P-1002

# ANALYST AID

PROJECT WORK DONE AT

**TATA CONSULTANCY SERVICES, TRIVANDRUM**

**PROJECT REPORT**

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE OF
**MASTER OF COMPUTER APPLICATIONS**
OF BHARATHIAR UNIVERSITY, COIMBATORE.

SUBMITTED BY

**NIHAS . A**
**0038M1043**

GUIDED BY

EXTERNAL GUIDE
**Dr. S . Neethi,**
Associate Consultant,
TATA Consultancy Services,
Trivandrum.

INTERNAL GUIDE
**Mr. A. Muthukumar, M.Phil.,**
Assistant Professor,
Dept. of Computer Science & Engg.,
Kumaraguru College of Technology,
Coimbatore.

Department of Computer Science and Engineering
Kumaraguru College of Technology
Coimbatore – 641006
April 2003.

**Department of Computer Science and Engineering**

**KUMARAGURU COLLEGE OF TECHNOLOGY**

COIMBATORE – 641 006.

## CERTIFICATE

This is to certify that the project work titled

**ANALYST AID**

Done by

**NIHAS. A.**

**0038M1043**

Submitted in partial fulfillment of the requirements for the award of the degree of

**Master of Computer Applications of Bharathiar University.**

[illegible]
Professor and Head

[illegible]
Internal Guide

Submitted to University Examination held on 16-04-2003.

[illegible]
Internal Examiner

[illegible]
External Examiner
16/4/03

TATA

04 April 2003

**CERTIFICATE**

This is to certify that **Mr Nihas A**, a student of MCA from Kumaraguru College of Technology, Coimbatore has undergone Project Training in our organization between 16 December 2002 to 30 March 2003

He has undertaken a project titled "**ANALYST AID**"

During his Project Training he has performed well and earned a name as a sincere and hardworking person.

We wish him all the best in his future endeavors.

For TATA CONSULTANCY SERVICES

[illegible]
( Dr. S. NEETHI)
ASSOCIATE CONSULTANT
PROJECT GUIDE

[illegible]
(PRAMOD CHANDRASEKHAR)
ASSISTANT MANAGER HR

**TATA** CONSULTANCY SERVICES
A Division of **Tata Sons Limited**
Technopark Campus, Kariyavattom P.O., Thiruvananthapuram - 695 581, India
Tel : 91 - 471 - 700671 Fax : 91 - 471 - 700682 E-mail : tictvm@tvm.tcs.co.in Website : www.tcs.com
Registered Office : Bombay House 24, Homi Mody Street, Mumbai 400 001

## DECLARATION

I hereby declare that this project entitled **ANALYST AID,** submitted to the Bharathiyar University as the project of Master of Computer Application Degree, is a record of original work done by me under the supervision and guidance of **Dr. S. Neethi**, Associate Consultant, TATA Consultancy Services, Trivandrum and **Mr. A. Muthukumar**, M.Phil., Course Co-ordinator, MCA, Kumaraguru College Of Technology, Coimbatore. And, this project work has not formed the basis for the award of any Degree/Diploma/Associateship/Fellowship or similar title to any candidate of any university.

Place: Coimbatore

Date: 16-04-2003

[illegible]

(Nihas A)

# ACKNOWLEDGEMENT

It is my privilege to thank some of the eminent persons who have rendered invaluable help in the successful completion of my project work.

First and foremost, I would like to express my deep sense of gratitude to ***Dr. S. Thangasamy, B.E (Hon's), Ph.D.,*** Head of Department of Computer Science and Engineering for his valuable guidance and for all the help extended during the course of the project work and its successful completion.

I fervently express my gratitude to ***Mr. R. Narayanan***, Vice President (Training & Education), Tata Consultancy Services (TCS), Thiruvananthapuram for providing an opportunity to carry out my project work at TCS.

I extend my sincere thanks to ***Mr. K. Lalita Prasad***, Training-in-Charge, TCS for permitting me to carry out the project work at TCS.

