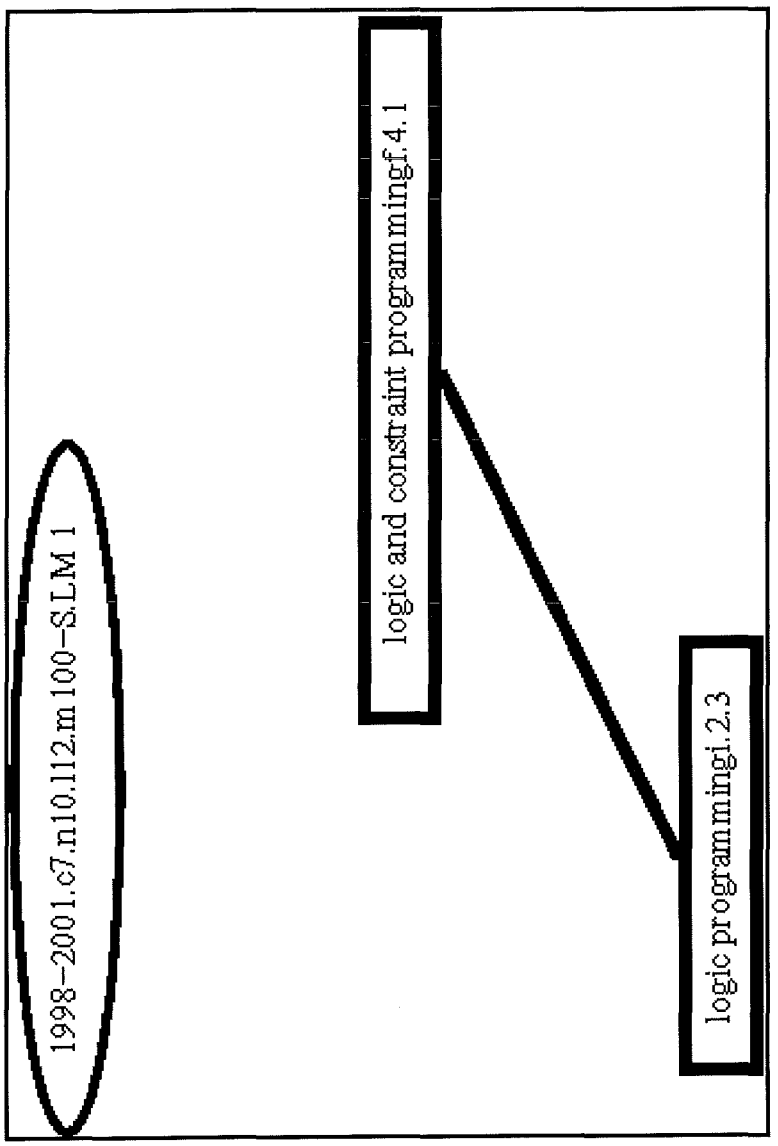
^software managementk.6.3^ ^testing and
debuggingd.2.5^ 8 0.001458 2 9

^software developmentk.6.3^ ^testing and
debuggingd.2.5^ 11 0.001163 2 2

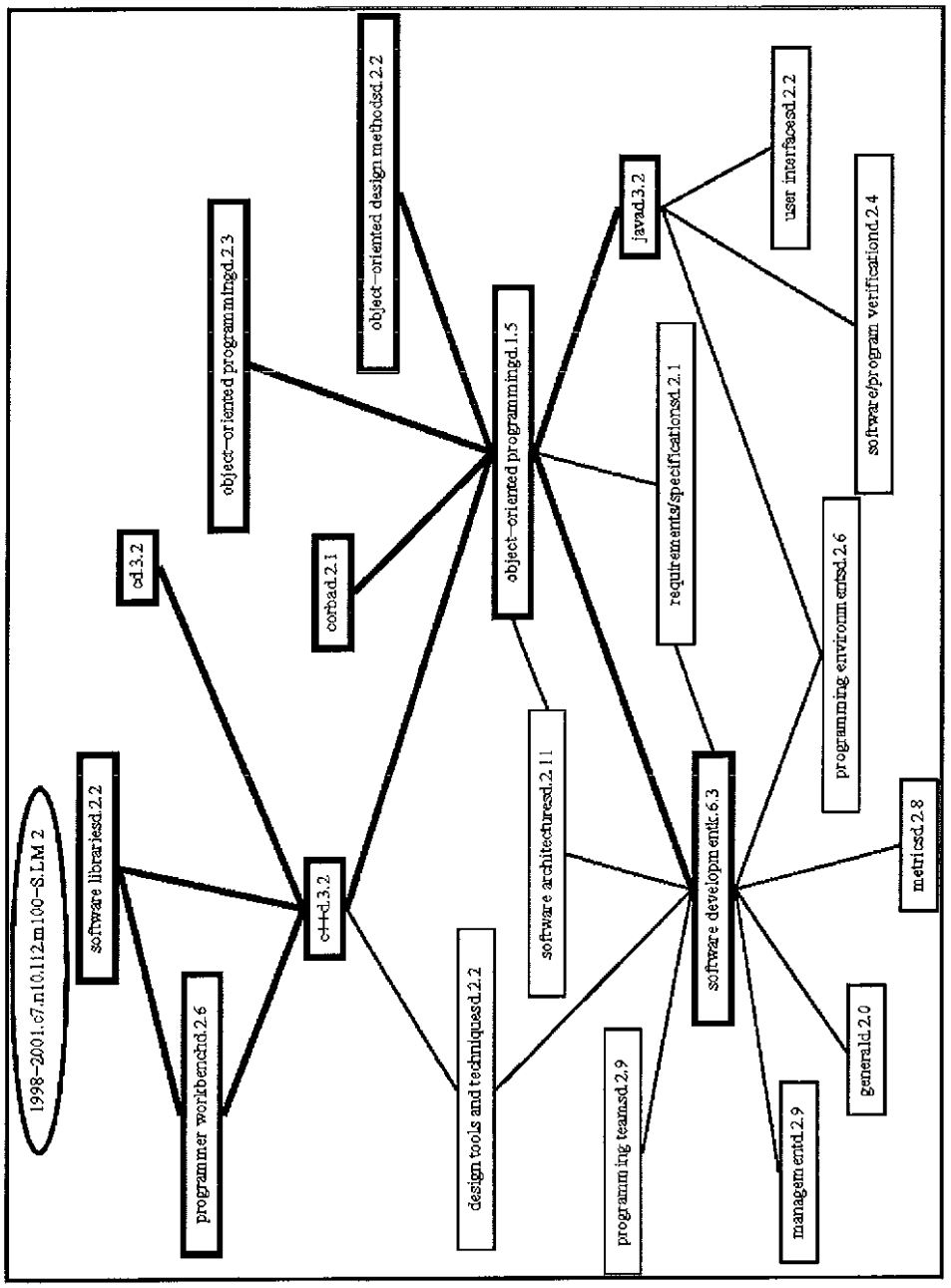
0.004965 0.061484 0.000377 2873 211

APPENDIX H

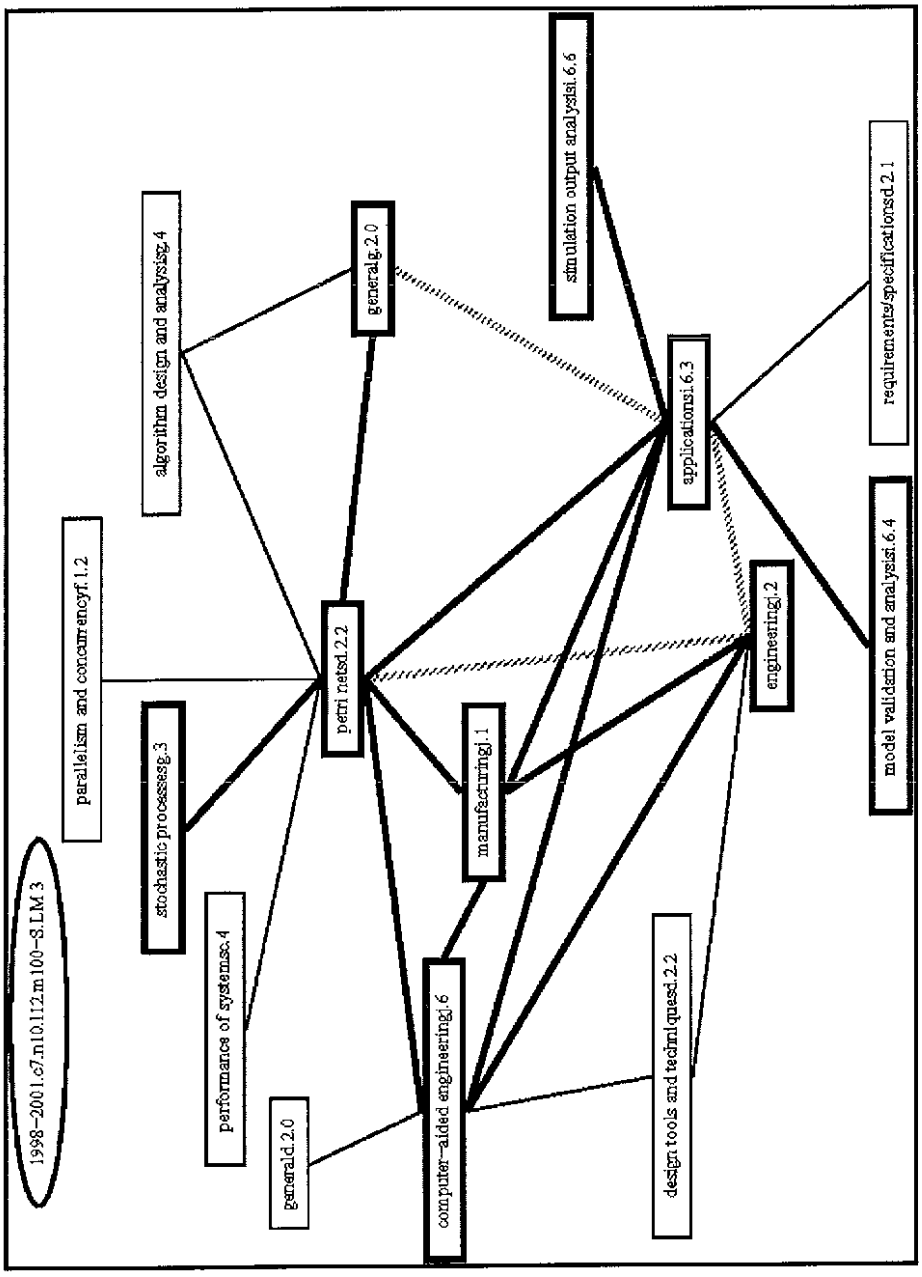
Keyword Maps



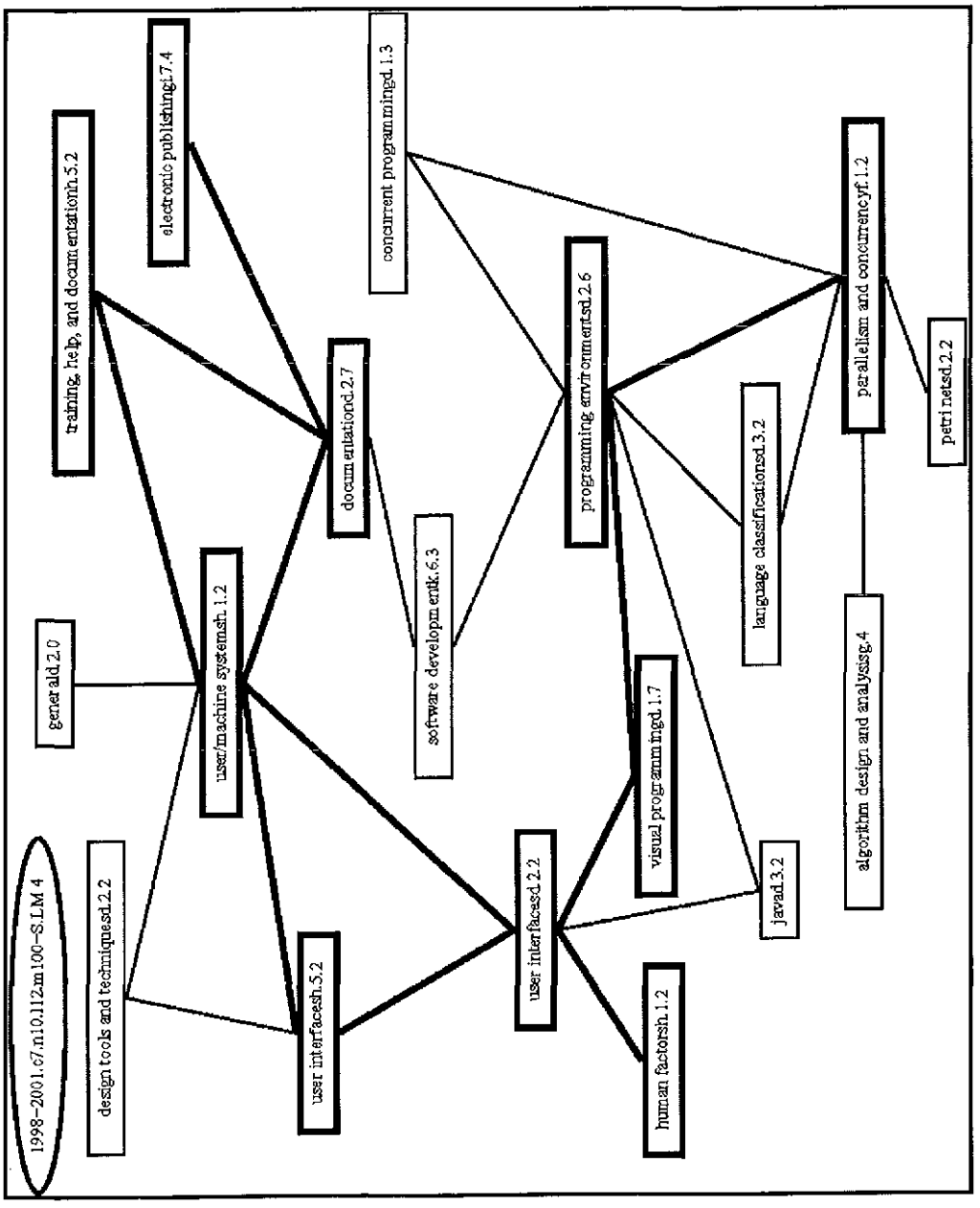
Map-1: Logic and constraint programming



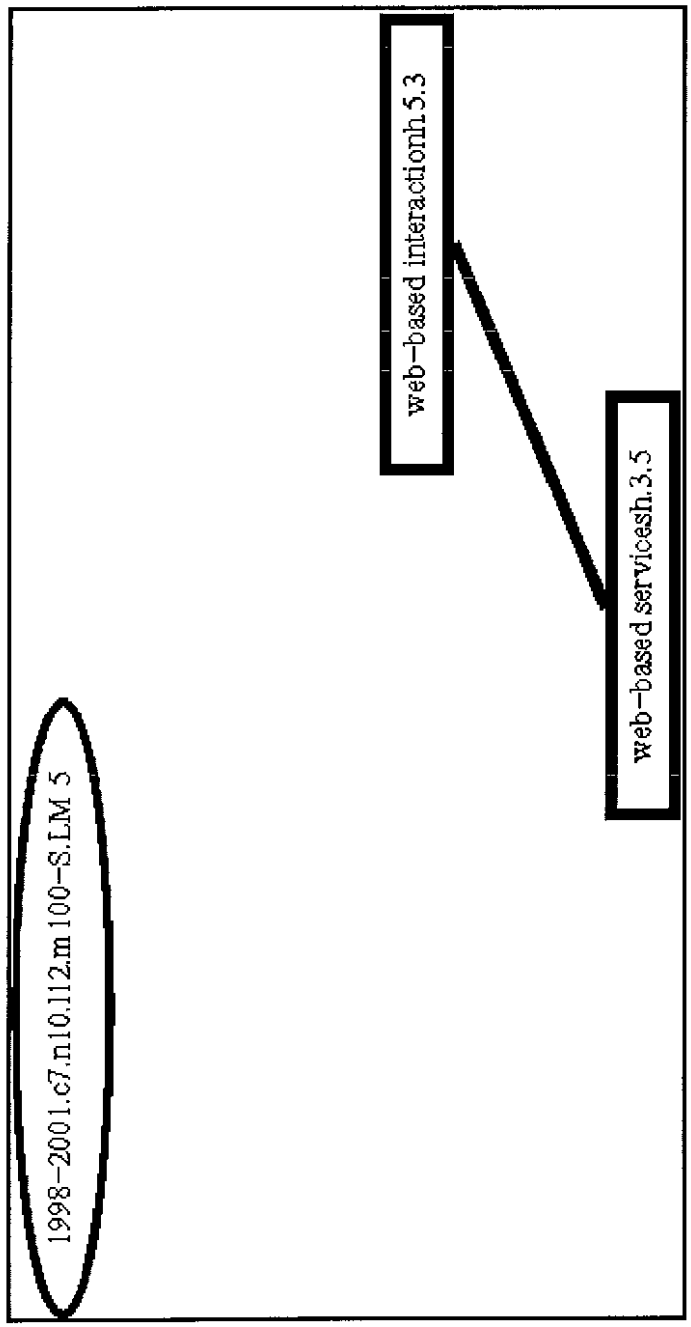
Map-2: Software development / object-oriented programming