I would like to acknowledge my debt to ***Mr. Pramod Chandrasekhar***, Assistant Manager, HR, TCS who has been a constant source of inspiration throughout the project work. I am immensely grateful to him for guiding me in my personality development.

I especially acknowledge the valuable guidance and support give to me by ***Dr. Seenivasagam Neethi*** (External Guide), Associate Consultant, TCS.

I also thank ***Ms. Meera S.***, Assistant Consultant, TCS without whose encouragement and constructive criticism, the final version of the project could not have been perfected.

I sincerely appreciate the assistance and co-operation extended to me by ***Ms. Georgeen***, the ***Infrastructure Development and Management*** (IDM) staff, and the ***Admin Department*** of TCS.

I owe a great deal to my respected project guide ***Mr. A. Muthukumar, MCA,*** Lecturer Department of Computer Science and Engineering, for the motivation, constant encouragement and kind suggestions in every step throughout this project.

I thank the Library Staff of TCS, who backed me by providing the required reference books, a vital element in the successful completion of the project.

Last but not least, I thank all the faculty members and other staff of the Department of Computer Science, all the staff of TCS and my friends for their timely help, advice and suggestions which contribute either directly or indirectly to the success of the project.

# ABSTRACT

***Analyst Aid***, a software developed at TATA Consultancy Services, Technopark, Thiruvananthapuram, is a system which assists the requirement understanding phase in the software development life cycle. The system helps the user in determining the key factors in the consultancy problem and their relationships and thereby structuring the problem. As we know, the understanding of a system cannot be automated completely; it requires user intervention in finalizing decisions. The user plays a major part in the act.

The software has two subsystems working hand in hand – an analyzer and a graphical editor. The analyzer is a fully graphically supported software that helps in performing the analysis of the problem to be solved. Meanwhile, the graphical editor is equipped with tools that help in the structured representation of the perceptions, beliefs or values of the entities that define the problem. The user inputs a problem descriptive scenario to the analyzer through its text editor. The subsystem identifies the key elements or components in the document and does the SNAC analysis (where SNAC is Stakeholders, Needs, Alterables and constraints) of the problem, which is a requirement analysis method. The user of the system is free to change these components identified by the system thereby increasing the flexibility of the system. The analyst can then use the graphical editor to draw a Cybernetic Influence Diagram (CID) connecting the key factors in the problem, which structures the problem.

***Analyst Aid*** is a system developed in the Linux platform. The languages used for development are C++. The user interface has been realized using the Qt library for X11. It provides application developers with all the functionality needed to build state-of-the art graphical user interfaces. The language processing tools like Flex and GNU Bison have been used. Flex is a scanner generator tool and Bison is a parser generator.

# CONTENTS

# INTRODUCTION

# 1. INTRODUCTION

## 1.1. Project Profile

| | | |
|---|---|---|
| *Project Title* | : | Analyst Aid |
| *Official Name* | : | SE Tool Box |
| *Project Objective* | : | Develop a tool that assists an analyst in a consultancy problem |
| *Organization* | : | TATA Consultancy Services |
| *External Guides* | : | Dr. Seenivasagam Neethi, Associate Consultant, TATA Consultancy Services, Technopark, Thiruvananthapuram. |
| *Internal Guide* | : | Mr. MuthuKumar A, MCA Course Coordinator, Department of Computer Science, Kumaraguru College Of Technology, Coimbatore. |

## 1.2. About the Organization

TATA Consultancy Services (TCS) is a division of Tata Sons Ltd., the holding company of the $11.3 billion Tata Group, India's best-known business conglomerate. Established in 1968, its founding was based on the understanding that the management problems in Indian industry could be resolved through the effective use of information technology. Under the leadership of Dr. F.C. Kohli, TCS spearheaded the pioneering efforts in creating a globally recognizable brand for the Indian software industry.

Research and development play a critical role at TCS. The Systems Engineering and Cybernetics Center (SECC), TCS, initiated the dissemination of Systems Engineering and Cybernetics Techniques in TCS. TCS invests about 4 percent of its annual revenues in training, a shining example of which can be seen at the state-of-the-art training center in Thiruvananthapuram in the south India state of Kerala. The 'Induction Training Program' (ITP), which is for all the recruits from engineering colleges, is a specially designed, 77-day training course at the Thiruvananthapuram facility. The ITP is conducted with the objective of transforming engineers from diverse disciplines into software professionals.

## 1.3. About the System

The system ***Analyst Aid*** can be used by an analyst while understanding a problem for which a software solution is to be found. The system assists an analyst in understanding and structuring the problem. The problems that the analyst has to deal with can be any consultancy problem. To arrive at a perfect solution for any problem requires a better understanding of it. Here understanding does not mean mere understanding, but the right understanding, which is not an easy task. The software is named as ***Analyst Aid*** as its duty is to assist or aid in the analysis of a given problem.

Often when an analyst is presented with a consultancy problem, there can be some typical gaps in cognition, perception, or communication. It is typical that the problem which, the customer poses is not the real problem. This is termed as the cognitive gap which, is the difference between the real problem and the problem as understood by the customer. Moreover there can be gaps between perceptions of various stakeholders. The gaps in the consultancy problem can be further augmented by the communication gap between the customer and the analyst. This situation leads the analyst to study the problem in detail for a structured understanding which helps in bridging the three types of gaps.

Here, a Systems Thinking approach has been followed. This is a way of thinking that focuses on the relationships between parts forming a connected whole for a purpose. System Thinking forms the basis for clear thought and communication. Thinking systemically helps one to look at a system as interconnected parts functioning as a whole and thus enables the analyst to reach the right solution to the right problem.

There are several structured methods that facilitate systems thinking like Interpretive Structural Modeling (ISM), SNAC (Stakeholders, Needs, Alterables and Constraints) analysis, Cybernetic Influence Diagram (CID) analysis etc. Of these ISM is used for solving complex problems. We are confining our study on non-complex problems. So SNAC analysis and CID analysis are taken into consideration.

In SNAC analysis the first step is to create a problem descriptive scenario. A typical problem description should contain all the requirements needed for the system. The descriptive scenario is a document that furnishes information relevant to the organization. Information contained would relate to the following aspects: mission and strategies of the organization, internal status of the organization, environmental information to identify sources of constraints, opportunities and threats. The software is able to identify the key elements in the document i.e., the words and phrases that have significant relevance to the situation mentioned in the descriptive scenario. Apart from identifying the keywords in the document, the system also classifies the key elements into four categories, the Stakeholders, Needs, Alterables and Constraints (SNAC). The next step is to represent the relationship between each of these identified SNAC components. Representing the relationship is important because it finally helps the user in framing the objectives of the system. The user of the system is free to access the graphical editor at every stage to draw the Cybernetic Influence Diagram (CID) of the components that the system identified. The editor is enabled with different tools, which helps the user to draw the CID with the factors that he identified in the system or he can import the factors that the system identified automatically from the SNAC analysis phase. Analysis of the CID facilitates identification of the elements crucial for the problem situation, the extent of influence of the factors in the diagram etc. By this way he will be able to verify the results obtained by the analyzer with the CID. The main advantage of the CID editor is that it provides an efficient loop identification, loop analysis and merge-burst point analysis algorithm.

# BACKGROUND

# 2. BACKGROUND

## 2.1. The Procedure of SNAC Analysis

SNAC (Stakeholders, Needs, Alterables and Constraints) analysis is used to generate a comprehensive and exhaustive set of objectives for an organization.

### 2.1.1. The SNAC Process

1. The first step in SNAC analysis is in preparing a descriptive scenario of the organization. A descriptive scenario is a document that furnishes information relevant to the organization. The descriptive scenario is prepared after extensive interaction with the client.