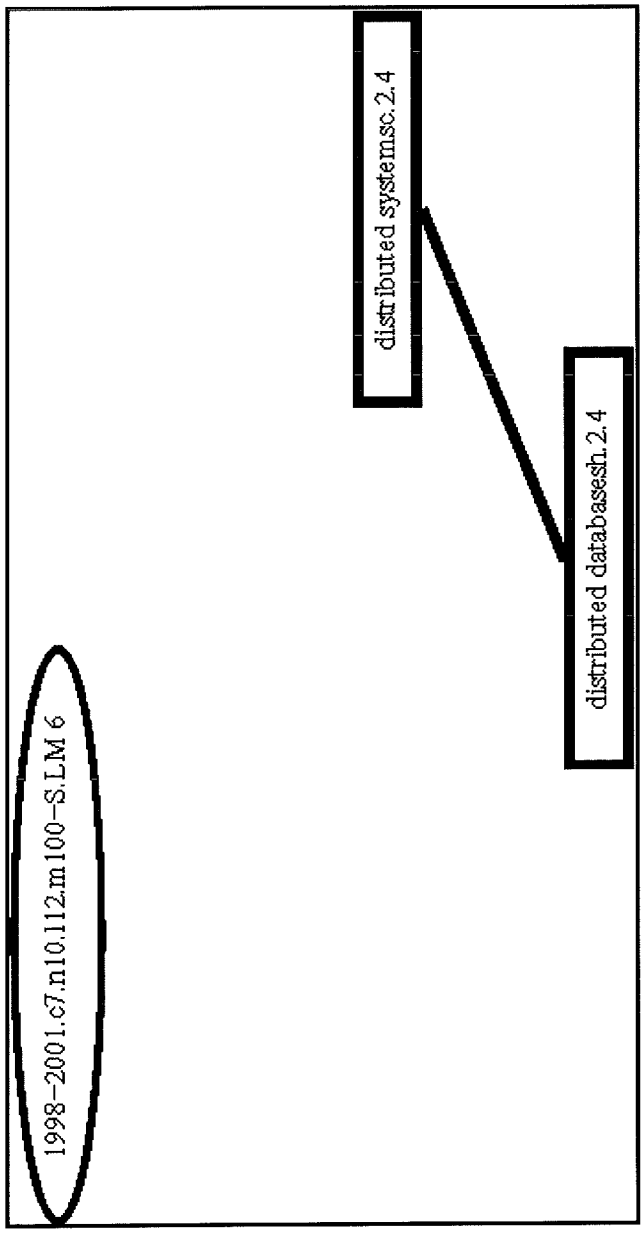
Map-3: Applications / Petri nets / computer-aided engineering



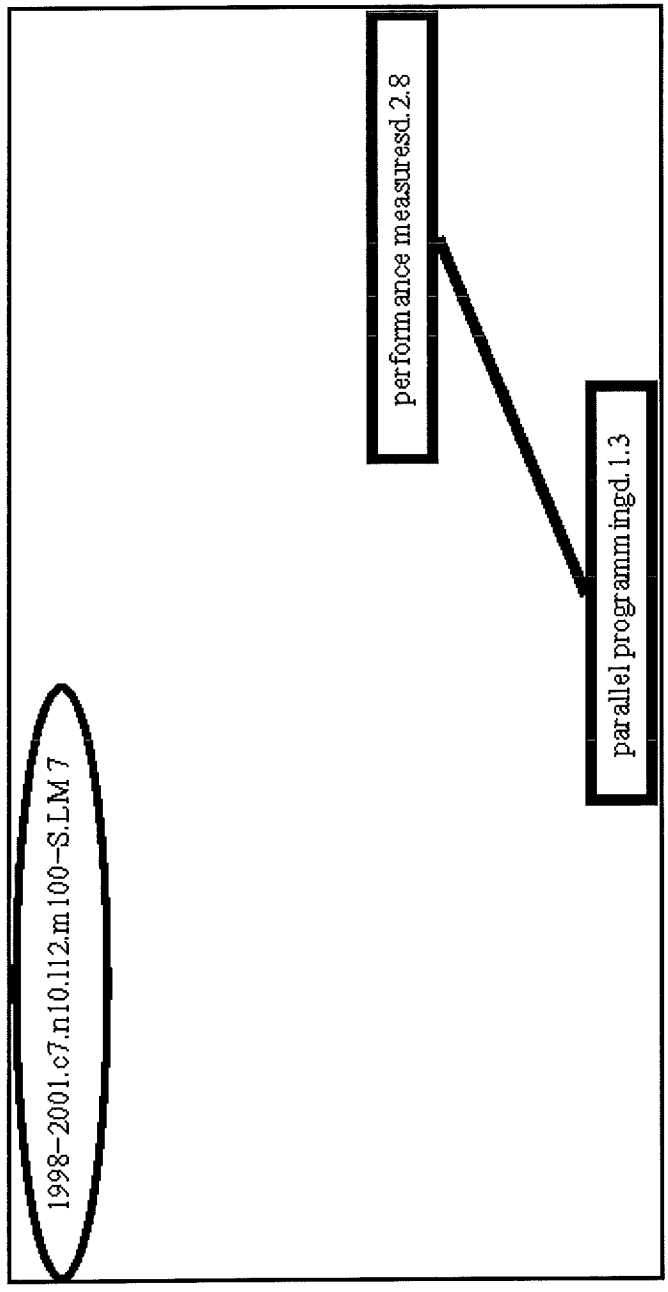
Map-4: User interfaces / documentation



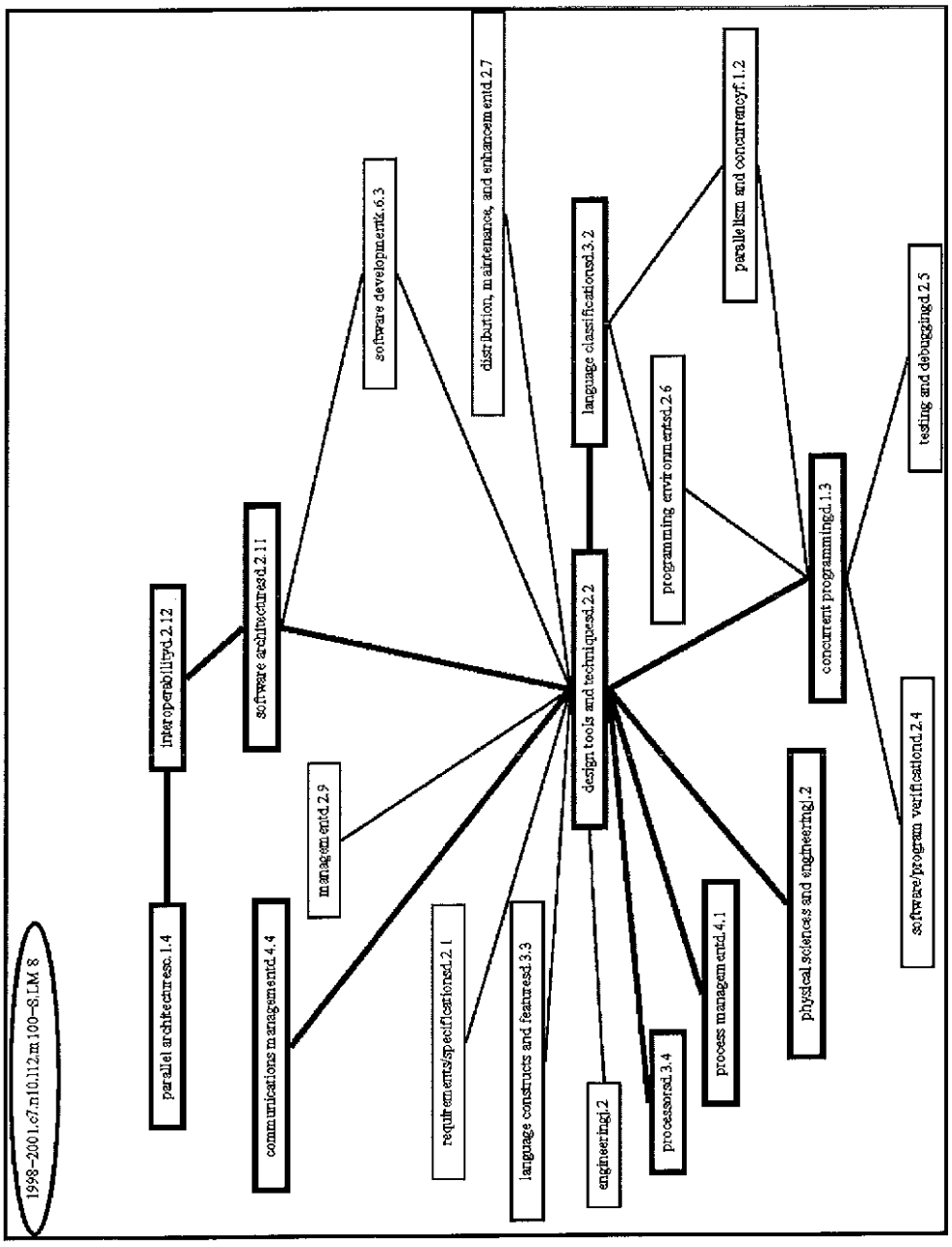
Map-5: Web-based services



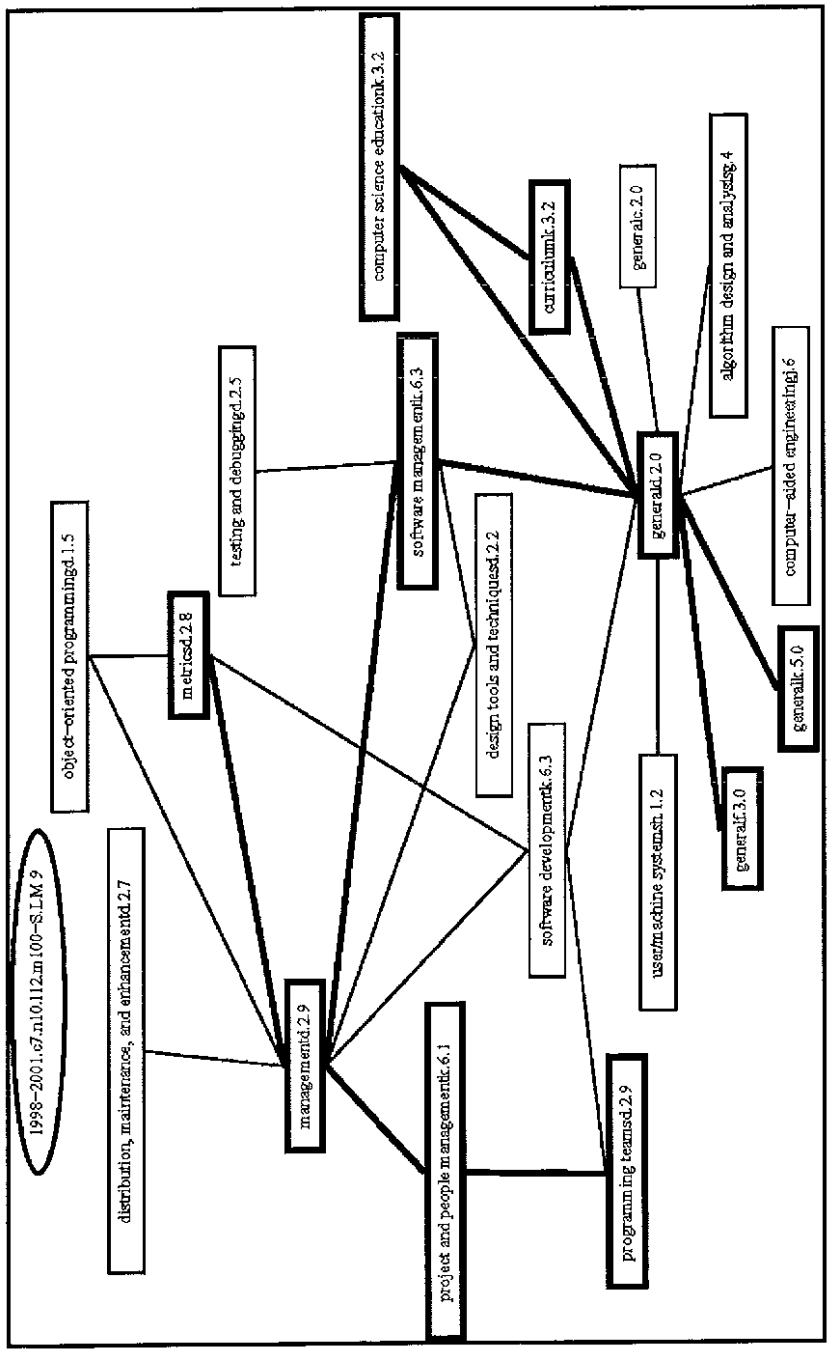
Map-6: Distributed systems



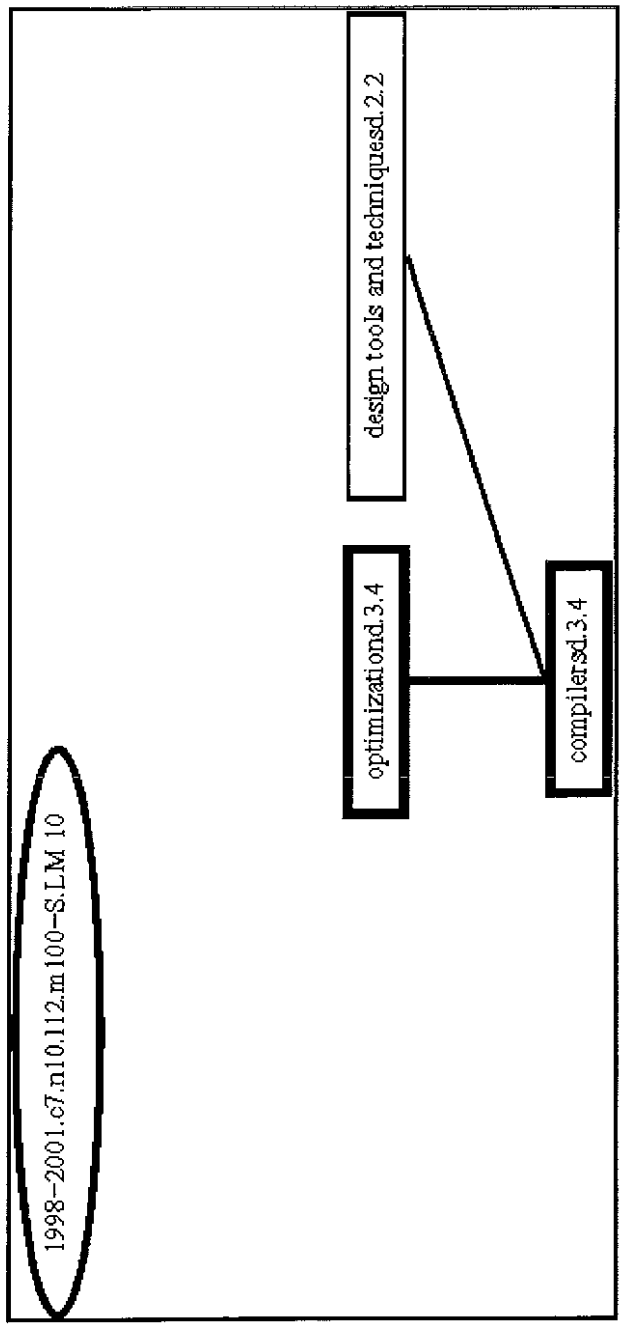
Map-7: Performance measures / parallel programming