2. The descriptive scenario should be screened for key elements. Key elements are words and phrases that have significant relevance to the situation mentioned in the descriptive scenario. In other words, each element represents a set of ideas, keeping in view the context within which these elements are used.

3. The next step in SNAC analysis is to classify the key elements into four categories: Stakeholders, Needs, Alterables and Constraints. These categories are defined below:

   a. ***Stakeholders*** are individuals, groups or agencies who have a stake in the organization, their needs have to be fulfilled by the organization, either by choice or by obligation. Stakeholders can be also be defined as those who will be affected by the policy of the organization.

   b. ***Needs***, refer to the requirements of stakeholders, which are to be fulfilled by the organization.

   c. ***Alterables***, are parameters, events or processes that can be controlled or altered to fulfill the needs of stakeholders.

d. ***Constraints*** are limitations imposed by factors that are not controllable by the organization. Qualification is generally required for classifying an element as a constraint.

4. In principle, the same key element can be classified as a Need, an Alterable or a Constraint

5. Objectives are developed based on the criteria that each objective satisfied Needs of the Stakeholders either by overcoming a constraint or by changing an Alterable.

6. Cross interaction matrix can be prepared to get a clear understanding of the interactions among various key elements. A Cross interaction matrix is one that represents interactions between Stakeholder and Needs and Alterables, Constraints and Alterable, Needs and Constraints. Matrix elements indicate the extent of interaction among Key elements. The output from SNAC analysis contains objectives both at corporate and department level. To arrive at specific department objectives, the objective set has to be mapped against the functional structure of the organization.

## 2.2. Cybernetic Influence Diagram (CID)

It is a method for problem structuring. It is a structured representation of the perceptions, beliefs or values of the various stakeholders that define the problem. This representation enables the client to reflect on or refine his/her ideas regarding the problem situation. It thus guides careful problem construction through broadening of issues until they can be explained.

CID aims at helping to client/consultant team to:

- ✓ Understand the perceptions of the various stakeholders regarding the situation.
- ✓ Gather the explanations of the stakeholders as to why the situation is as it is and why it matters to them.

- ✓ Gradual uncovering of the various facets of the situation under examination and establishment of linkages among of these.
- ✓ Identify the key issues within the problem situation and their relationship.
- ✓ Define the nature of the problem.
- ✓ Establish the nature of the system within which the issues are defined.

Analysis of the CID facilitates identification of a small number of elements crucial for the problem situation. It helps manage complexity without reducing it by identifying the network of interrelated problems that make up the issues as posed initially by the client.

### 2.2.1 Steps involved in the preparation of CID

The following steps are involved in the preparation of a CID. The process is not sequential but is iterative in nature, depending on the ability to draw out the perceptions of the stakeholders, and understanding of the complexity of the situation under study.

#### i. Knowledge Acquisition

Based on the preliminary understanding of the situation under study all the relevant empirical and theoretical information is collected. Initially the richest possible picture of the situation under consideration should be arrived at by collecting as many perceptions and aspects of the problem as possible from a wide range of sources and people with a stake in the problem situation. It is important to refrain from imposing any analysis at this stage.

#### ii. Sources of data

The forms and sources of data may be diverse and are collected both from primary and secondary sources. Literature survey, discussions and interviews with different sets of stakeholders of the system will provide a better understanding of the situation. Nominal group technique can be used to elicit optimum information in a short span of tome and promote a systematic brainstorming by stimulating creative thinking.

### iii. Descriptive scenario

The consultants' understanding of the situation is compiled in the form of a descriptive scenario of the problem situation. This statement should include all that internal and external factors that impinge on the situation and their interrelationships.

### iv. Identification of factors or elements

From the descriptive scenario, the factors or key elements that constitute the situation are identified keep in view the purpose of the analysis. They originate from the perceptions and mental images of the stakeholders involved. Key elements are words and phrases that have significant relevance to the situation as described in the descriptive scenario. In other words, each element represents a set of ideas that are relevant for the problem situation. They key elements can be reviewed for clubbing of two or more elements or elaboration as required.

### v. Identification of relationships among factors

This crucial step involves establishing the perceived direct relationship between the factors based upon the understanding of the problem situation while refraining from the value judgments. A matrix may be developed to plot the relationships among key elements. CID can be visually depicted by having key elements as nodes and relationships among the key elements as directed links. Each link represented by arrow from one element to another represents a proposition. The direction of the link depicts a relationship like 'causes', 'influences', 'leads to', 'inheres', 'decreases' etc. The translation of all propositional relationships into arrow-links produces a set of interconnected and interacting feedback cycles.

### vi. Ratify the inter-linkages

The primary use of the CID till this stage is not as a problem-solving tool but rather as a reflective device. The client is invited at this stage to correct the impressions gained by the consultant. In so doing, the client may also reflect on the influences between a few elements that he/she perceives to be of interest. The resultant diagram needs to be ratified with the various stakeholders/experts/clients with respect to the factors and the relational links.

### vii. Modify the CID

Based on the discussions with the client and various stakeholders, the relationships represented in the CID have to be modified. This needs to be done till the consultant is confident of having arrived at a representation that is reasonably acceptable to all stakeholders. Drawing the CID is an iterative process and it undergoes various changes before taking its final form.

### 2.2.2 Analysis of the CID

CID enables a consultant to manage complexity without reducing it. It has a potential to unearth the network of underlying problems and also suggest what can and cannot be done about the problem. Hence, it is necessary to understand how this knowledge can be utilized under various problem situations. Interpretation of the CID has to be undertaken in relation to the purpose of analysis. The following may be useful in the context of planning for an organization.

### i. Merge and Burst Points

The presence of merge points and burst points indicates the criticality and importance of certain key factors. Merge points are those nodes where convergence of a lot of arrows takes place while burst points are the nodes from which a lot of arrows diverge. These elements are potent because they have ramifications for a large number of other elements. The burst points provide maximum leverage to the organization in implementing change, if it is within the control of the organization. In the case of merge points, a number of other factors need to be acted on before achieving desired change with respect to this element.

### ii. Domain Analysis

Calculate the total number of in-arrows and out-arrows from eacŀ node that is its immediate domain. Those nodes whose immediate domain is the most complex are central to the problem situation and may need to be explored in greater detail.

### iii. Feedback loops

The detection of feedback loops is central to the investigation. They imply the possible existence of growth, decline or feedback control and hence the organization's ability to react or exploit changes in the environment. The critical elements that need to be monitored by the organization can be identified through analysis of feedback loops. Also, if the management wants to stimulate changes they can introduce strategic relationships and create significant loops. Identification of the feedback loops also facilitates review of the cause and effect as perceived by the stakeholder. If true feed back loops are identified, then there is a need to establish the nature of feedback.

#### a. Negative Feedback Loops

Feedback loops are negative when the loop contains an odd number of negative links. They depict self-control and are oriented towards reaching or maintaining a stable equilibrium state. That is any perturbation in the state of variables will result in stabilizing dynamics to bring activity under control.

#### b. Positive Feedback Loops

An even number of negative links or all positive links suggest a positive feedback cycle that is oriented towards producing cumulative change in a given state leading to exponential growth or decline. The change generating process may be directed towards purposive growth or decline.