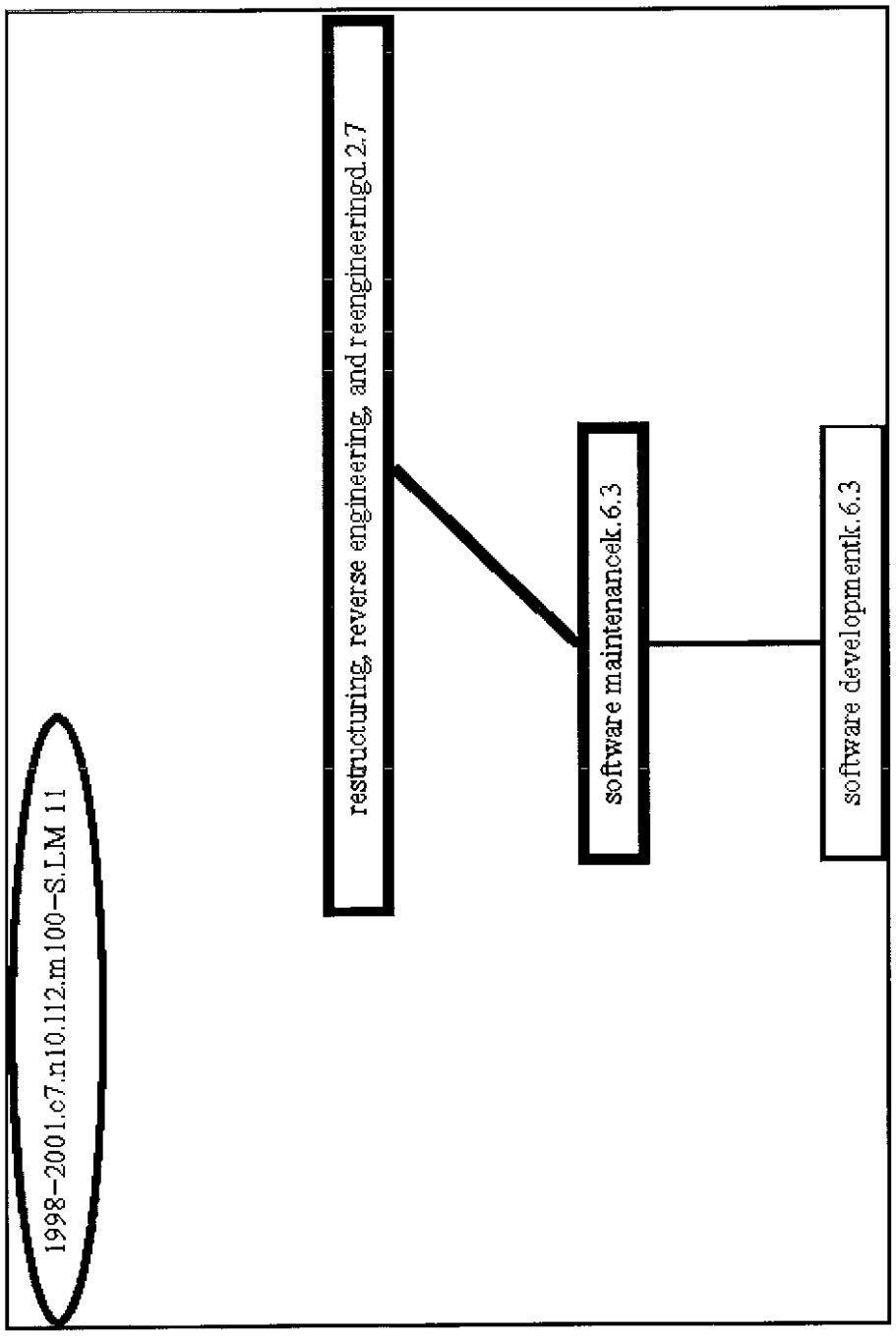
Map-8: Design tools and techniques



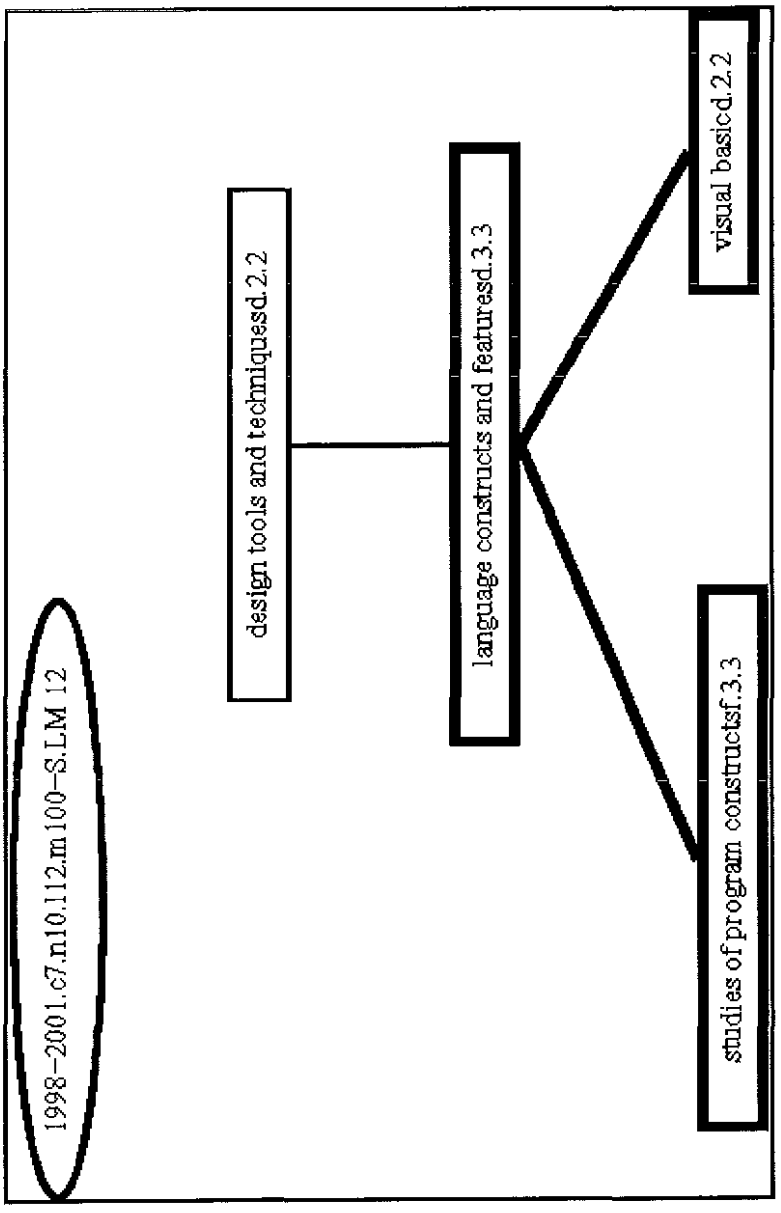
Map-9: Management / metrics



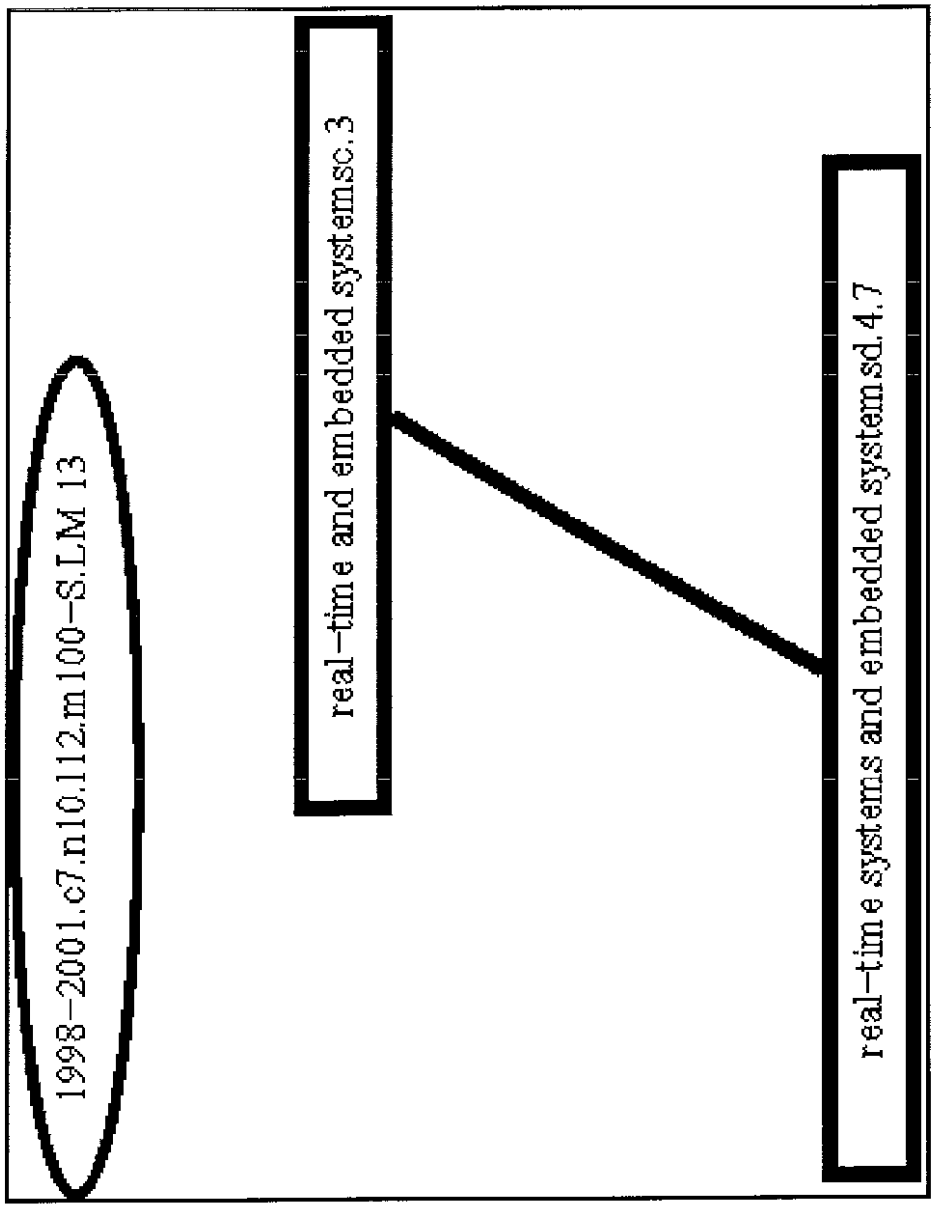
Map-10: Compilers / optimization



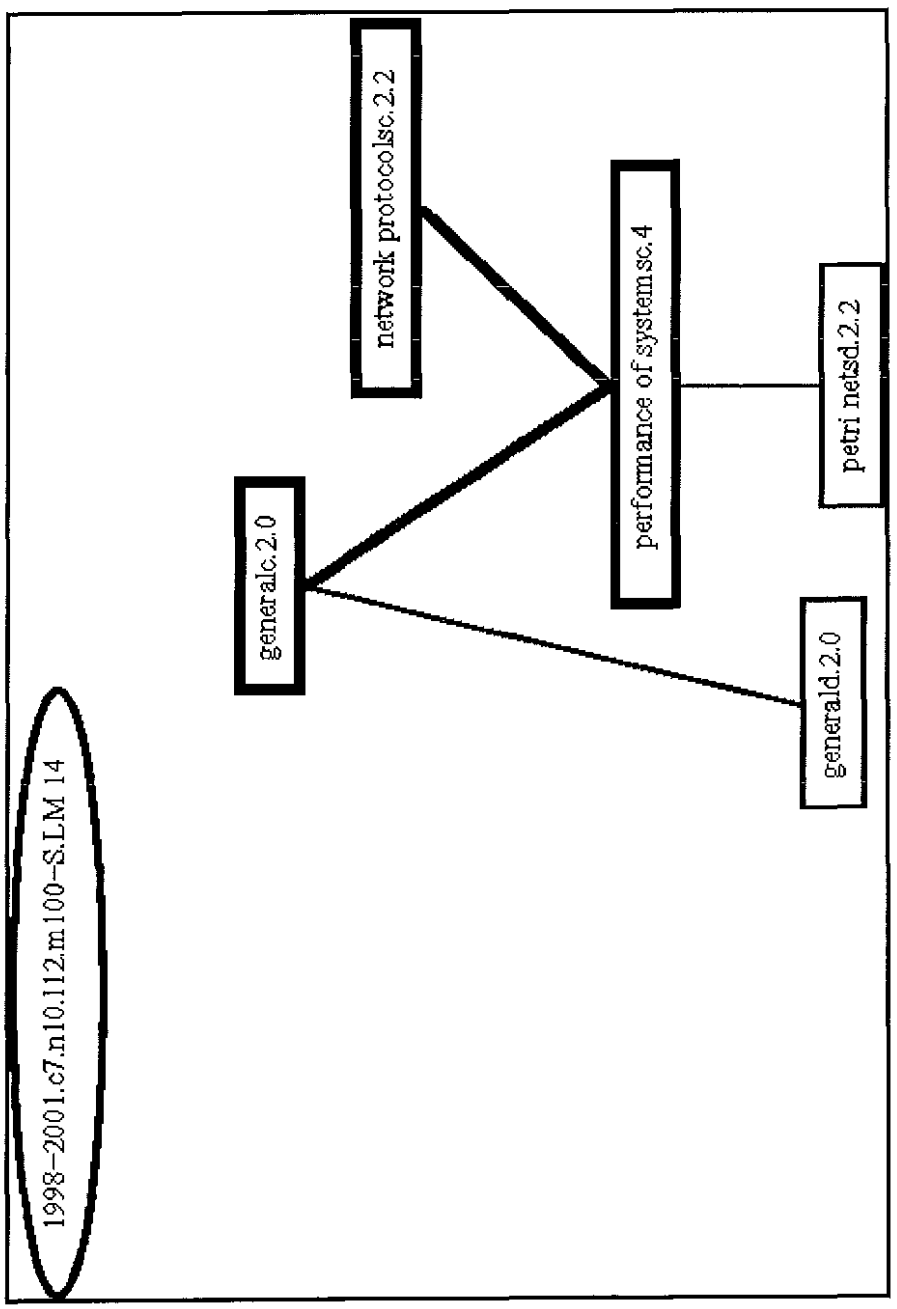
Map-11: Software maintenance



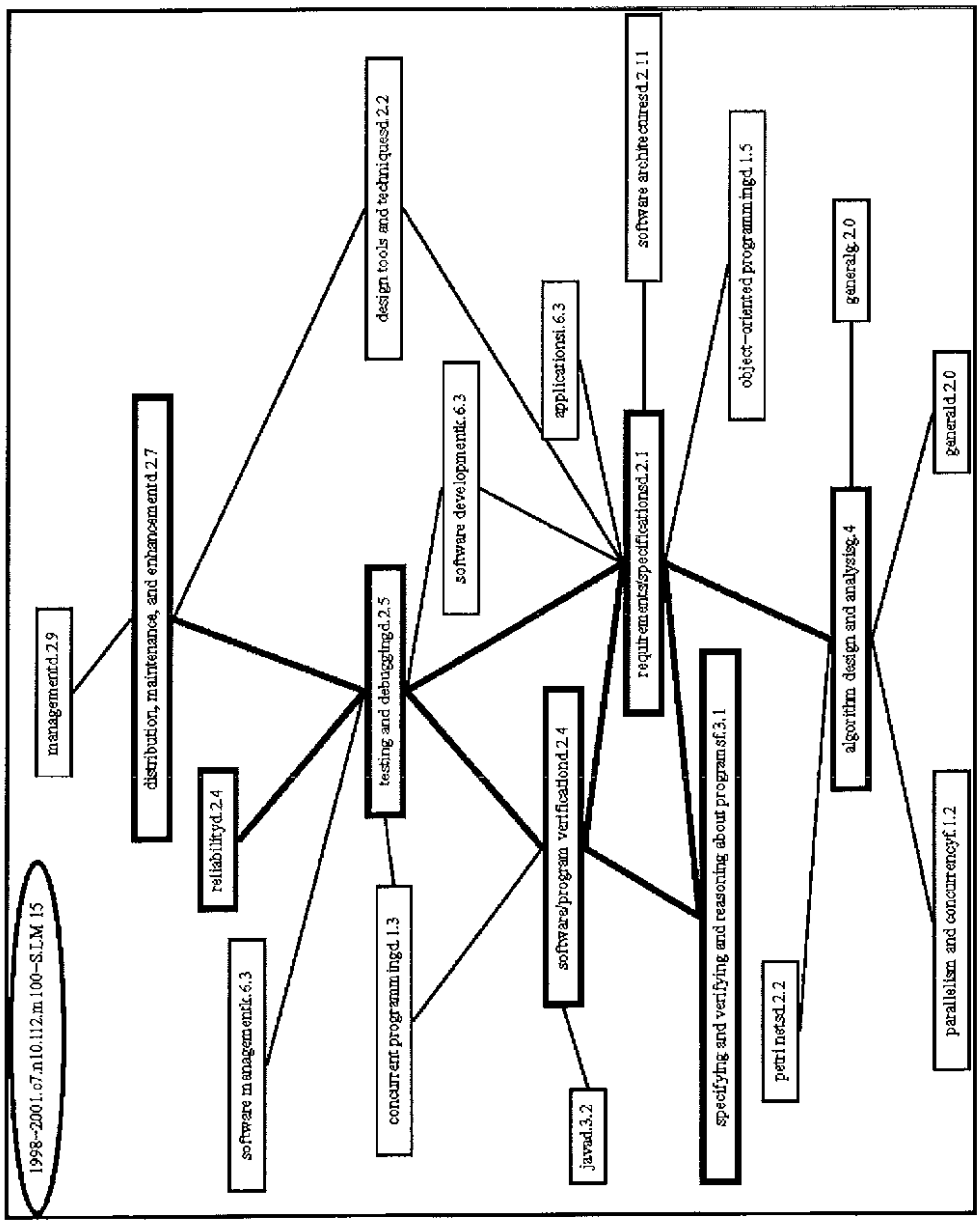
Map-12: Language constructs and features



Map-13: Real-time and embedded systems



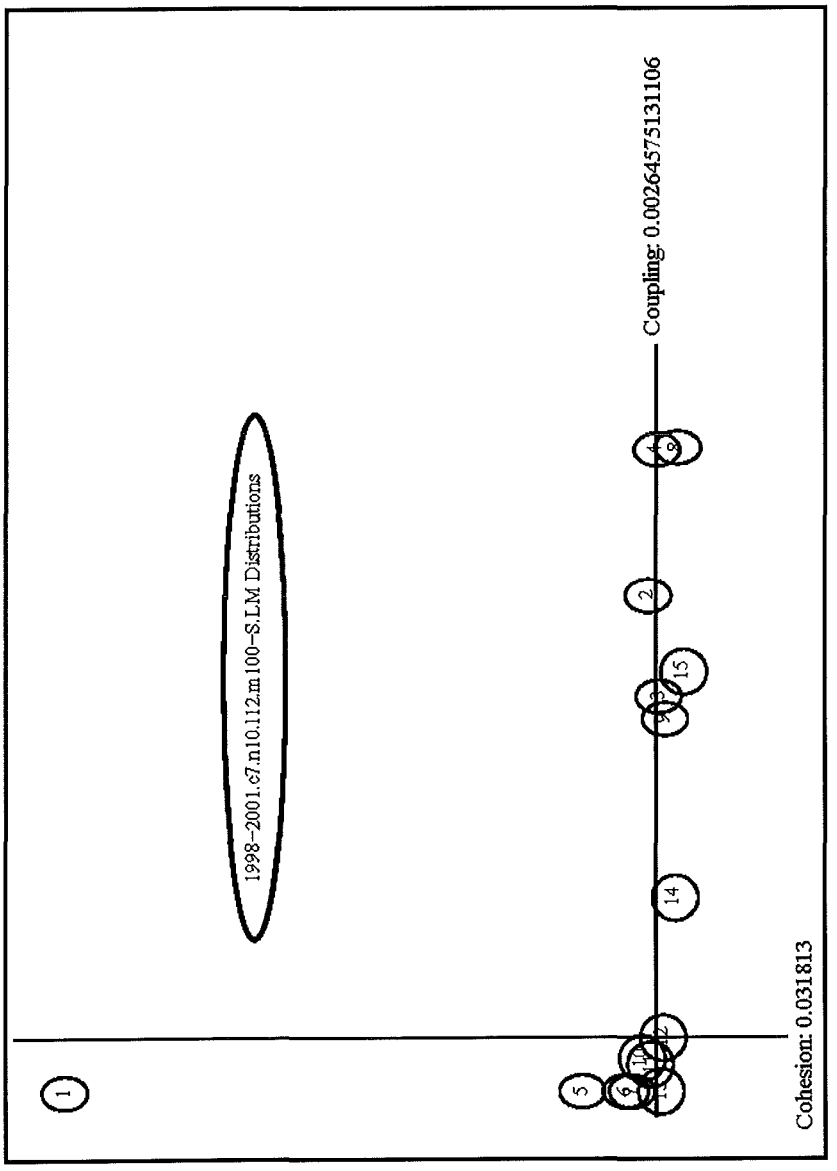
Map-14: Performance of systems / network protocols



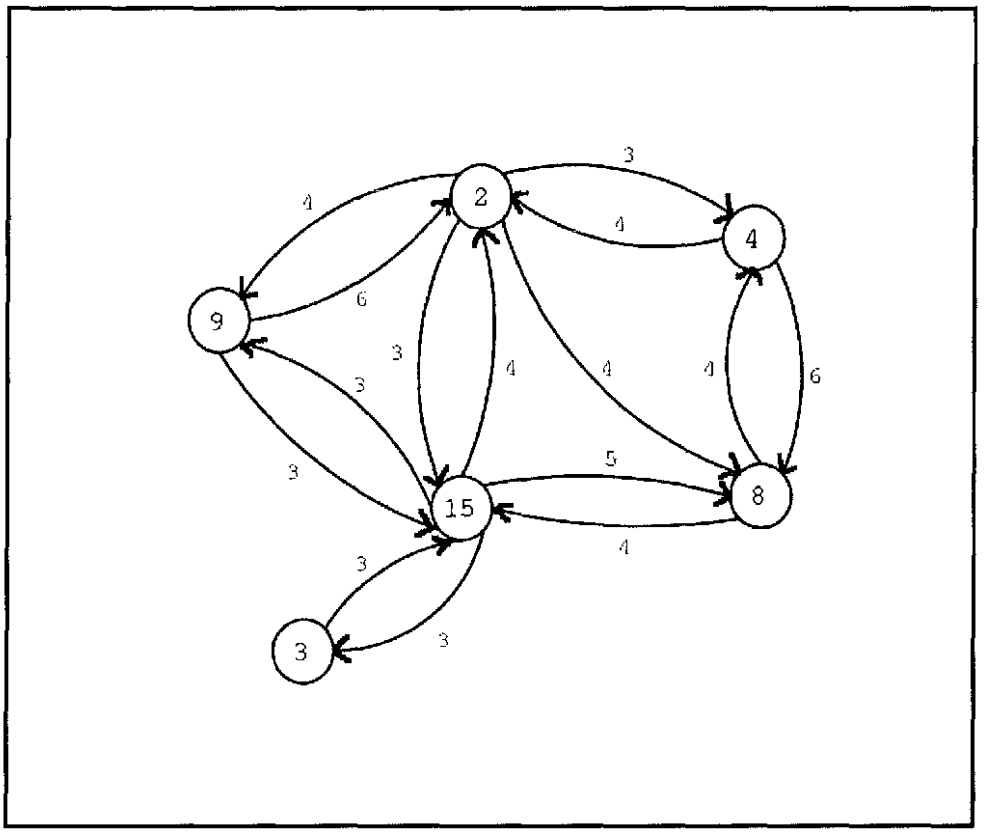
Map-15: Requirements-specifications / testing and debugging

APPENDIX I

Keyword Analysis Plots



Coupling-Cohesion Plot for Keyword Data



Supernetwork for Keyword Data

APPENDIX J

Sorted Index of Title Terms

20 teaching
20 year
19 environment
19 foundation
19 legacy
19 monitoring
19 part
19 style
18 calculus
18 learning
18 multimedia
18 security
18 visual basic
18 window
18 workflow
17 building
17 journal
17 object-oriented software
17 petri net
17 poster
17 software maintenance
17 user
16 code
16 collaboration
16 computer science
16 configuration
16 evolution
16 pointer
16 prediction
16 programmer
16 real-time system
16 representation
16 simulation
16 software system
16 tutorial
16 understanding
15 case
15 cost
15 empirical study
15 impact
15 lesson
15 panel
15 problem
15 product
15 programming language
15 software quality
15 software reliability
15 space
14 &mdash
14 consistency
14 fortran
14 net
14 perspective
14 resource
14 software reuse
14 student
13 abstract
13 complex
13 effect
13 enterprise
13 future
13 hypermedia
13 infrastructure
13 microsoft
13 network
13 object-oriented programming
13 report
13 solution
13 survey
13 visualization
13 web
13 world
12 abstraction
12 algorithm
12 comparison
12 concurrent program
12 defect
12 more
12 overview
12 parallel program
12 role
11 correctness
11 development process
11 editorial
11 extension
11 guide
11 hardware
11 investigation
11 platform
11 principle
11 procedure
11 production
11 reasoning
11 reference
11 reusability
11 risk
11 scheduling
11 selection
11 software engineering education
11 synchronization
11 workshop session
10 active
10 com
10 configuration management
10 diagram
10 distributed object
10 domain
10 editor
10 exception
10 hierarchy
10 image
10 inspection
10 library
10 mechanism
10 message

10 mobile agent
10 object-oriented design
10 paradigm
10 performance analysis
10 rapid proto-typing
10 relation-ship
10 research
10 reverse
10 review
10 suite
10 test
10 theory
10 training
10 tutorial session
10 use case
10 verifying
10 way
9 access
9 alternative
9 apl
9 challenge
9 client
9 comprehen-sion
9 definition
9 design , implement-ation
9 effective-ness
9 embedded system
9 fault
9 generator
9 java program
9 object technology
9 portable
9 poster session
9 programming environ-ment
9 proof
9 recovery
9 repository
9 scheme
9 software component
9 software process improve-ment
9 software project
9 software testing
9 source
9 stl
9 type
9 version
9 viewpoint
9 vs
9 work
8 application develop-ment
8 benefit
8 business process
8 case tool
8 class-ification
8 curriculum
8 efficiency
8 effort
8 failure
8 feature
8 formalism
8 handling
8 instrument-ation
8 linux
8 middleware
8 partial evaluation
8 performance evaluation
8 power
8 productivity
8 retrieval
8 robust
8 software engineer
8 software perform-ance
8 software tool
8 template
8 toolkit
8 y2k
7 3rd ed
7 agent
7 allocation
7 box
7 broker
7 bug
7 collection
7 commentary
7 company
7 component-based system
7 composition
7 computer
7 cost estimation
7 cot
7 depend-ability
7 editorial pointer
7 example
7 execution
7 experience report
7 experiment
7 extraction
7 forum
7 industry
7 inter-oper-ability
7 monitor
7 object-oriented system
7 opportunity
7 panel session

7 path
7 pattern language
7 perl
7 priority
7 protocol
7 query
7 race
7 refinement
7 reusable software
7 reverse engineering
7 search
7 software evolution
7 software product line
7 specification language
7 time
7 timing
7 transaction
7 tuning
7 virtual environment
7 writing
6 abstract interpretation
6 adaptation
6 analyzer
6 animation
6 architectural style
6 aspect
6 assignment
6 cluster
6 componentware
6 concurrent system
6 conflict
6 contribution
6 customer
6 distance
6 engineer
6 exploration
6 fault-tolerant
6 formal approach
6 formal verification
6 function
6 guideline
6 individual
6 inference
6 integrity
6 interaction detection
6 invariant
6 loto
6 manipulation
6 mapping
6 migration
6 mpi
6 object model
6 object-z
6 organization
6 paper
6 parallelization
6 partitioning
6 portability
6 predicate
6 presentation
6 primer
6 reduction
6 requirement specification
6 restructuring
6 scalability
6 simple
6 software design
6 software developer
6 statechart
6 task
6 tcl
6 timed petri net
6 tk
6 translation
6 unit
6 unix
6 visualizing
5 acm
5 action
5 algorithms
5 application framework
5 automata
5 automating
5 benchmark
5 business object
5 characteristic
5 characterization
5 codesign
5 compiler
5 computation
5 computer programming
5 concept analysis
5 constraint
5 conversion
5 coupling
5 crisis
5 delivery
5 delphi
5 dependency
5 deployment
5 development project
5 distributed system
5 education
5 empirical analysis
5 feature interaction
5 field
5 goal
5 good