# DEVELOPMENT ENVIRONMENT

# 3. DEVELOPMENT ENVIRONMENT

## 3.1 Hardware and Software platforms

| | | |
|---|---|---|
| **Hardware** | : | PC with X11 Linux Operating System. |
| **Software** | : | C++, Qt library |
| **Software Tools** | : | Brill Tagger, flex, GNU bison |

### 3.1.1 About Linux

Linux is a Unix clone written from scratch by Linus Torvalds with assistance from a loosely-knit team of hackers across the Net. It has all the features as expected in a modern fully-fledged Unix, including true multitasking, virtual memory, shared libraries, demand loading, shared copy-on-write executables, proper memory management and TCP/IP networking. Linux is a free (GPL Licensed), from scratch operating system based heavily on the POSIX and UNIX API's. It supports both 32 and 64 bit hardware and provides a stable multi-user internet ready operating system. It works on IBM PC compatibles and 386 or higher processor.

Linux itself is not Unix, although many people call it that and is very hard pushed to tell the difference. This is because the Unix trademark is specific to systems that meet a complex set of X/Open standards and has a cost. Some of the many applications for Linux are X11 Desktop, File server, Computing Backend, Web Server, Usenet News, Terminal Server, FTP Archive, and Firewall. Linux uses Internet and industry standard components and protocols giving a system with complete network integration. The operating system can act as a serve for most major file serving protocols, and provide all the major Internet applications. The X window system provides a networked and platform independent graphical interface that (unlike proprietary user interfaces) allows one desktop to access applications running on multiple machines across local and wide area networks. Linux is normally obtained as a "distribution". This is a combination of the Linux operating system kernel and other tools, utilities and applications. Some of these are available for free over the Internet and others on CD-ROM. Because Linux itself is free software that can be freely copied, many distributions are available both over the Internet and sold on CD-ROM with added convenience and support.

### 3.1.2 About X Windows

The X Window System, more simply 'X' or 'X11', is judged worldwide to be one of the most successful open source, collaborative technologies developed to date. It is the de facto standard graphical engine for the UNIX and Linux operating systems and provides the only common windowing environment bridging the heterogeneous platforms in today's enterprise computing. The inherent independence of the X Window System from operating system and hardware, its network-transparency, and its support for a wide range of popular desktops are responsible for its continuing and growing popularity. All major hardware vendors support the X Window System.

The X Protocol was developed in the mind 1980's amid the need to provide a network transparent graphical user interface primarily for the UNIX operating system. X provides for the display and management of graphical information, much in the same manner as Microsoft's Windows and IBM's Presentation Manager. They key difference is in the structure of the X Protocol. Whereas Windows and Presentation Manager simply display graphical applications local to the PC, the X Protocol distributes the processing of applications by specifying a client-server relationship at the application level. The what to do part of the application is called an X client and is separated from the how to do part, the display, called the X server. X clients typically run on a remote machine which has excess computing power and displays on an X server. The benefit is true client-server and distributed processing.

The X Protocol defines a client-server relationship between an application and its display. To meet this the application (called an X client) is divorced from the display (known as the X server). X further provides a common windowing system by specifying both a device dependent and an independent layer, and basing the protocol on an asynchronous network protocol for communication between an X client and X server. In effect, the X Protocol hides the peculiarities of the operating system and the underlying hardware. This masking of architectural and engineering differences simplifies X client development and provides the springboard for the X Window System's high portability.

The software ***Analyst Aid*** is developed in Linux environment and requires a good user interface. X11 provides a strong basement for such GUI applications.

### 3.1.3 About C++

C++ was initially designed and implemented by Dr. Bjarne Stroustrup at AT & T Labs (then AT & T Bell Labs). The first commercial release happened in 1985. The language gained widespread use in industry and academia during the 1980s, and around 1990 the major computer and software tools suppliers started to provide C++ to their users as a major implementation tool.

C++ is a general purpose programming language with a bias towards systems programming that

- is a better C
- supports data abstraction
- supports object-oriented programming
- supports generic programming

C++ supports low-level programming in traditional styles, data abstraction, object-oriented programming, and generic programming. C++ was initially developed from the C programming language by the addition of facilities for object-oriented programming from the SIMULA programming language. C++ is distinguished among progamming languages by a combination of efficiency (like C and Fortran) and abstraction facilities. The "abstraction facilities" provided by C++ allow programs to be expressed in terms natural to application designers (rather than lower-level computer-oriented terminology). The C++ abstraction mechanisms are distinguished by their run-time efficiency.