5 ieee transactions on software engineering
5 implication
5 information system
5 innovation
5 internet application
5 iso
5 iterator
5 laboratory
5 legacy system
5 load
5 measure
5 meta-computing
5 mobility
5 note
5 novel
5 object-oriented program
5 powerbuilder
5 practical guide
5 practical programmer
5 presence
5 process model
5 program analysis
5 progress
5 propagation
5 question
5 reachability
5 reality
5 reflection
5 regression testing
5 response
5 reusable software component
5 rule
5 safety
5 scenario
5 schemas
5 simplicity
5 software configuration management
5 software cost
5 software engineering research
5 software inspection
5 structure
5 system design
5 technical communication
5 tip
5 transition
5 view
5 visual c++
5 visual language
5 web site
5 workbench
5 workflow management
5 world wide web
5 xml

APPENDIX K

CAIR LM File for Titles

```
Run Parameters:  Eliminate by Nodes
Pass Two Node Filter: Both nodes.  Link Selection:
Strength and Max. Nodes.  Min. Strength: 0.000000.
Min. Co-Occurrence: 5.  Max links: 12.  Max maps
100.  Max nodes 10.  Maps Produced: 16

1 5 4
^detection^ 29 1 3 1 1
^feature interaction^ 5 1 2 1 1
^interaction detection^ 6 1 1 1 1
^race^ 7 1 1 1 1
^interaction^ 38 1 1 1 1
^feature interaction^ ^interaction detection^ 5
0.833333 1 0
^detection^ ^race^ 6 0.177340 1 0
^detection^ ^feature interaction^ 5 0.172414 1 0
^detection^ ^interaction^ 6 0.032668 1 0
0.303939 0.000000 0.000000 16 12

2 2 1
^tk^ 6 1 1 1 2
^tcl^ 6 1 1 1 2
^tcl^ ^tk^ 5 0.694444 1 0
0.694444 0.000000 0.000000 6 5

3 2 1
^exception^ 10 1 1 1 3
^handling^ 8 1 1 1 3
^exception^ ^handling^ 6 0.450000 1 0
0.450000 0.000000 0.000000 4 6

4 2 1
^server^ 20 1 1 1 4
^client^ 9 1 1 1 4
^client^ ^server^ 8 0.355556 1 0
```

```
0.355556 0.000000 0.000000 7 8

5 20 24
^analysis^ 213 5 12 1 5
^performance^ 86 3 6 1 5
^design^ 119 3 5 1 5
^software^ 236 6 3 2 8
^evaluation^ 68 1 3 1 5
^pointer^ 16 1 2 1 5
^petri^ 46 1 2 1 5
^program^ 132 2 2 2 15
^approach^ 145 4 2 2 15
^design , implementation^ 9 1 1 1 5
^technique^ 72 2 1 2 13
^testing^ 92 2 1 2 15
^editorial^ 11 1 1 1 5
^real-time^ 49 2 1 2 12
^model^ 49 2 1 2 15
^method^ 71 3 1 2 16
^application^ 187 5 1 2 12
^study^ 96 4 1 2 12
^measurement^ 25 1 1 1 5
^net^ 17 1 1 1 5
^editorial^ ^pointer^ 7 0.278409 1 0
^net^ ^petri^ 14 0.250639 1 0
^design^ ^design , implementation^ 9 0.075630 1 0
^analysis^ ^pointer^ 8 0.018779 1 0
^analysis^ ^performance^ 16 0.013975 1 0
^evaluation^ ^performance^ 9 0.013851 1 0
^analysis^ ^petri^ 11 0.012349 1 0
^measurement^ ^performance^ 5 0.011628 1 0
^design^ ^evaluation^ 9 0.010010 1 0
^analysis^ ^design^ 12 0.005681 3 0
^analysis^ ^program^ 15 0.008003 2 15
^analysis^ ^testing^ 10 0.005103 2 15
^performance^ ^program^ 7 0.004316 2 15
^analysis^ ^software^ 14 0.003899 2 8
^analysis^ ^real-time^ 6 0.003449 2 12
^analysis^ ^model^ 6 0.003449 2 15
^design^ ^method^ 5 0.002959 2 16
^approach^ ^performance^ 6 0.002887 2 15
^analysis^ ^approach^ 9 0.002623 2 15
```

```
^analysis^ ^study^ 7 0.002396 2 12
^performance^ ^software^ 6 0.001774 2 8
^analysis^ ^technique^ 5 0.001630 2 13
^application^ ^design^ 6 0.001618 2 12
^evaluation^ ^software^ 5 0.001558 2 8
0.069095 0.045664 0.000188 393 168

6 2 1
^estimation^ 21 1 1 1 6
^effort^ 8 1 1 1 6
^effort^ ^estimation^ 6 0.214286 1 0
0.214286 0.000000 0.000000 3 6

7 2 1
^software process^ 21 1 1 1 7
^improvement^ 23 1 1 1 7
^improvement^ ^software process^ 9 0.167702 1 0
0.167702 0.000000 0.000000 10 9

8 19 24
^software^ 236 6 10 1 8
^engineering^ 107 3 9 1 8
^software engineering^ 122 2 6 1 8
^methodology^ 20 2 3 1 8
^science^ 30 2 3 1 8
^ieee transaction^ 5 1 2 1 8
^application^ 187 5 2 2 12
^approach^ 145 4 2 2 15
^software development^ 46 2 1 2 12
^roadmap^ 25 1 1 1 8
^configuration^ 16 2 1 2 11
^development^ 95 2 1 2 12
^ieee transaction on software engineering^ 5 1 1 1 8
^project^ 39 2 1 2 11
^programming^ 198 3 1 2 15
^analysis^ 213 5 1 2 5
^study^ 96 4 1 2 12
^reverse^ 10 1 1 1 8
^workshop session^ 11 1 1 1 8
^engineering^ ^software engineering^ 41 0.128773 1 0
^roadmap^ ^software engineering^ 13 0.055410 1 0
^engineering^ ^ieee transaction^ 5 0.046729 1 0
```

```
^ieee transaction^ ^software engineering^ 5 0.040984
1 0
^ieee transaction on software engineering^ ^software
engineering^ 5 0.040984 1 0
^engineering^ ^software^ 31 0.038056 1 0
^engineering^ ^reverse^ 5 0.023364 1 0
^engineering^ ^workshop session^ 5 0.021240 1 0
^engineering^ ^science^ 8 0.019938 1 0
^engineering^ ^methodology^ 6 0.016822 1 0
^science^ ^software engineering^ 7 0.013388 3 0
^software^ ^software engineering^ 14 0.006807 3 0
^methodology^ ^software^ 5 0.005297 3 0
^configuration^ ^software^ 7 0.012977 2 11
^development^ ^software^ 14 0.008742 2 12
^application^ ^methodology^ 5 0.006684 2 12
^project^ ^software^ 7 0.005324 2 11
^programming^ ^science^ 5 0.004209 2 15
^analysis^ ^software^ 14 0.003899 2 5
^software^ ^study^ 9 0.003575 2 12
^approach^ ^engineering^ 7 0.003158 2 15
^approach^ ^software^ 10 0.002922 2 15
^application^ ^engineering^ 7 0.002449 2 12
^software^ ^software development^ 5 0.002303 2 12
0.035215 0.056242 0.000393 365 152

9 2 1
^experience^ 45 1 1 1 9
^report^ 13 1 1 1 9
^experience^ ^report^ 8 0.109402 1 0
0.109402 0.000000 0.000000 29 8

10 2 1
^reliability^ 21 1 1 1 10
^software reliability^ 15 1 1 1 10
^reliability^ ^software reliability^ 5 0.079365 1 0
0.079365 0.000000 0.000000 13 5

11 4 5
^management^ 55 1 3 1 11
^software^ 236 6 3 2 8
^configuration^ 16 2 2 1 11
^project^ 39 2 2 1 11
```

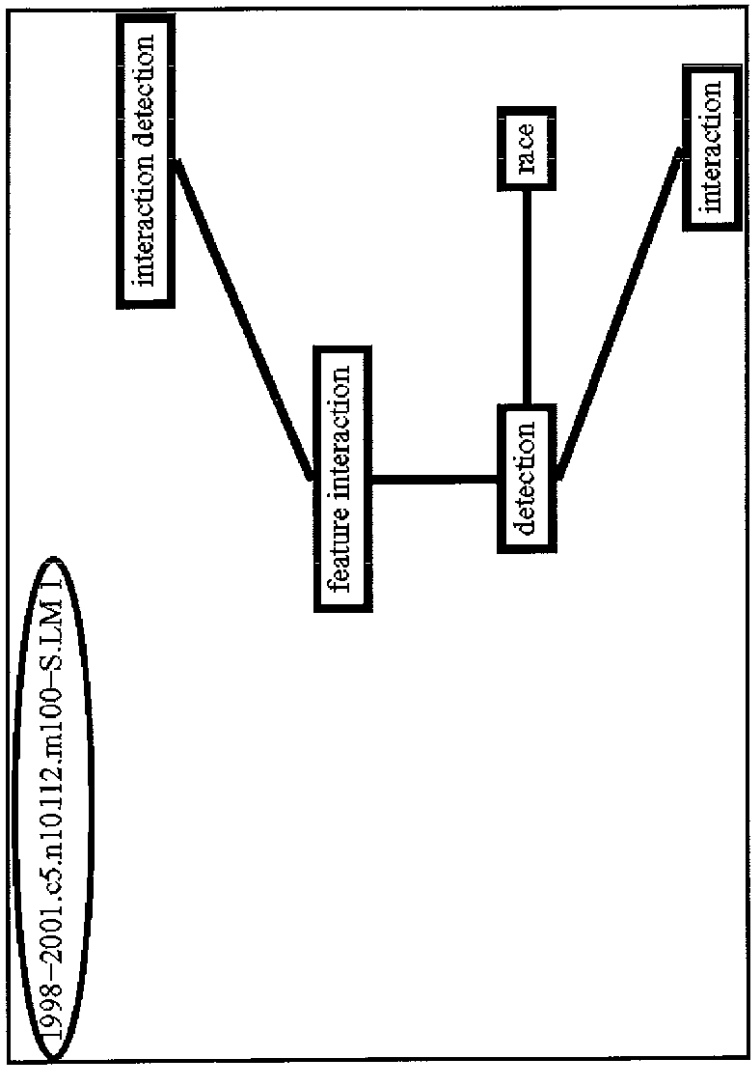
```
^configuration^ ^management^ 6 0.040909 1 0
^management^ ^project^ 5 0.011655 1 0
^configuration^ ^software^ 7 0.012977 2 8
^project^ ^software^ 7 0.005324 2 8
^management^ ^software^ 5 0.001926 2 8
0.026282 0.020227 0.000200 48 22

12 20 24
^application^ 187 5 14 1 12
^study^ 96 4 6 1 12
^development^ 95 2 4 1 12
^software^ 236 6 4 2 8
^real-time^ 49 2 3 1 12
^analysis^ 213 5 2 2 5
^software development^ 46 2 2 1 12
^approach^ 145 4 1 2 15
^tool^ 46 2 1 2 16
^method^ 71 3 1 2 16
^engineering^ 107 3 1 2 8
^design^ 119 3 1 2 5
^performance^ 86 3 1 2 5
^real-time system^ 16 1 1 1 12
^internet^ 30 1 1 1 12
^framework^ 47 1 1 1 12
^network^ 13 1 1 1 12
^metric^ 21 1 1 1 12
^programming^ 198 3 1 2 15
^methodology^ 20 2 1 2 8
^real-time^ ^real-time system^ 5 0.031888 1 0
^development^ ^software development^ 10 0.022883 1 0
^metric^ ^study^ 6 0.017857 1 0
^application^ ^internet^ 8 0.011408 1 0
^application^ ^framework^ 10 0.011378 1 0
^application^ ^network^ 5 0.010284 1 0
^development^ ^study^ 9 0.008882 1 0
^application^ ^development^ 12 0.008106 1 0
^application^ ^real-time^ 7 0.005348 1 0
^application^ ^study^ 5 0.001393 3 0
^development^ ^software^ 14 0.008742 2 8
^application^ ^programming^ 16 0.006914 2 15
^application^ ^methodology^ 5 0.006684 2 8
^application^ ^approach^ 13 0.006233 2 15
```

```
^application^ ^tool^ 7 0.005696 2 16
^method^ ^study^ 5 0.003668 2 16
^software^ ^study^ 9 0.003575 2 8
^analysis^ ^real-time^ 6 0.003449 2 5
^application^ ^engineering^ 7 0.002449 2 8
^analysis^ ^study^ 7 0.002396 2 5
^software^ ^software development^ 5 0.002303 2 8
^application^ ^design^ 6 0.001618 2 5
^application^ ^performance^ 5 0.001555 2 5
^application^ ^software^ 8 0.001450 2 8
0.012943 0.056733 0.000303 464 142

13 3 2
^technique^ 72 2 2 1 13
^comparison^ 12 1 1 1 13
^analysis^ 213 5 1 2 5
^comparison^ ^technique^ 5 0.028935 1 0
^analysis^ ^technique^ 5 0.001630 2 5
0.028935 0.001630 0.000003 81 10

14 2 1
^interface^ 28 1 1 1 14
^user interface^ 33 1 1 1 14
^interface^ ^user interface^ 5 0.027056 1 0
0.027056 0.000000 0.000000 20 5

15 17 21
^approach^ 145 4 8 1 15
^program^ 132 2 6 1 15
^programming^ 198 3 4 1 15
^analysis^ 213 5 4 2 5
^verification^ 64 1 3 1 15
^testing^ 92 2 3 1 15
^model^ 49 2 2 1 15
^performance^ 86 3 2 2 5
^application^ 187 5 2 2 12
^specification^ 39 1 1 1 15
^property^ 21 1 1 1 15
^parallel^ 40 1 1 1 15
^2nd ed^ 22 1 1 1 15
^science^ 30 2 1 2 8
^engineering^ 107 3 1 2 8
```

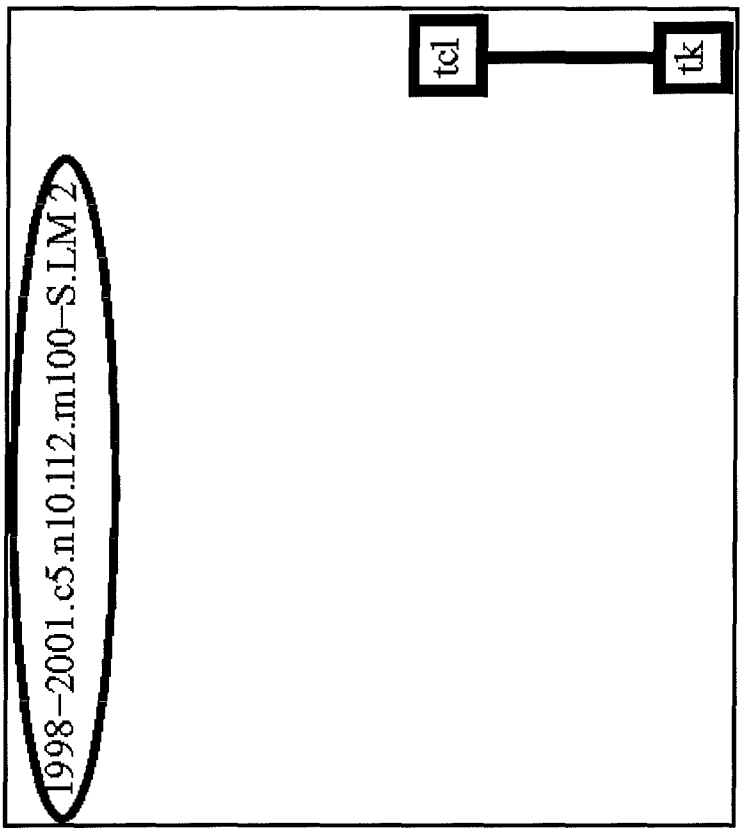
```
^software^ 236 6 1 2 8
^software engineering^ 122 2 1 2 8
^specification^ ^verification^ 8 0.025641 1 0
^property^ ^verification^ 5 0.018601 1 0
^2nd ed^ ^programming^ 8 0.014692 1 0
^parallel^ ^program^ 8 0.012121 1 0
^program^ ^verification^ 6 0.004261 1 0
^model^ ^program^ 5 0.003865 1 0
^program^ ^testing^ 6 0.002964 1 0
^approach^ ^programming^ 9 0.002821 1 0
^approach^ ^testing^ 6 0.002699 1 0
^analysis^ ^program^ 15 0.008003 2 5
^application^ ^programming^ 16 0.006914 2 12
^application^ ^approach^ 13 0.006233 2 12
^analysis^ ^testing^ 10 0.005103 2 5
^performance^ ^program^ 7 0.004316 2 5
^programming^ ^science^ 5 0.004209 2 8
^analysis^ ^model^ 6 0.003449 2 5
^approach^ ^engineering^ 7 0.003158 2 8
^approach^ ^software^ 10 0.002922 2 8
^approach^ ^performance^ 6 0.002887 2 5
^analysis^ ^approach^ 9 0.002623 2 5
^approach^ ^software engineering^ 5 0.001413 2 8
0.009741 0.051230 0.000261 434 141

16 6 5
^method^ 71 3 4 1 16
^tool^ 46 2 2 1 16
^application^ 187 5 1 2 12
^study^ 96 4 1 2 12
^design^ 119 3 1 2 5
^software^ 236 6 1 2 8
^method^ ^tool^ 6 0.011023 1 0
^application^ ^tool^ 7 0.005696 2 12
^method^ ^study^ 5 0.003668 2 12
^design^ ^method^ 5 0.002959 2 5
^method^ ^software^ 5 0.001492 2 8
0.011023 0.013815 0.000057 222 26
```