The designers of C++ wanted to add object-oriented mechanisms without compromising the efficiency and simplicity that made C so popular. One of the driving principles for the language designers was to hide complexity from the programmer, allowing him to concentrate on the problem at hand.

Because C++ retains C as a subset, it gains many of the attractive features of the C language, such as efficiency, closeness to the machine, and a variety of built-in types. A number of new features were added to C++ to make the language even more robust, many of which are not used by novice programmers. C++ is rich with a set of libraries. As our system has to do a large amount of string manipulations, the string library was used most. Some of the functions in the string library and there syntax are listed below.

**a. strlen**

**Function** : Calculates the length of a string.

**Syntax** : size_t strlen (const char * s)

**Return value** : The function strlen returns the number of characters in 5, not counting the NULL-terminating character.

**Description** : The function strlen calculates the length of s.

**b. strcat**

**Function** : Appends one string to another.

**Syntax** : char *strcat (char *dest, const char *src)

**Return value** : The function returns a pointer to the resultant concatenated string.

**Description** : The function strcat appends a copy of the source to the end of destination. The length of resulting string is strlen (dest) + strlen (src).

**c. strcmp**

**Function** : Compares one string with another.

**Syntax** : int strcmp (const char *s1, const char *s2)

**Return value** : The function returns a value that is

<0, if s1 is less than s2

==0, if s1 is same as s2

>0, if s1 is greater than s2

**Description** : The function strcmp performs comparison between s1 and s2, beginning with the first char in each string and continuing for subsequent characters until the corresponding characters differ or until the end of one of the strings is reached. The characters are treated as unsigned character type.

**d. strcpy**

**Function** : Copies one string to another.

**Syntax** : char *strcpy (char *dest, const char *src)

**Return value** : The function returns dest.

**Description** : The function strcpy copies the string src to dest, stopping after the terminating NULL character.

**e. strcspn**

**Function** : Scans a string for the initial segment not containing any subset of a given set of characters.

**Syntax** : size_t strcspn (cost char *s1, const char *s2)

**Return value** : The function returns the length of the initial segment of the string s1 that consists entirely of characters not present in the string s2.

**Description** : The function strcspn scans a string for the initial segment not containing any subset of a given set of characters.

**f. strstr**

**Function** : Scans a string for the occurrence of a given substring.

**Syntax** : char *strstr (const char *s1, const char *s2)

**Return value** : The function returns a pointer to the element in s1, where s2 begins (points to s2 in s1). If s2 does not occur in s1, strstr returns NULL.

**Description** : The function strstr scans s1 for the first occurrence of the substring s2.

**g. strtok**

**Function** : Searches one string for tokens that are separated by delimiters defined in a second string.

**Syntax** : char strtok (char s1, const char *s2)

**Return value** : The function returns a pointer to the token found in s1. The NULL pointer is returned when there are no more tokens.

**Description** : The function considers the string s1 to consist of a sequence of zeros or more text tokens, separated by combinations of one or more characters from the separator string s2. The first call to strtok returns a pointer to the first character of the first token in s1 and writes the NULL character into s1 at the pint immediately following the returned token. Subsequent calls with NULL as first argument will work through string s1 in this way, until no token remains. The separator string s2, can vary from call to call.

**h. strncpy**

**Function** : Copies a given number of bytes from one stream to another, truncating or padding as necessary.

**Syntax** : char* strncpy (char* dest, const char* src, size_t maxlen)

**Return value** : The function strncpy returns dest

**Description** : The function strncpy copies a maximum of maxlen characters from src to dest, truncating or NULL-padding the string dest. The target string dest, might not be NULL-terminated if the length of src is maxlen or more.