APPENDIX L

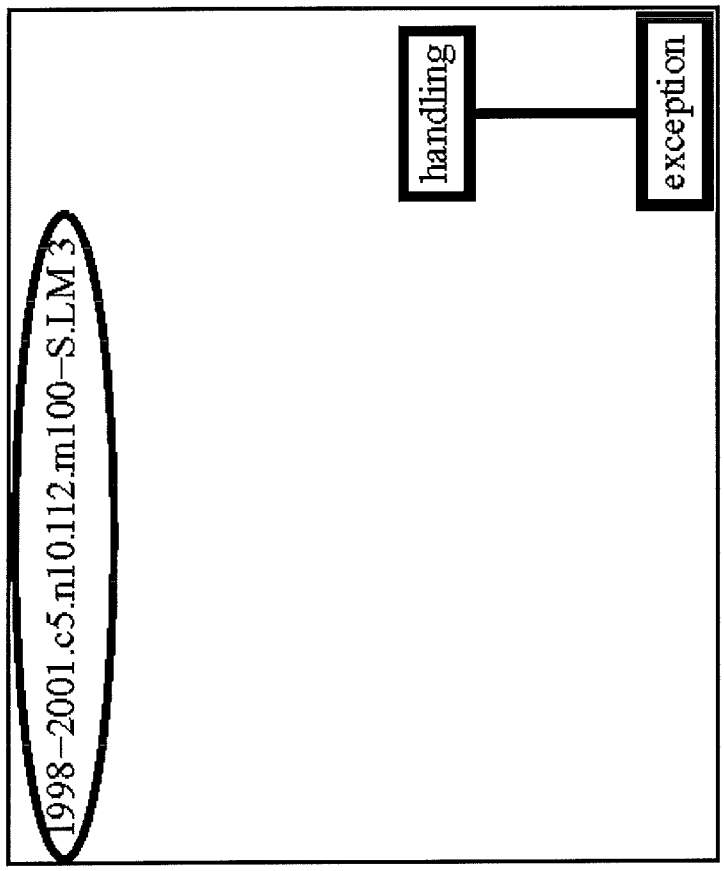
Title Maps



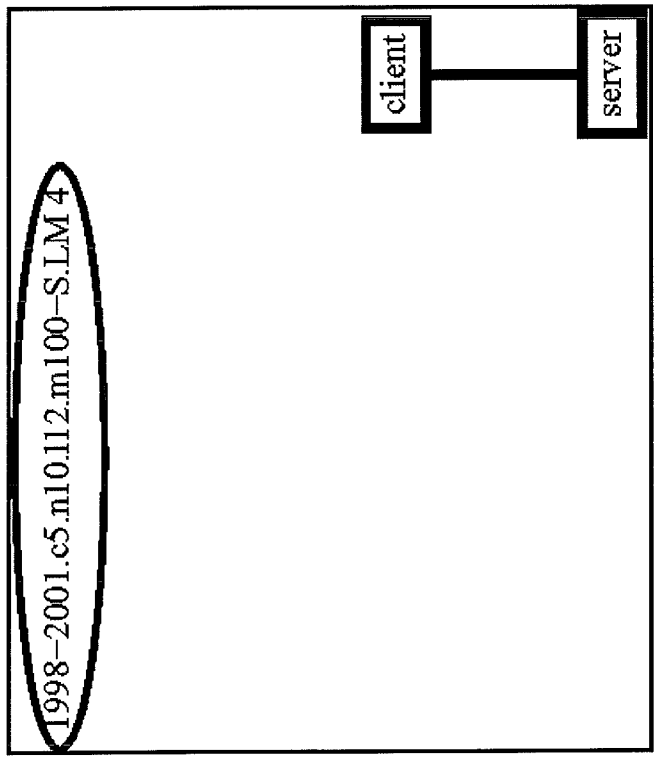
Map-1: Interaction-Detection



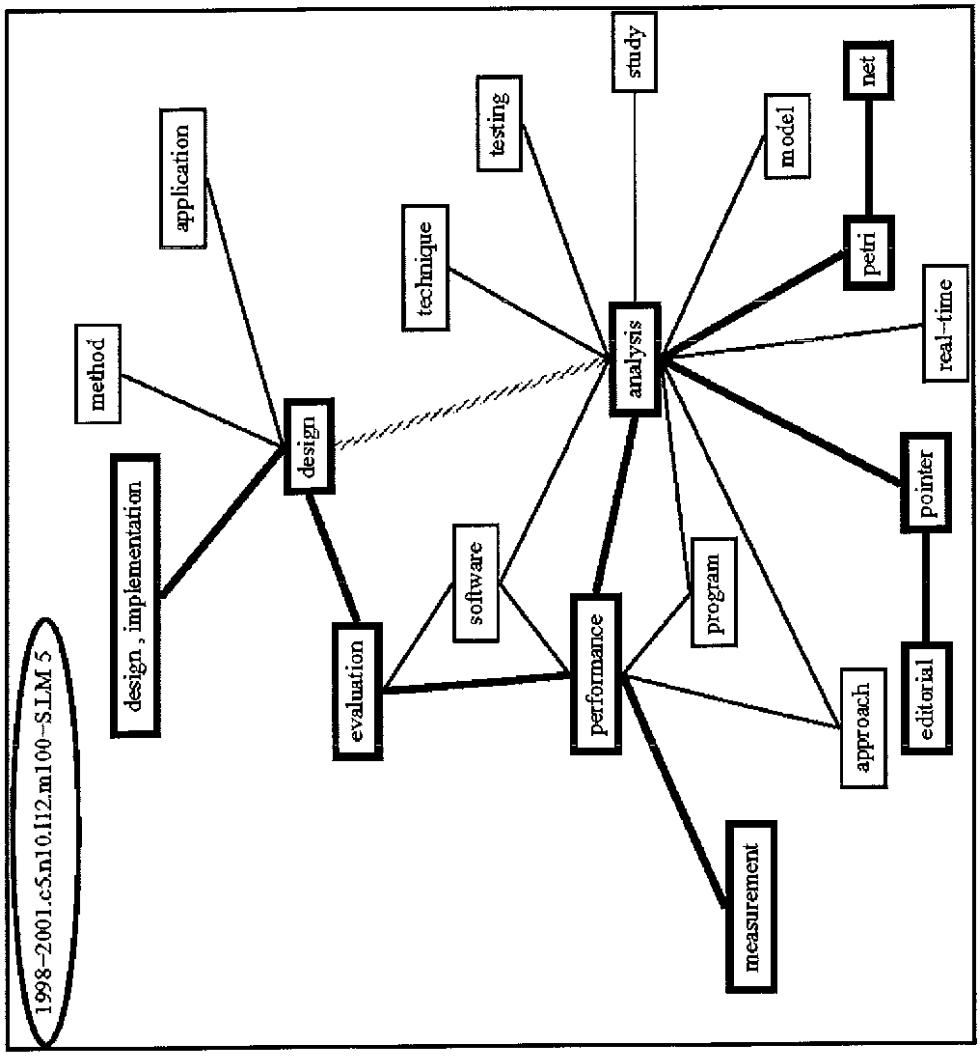
Map-2: TCL-TK



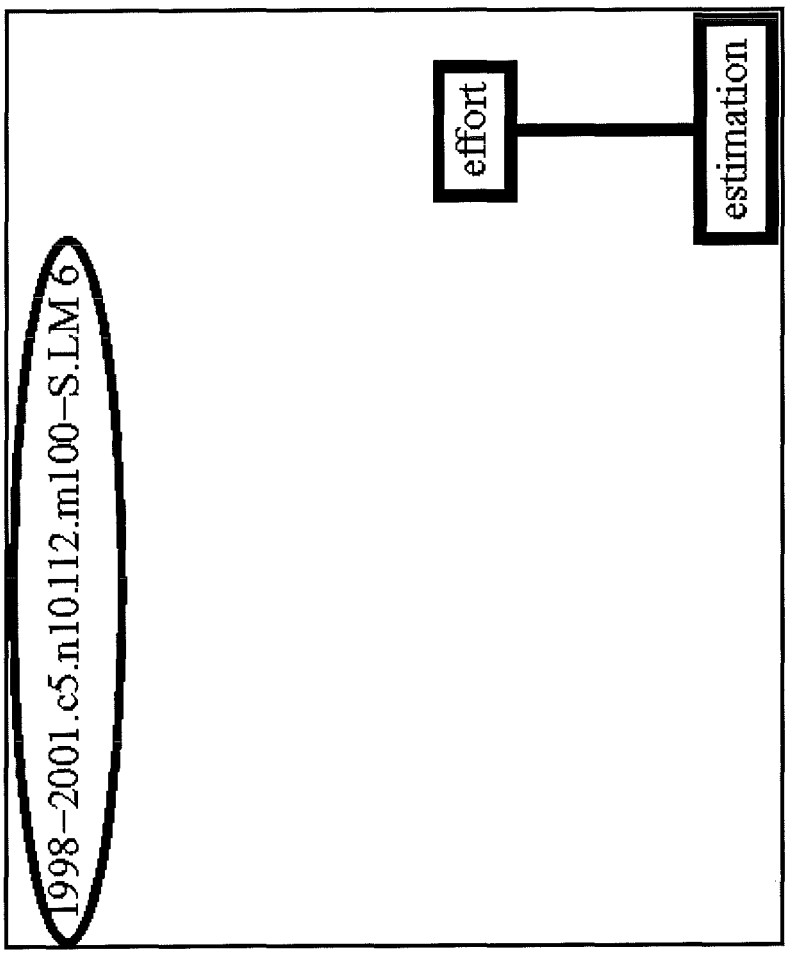
Map-3: Exception-Handling



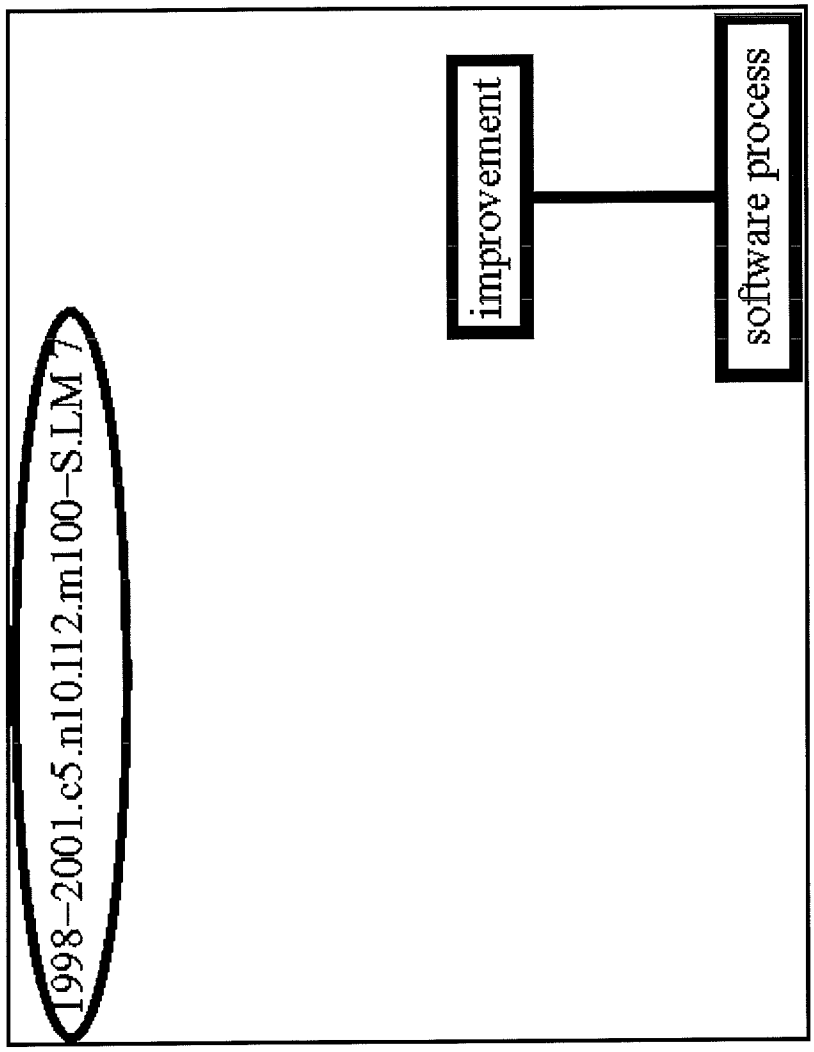
Map-4: Client-Server



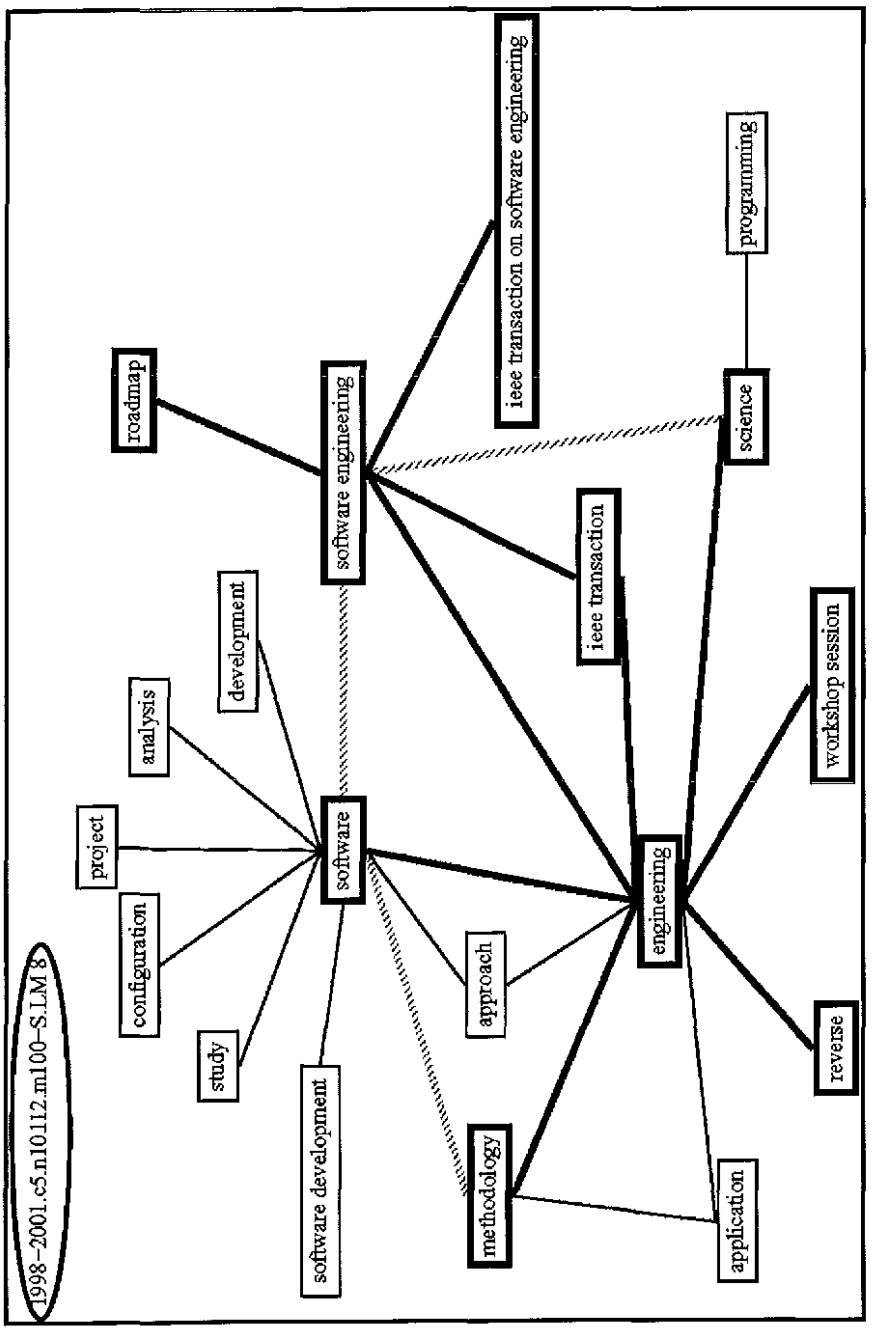
Map-5: Analysis - Performance



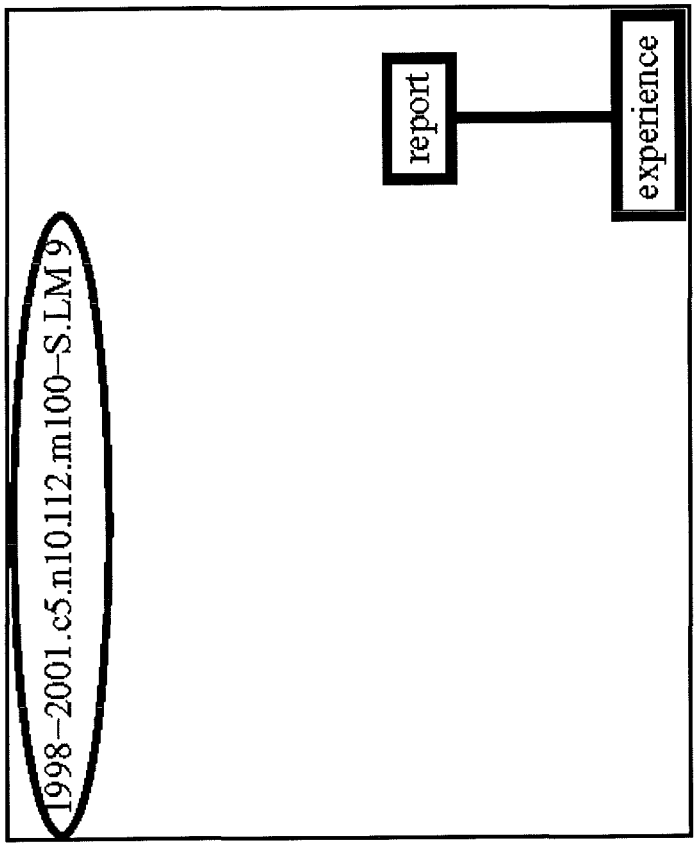
Map-6: Effort - Estimation



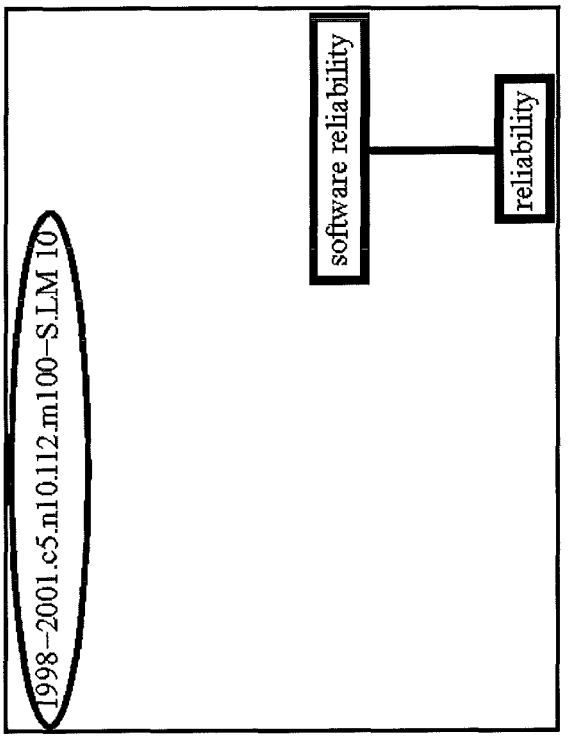
Map-7: Software Process - Improvement



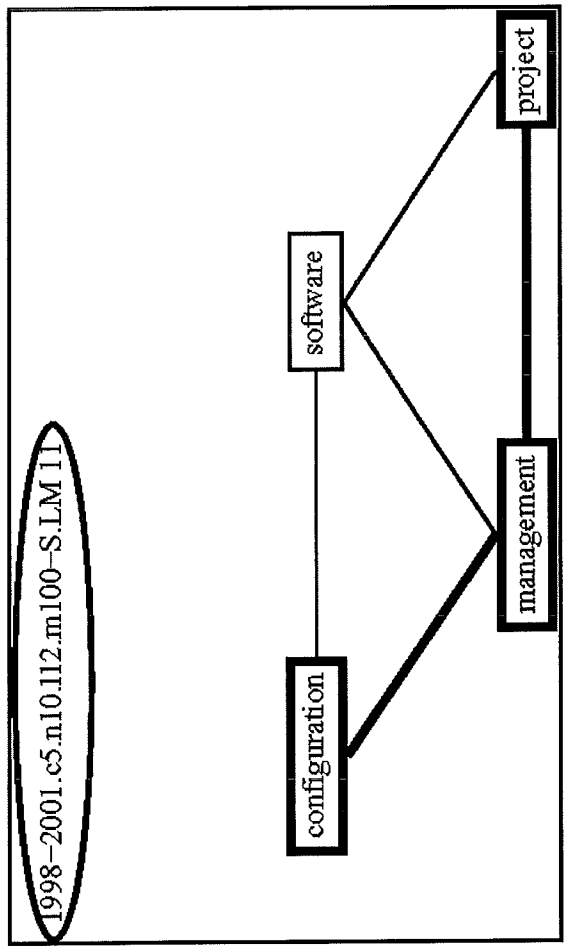
Map-8: Software Engineering



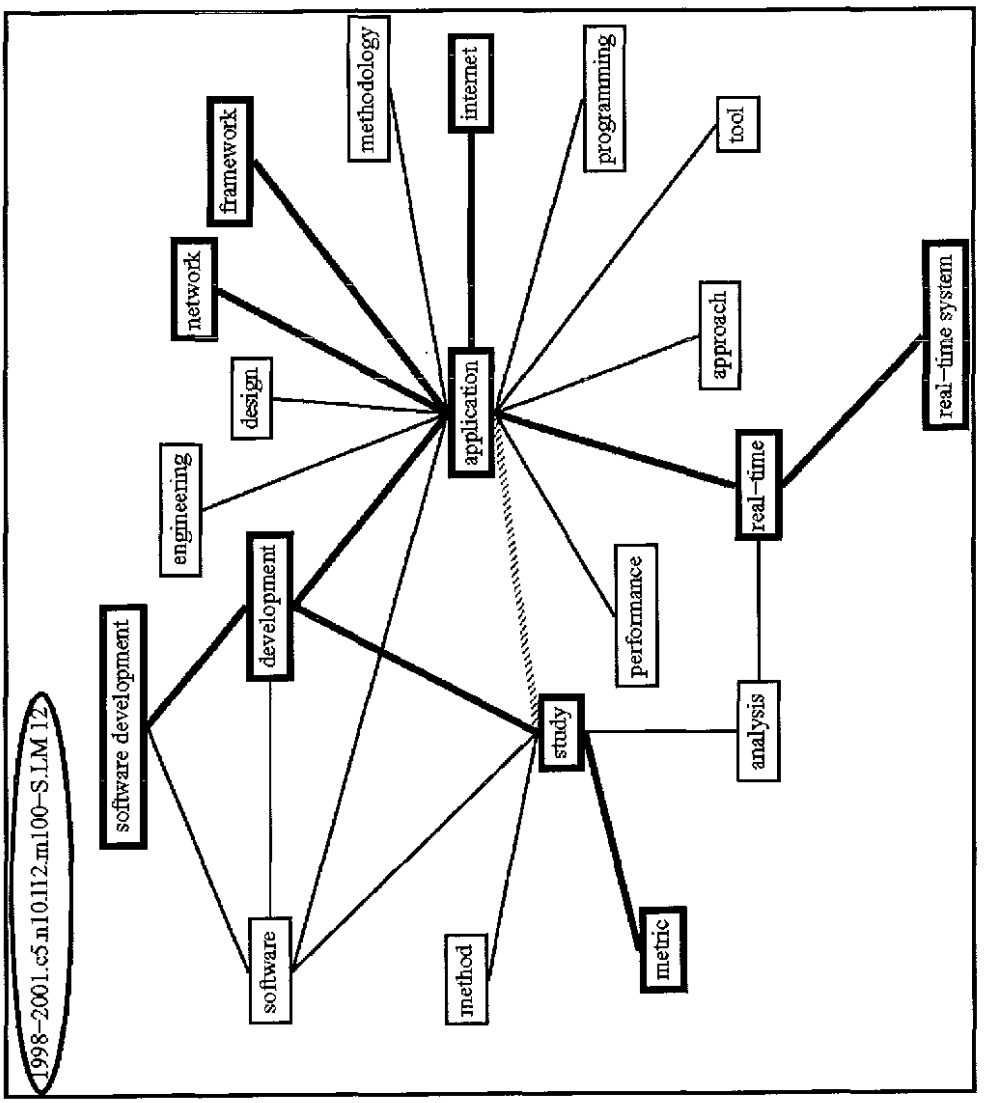
Map-9: Report -Experience



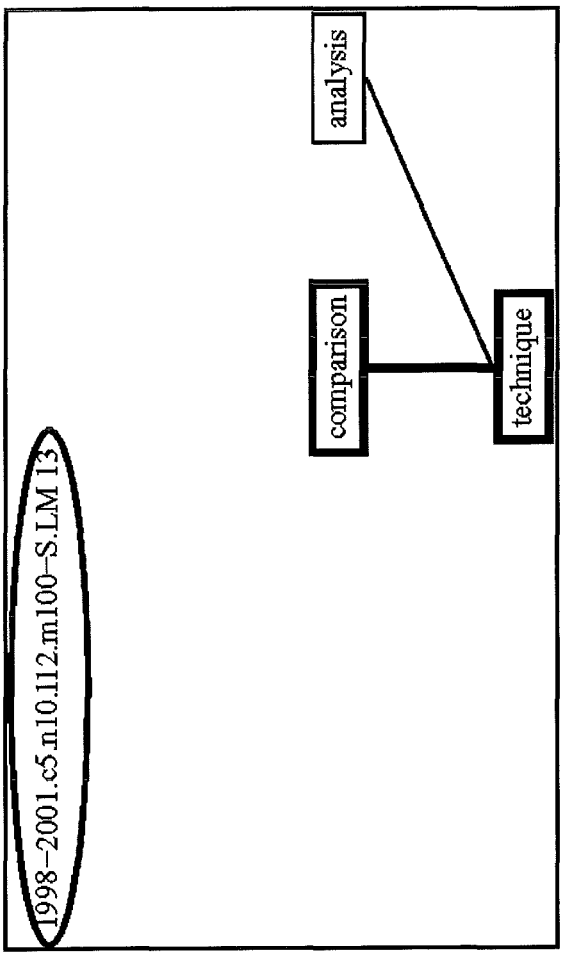
Map-10: Software Reliability



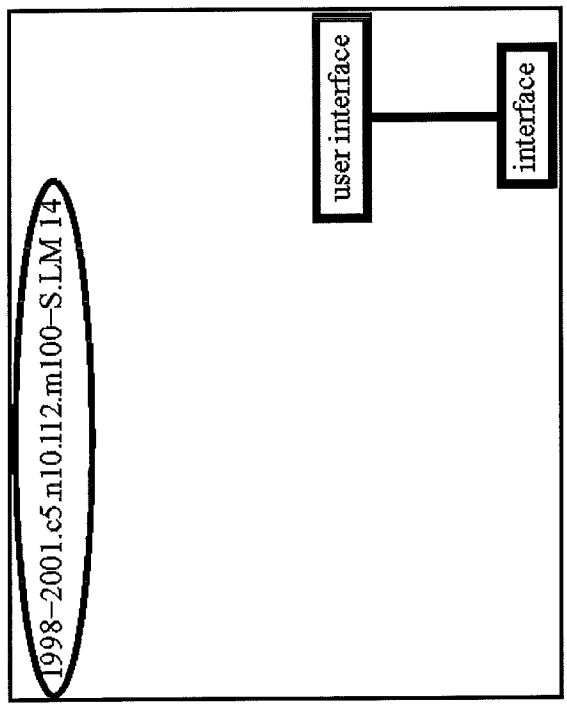
Map-11: Project - Management



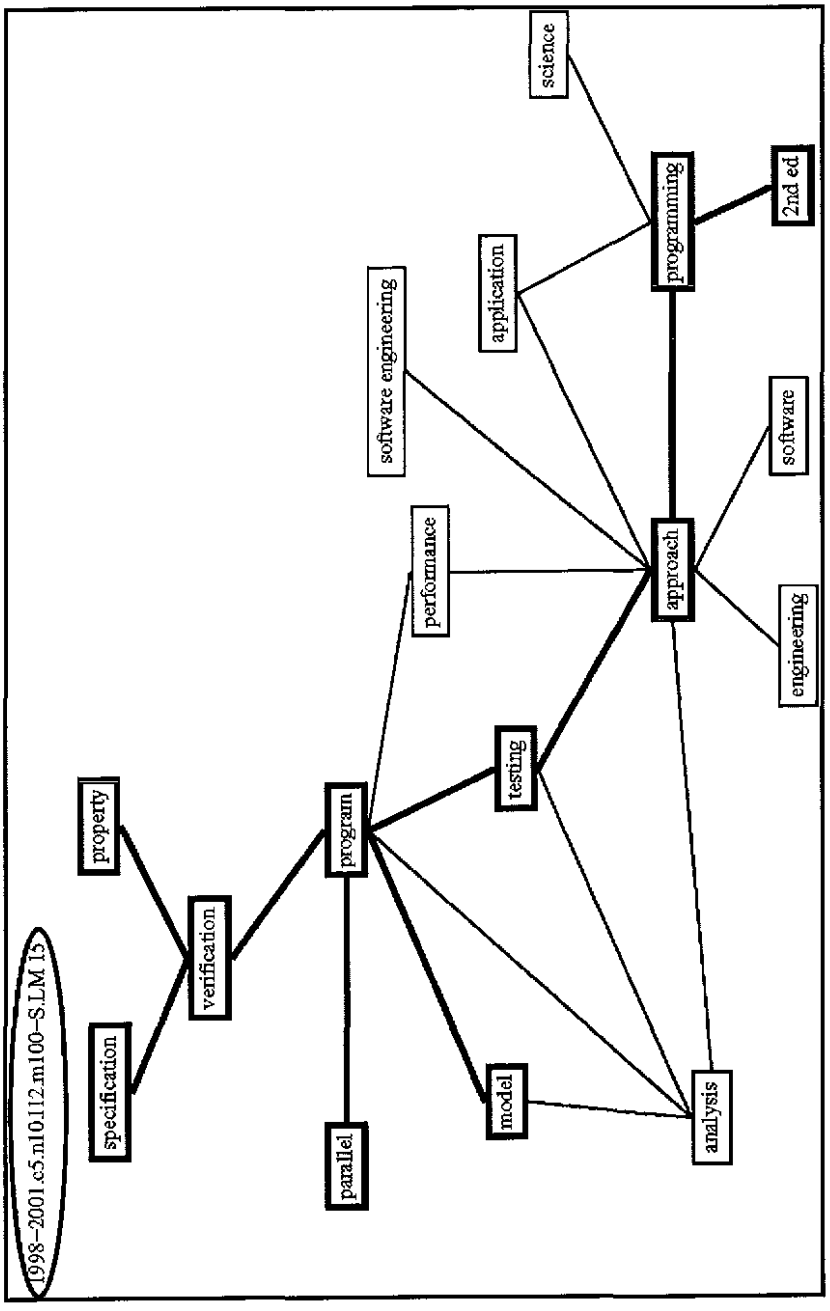
Map-12: Application - Development



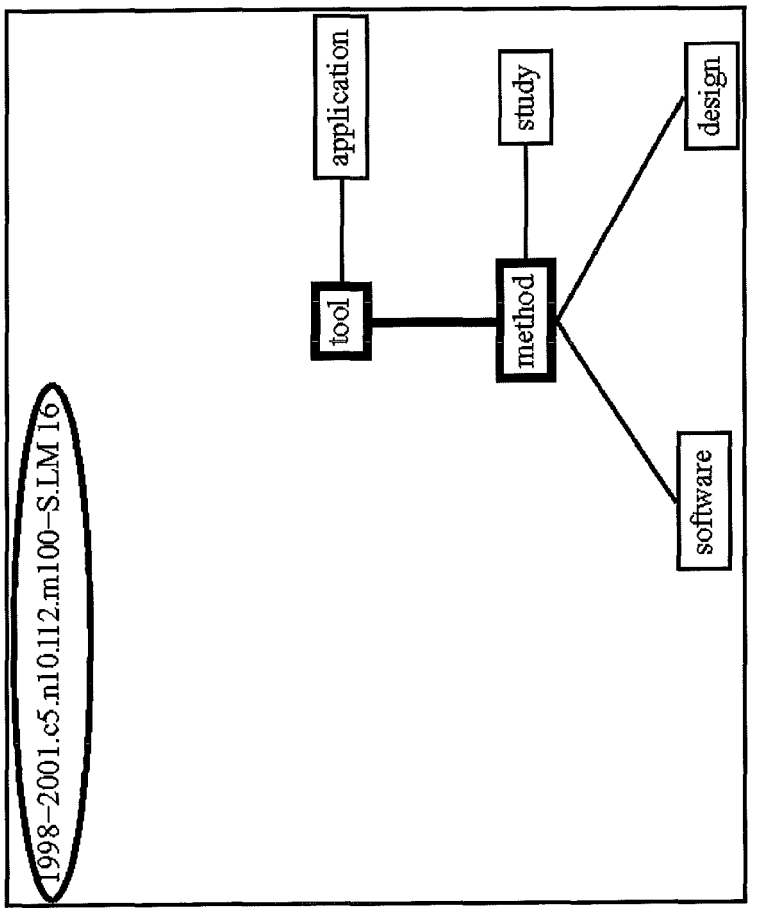
Map-13: Comparison - Technique



Map-14: User Interface



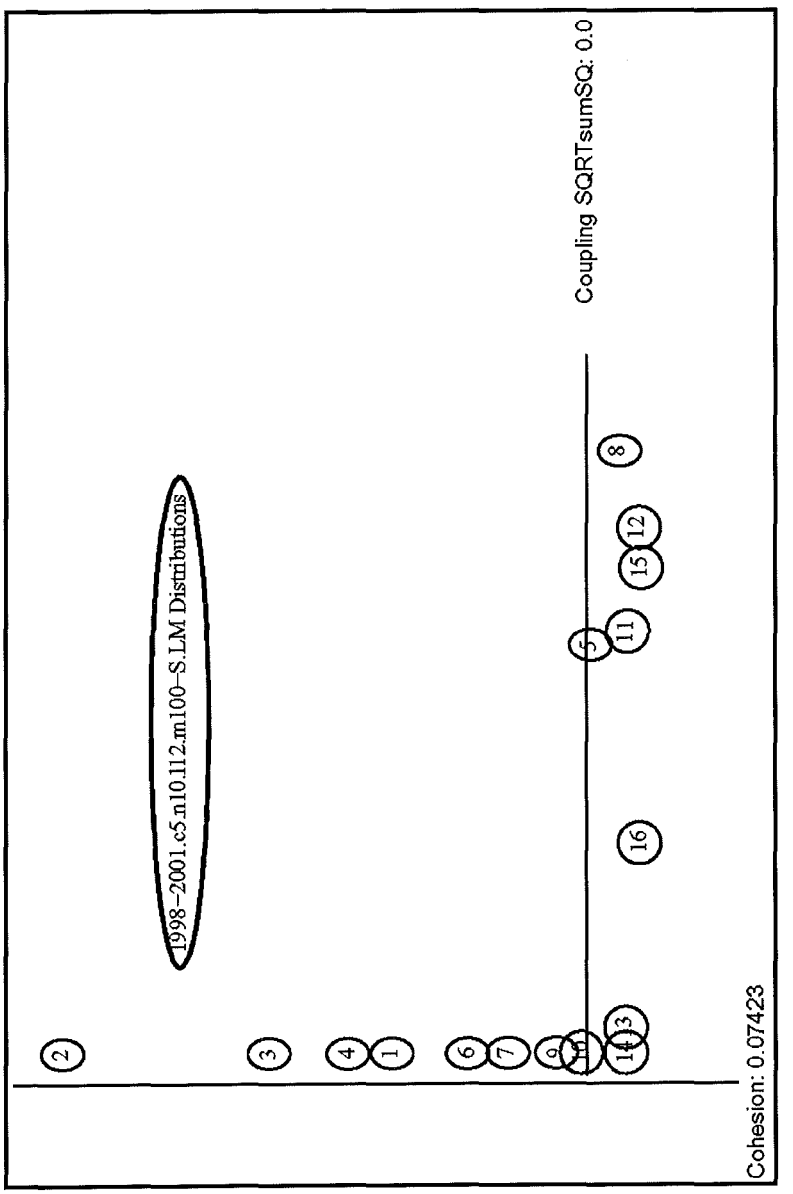
Map-15: Program - Verification



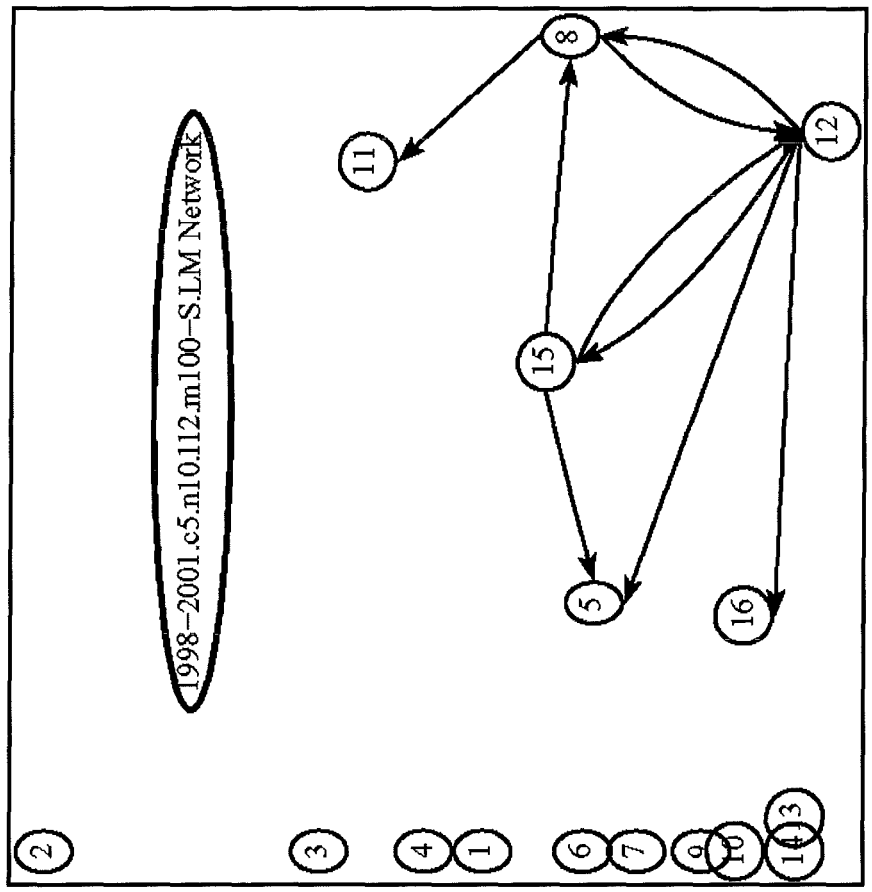
Map-16: Method - Tool

APPENDIX M

Title Analysis Plots



Title Coupling-Cohesion Plot



Title Supernetwork

REFERENCES

[CALLON86]
Callon, M., et al., Mapping of the Dynamics of Science and Technology, MacMillan, London, 1986.

[CALLON91]
Callon, M., et al., "Co-word Analysis as a Tool for Describing the Network of Interactions between Basic and Technological Research: The Case of Polymer Chemistry," Scientometrics 22, 1 (1991), pp. 153-203.

[CHAPMAN97]
Chapman, N., "Petri Net Models," SURPRISE 97: Surveys and Presentations in Information Systems Engineering, Imperial College of Science Technology and Medicine, London, UK, (May - June 1997).

[COULTER91]
Coulter, N. S., "Changes to the CR Classification System," Computing Reviews 32, 1 (January 1991), pp. 7-10.

[COULTER96]
Coulter, N. S. and I. Monarch, "Best Practices: What Software Engineering Can Learn from Manufacturing Engineering," SEI Software Engineering Symposium, Pittsburg, September 1996, pp. 7-8.

[COULTER98A]
Coulter, N. S., "ACM's Computing Classification System Reflects Changing Times," Communications of the ACM 40, 12 (December 1997), pp. 111-112.

[COULTER98B]
Coulter, N. S., et al., "Software Engineering as Seen through Its Research Literature: A Study in Co-Word Analysis," Journal of the American Society For Information Science 49, 13 (1998), pp. 1206-1223.

[COURTIAL89]
Courtial, J. P. and J. Law, "A Co-word Study of Artificial Intelligence," Social Studies in Science 19 (1989), pp. 301-311.

[HAMMOND99]
Hammond, T., et al., "CAIR-Prep," Team Project for CEN 4010: Principles of Software Engineering, taught by Dr. Neal S. Coulter, Florida Atlantic University, Boca Raton, Florida, 1998.

[KESSLER63]
Kessler, M., "Bibliographic Coupling between Scientific Papers," American Documentation 14 (1963), pp. 10-25.

[LAW92]
Law, J. and J. Whittaker, "Mapping Acidification Research: A Test of the Co-word Method," Scientometrics 23, 3 (1992), pp. 417-461.

[SAMMET82]
Sammet, J. E. and A. Ralston, "The New (1982) Computing Reviews Classification System - Final Version," Communications of the ACM 25, 1 (January 1982), pp. 13-25.

[SAMMET83]
Sammet, J. E., "Summary of Changes from 1982 to 1983 Version of CR Classification System," Computing Reviews 24, 1 (January 1983), pp. 7-8.

[SAMMET87]
Sammet, J. E., "Summary of Additions from 1983 to 1987 Version of CR Classification System," Computing Reviews 28, 1 (January 1987), pp. 5-6.

[SHAH97]
Shah, P. P., "Content Analysis of Design Practices," Master's thesis, College of Engineering, Florida Atlantic University, Boca Raton, 1997.

[SMALL73]
Small, H., "Co-citation in the Scientific Literature: A New Measure of the Relationships between Documents," Journal of the American Society for Information Science 24 (1973), pp. 265-269.

[WHITTAKER89]
Whittaker, J., "Creativity and Conformity in Science: Titles, Keywords, and Co-word Analysis," Social Science in Science 19 (1989), pp. 473-496.

VITA

Terry Smith has a Bachelor of Science from the University of North Florida in Mathematics and Physics, 1989, and expects to receive a Master of Science in Computer and Information Sciences from the University of North Florida in April 2004. Dr. Neal Coulter of the University of North Florida is serving as Mr. Smith's thesis advisor.

Mr. Smith is the Director of Information Technologies for the College of Computing, Engineering, and Construction at the University of North Florida and has held this position since 2001. Mr. Smith has held a number of IT positions outside academia, but he is most proud of his years teaching at the University of North Florida.

Mr. Smith began his teaching career in 1989 as a teaching assistant in the Department of Mathematics

and Statistics. In 1991, he accepted the position of instructor of astronomy and physics in the Department of Natural Sciences. Later, from 1995 to 2000, he was an instructor in the Department of Computer and Information Sciences, where he taught a variety of freshman- through senior-level classes.

Mr. Smith has authored a number of published works, including printed books, journal articles, training manuals, a plethora of online reference materials, and one online book.

Mr. Smith has on-going interests in content analysis and its application to a wide variety of complex fields, including continued analysis of software engineering, astronomy, and linguistics.