### 3.1.4 GNU debugger (GDB)

GDB, the GNU Project debugger, allows to see what is going on 'inside' another program while it executes or what another program was doing at the moment it crashed.

GDB can do four main kinds of things (plus other things in support of these) to help the user catch bugs in the act:

- ✓ Start the program, specifying anything that might affect its behavior.
- ✓ Make the program stop on specified conditions.
- ✓ Examine what has happened, when the program stopped.
- ✓ Change things in the program, so that the user can experiment with correcting the effects of one bug and go on to learn about another.

The program being debugged can be written in C, C++, Pascal (and many other languages). Those programs might be executing on the same machine as GDB (native) or on another machine (remote). GDB can run-on most popular UNIX and Microsoft Windows variants.

### 3.1.5 About Qt

Qt is a cross-platform C++ GUI application framework. It provides application developers with all he functionality needed to build state-of-the-art graphical user interfaces. Qt is fully object-oriented, easily extensible, and allows true component programming.

Since its commercial introduction in early 1996, Qt has formed the basis of any thousands of successful applications worldwide. Qt is also the basis of the popular KDE Linux desktop environment, a standard component of all major Linux distributions.

Qt is supported on the following platforms:

- **MS/Windows** – 95, 98, NT, and 2000
- **Unix/X11** – Linux, Sun Solaris, HP-UX, Digital Unix, IBM AIX, SGI IRIX and a wide range of others.
- **Embedded** – Linux platforms with frame buffer support.

Qt/X11 does not require any additional graphical layer above X11, neither Xt nor Motif nor win32 emulation libraries. It is highly optimized native code that runs directly on top of the lowest graphical layer Xlib.

Qt includes a rich set of *Widgets* ('controls' in Windows terminology) that provide standard GUI functionality. Qt introduces an innovative alternative for inter-object communication, called *Signals* and Slots that replaces the old and unsafe callback technique. Qt also provides a conventional *Events* model for handling mouse clicks, key presses, etc. Qt's cross-platform GUI applications can use all the user interface functionality required by modern applications, such as menus, context menus, dockable toolbars, balloon help, drag and drop, etc. Intuitive naming conventions and a conventions and a consistent programming approach simplify coding. Qt also includes **Qt Designer**, a tool for designing user interfaces graphically. **Qt Designer** supports Qt's powerful *Layouts* in addition to absolute positioning. **Qt Designer** can be used purely as a design tool, or it can be used to create entire applications with the built-in C++ code editor.

Qt has excellent support for 2D and 3D graphics. Qt is the de-facto standard GUI toolkit for platform-independent Open GL programming Qt makes it possible to create platform-independent database applications using standard databases. Qt includes native drivers for Oracle, Microsoft SQL Server, Sybase Adaptive Server, PostgreSQL, MySQL and ODBC-compliant databases. Qt programs have native look and feel on all supported platforms using Qt's *Styles and Themes* support. From a single source tree, recompilation is all that is required to produce applications for Windows 95/98/NT4/ME/2000, Mac OS X, Linux, Solaris, HP-UX and many other versions of Unix with X11. Qt applications can also be compiled to run on Qt/Embedded. Qt's `qmake` build tool produces Makefiles or `.dsp` files appropriate to the target platform. Since Qt's architecture takes advantage of the underlying platform, many customers also use Qt for single-platform development on both Windows and on Unix because they prefer the Qt approach.

Qt uses Unicode throughout and has considerable support for internationalization. Qt includes a variety of domain-specific classes. For example, Qt has an XML module that includes SAX and DOM parsers. Objects can be stored in memory using Qt's STL-compatible collection classes. Local and remote file handling using standard protocols are provided by Qt's input/output networking classes. Qt applications can have their functionality extended by plugins and dynamic libraries. Plugins provide additional codecs, database drivers, image formats, styles and widgets. Libraries can offer an unlimited range of functionality.

Because of all these functionality that Qt provides, it was found to be the best for this application development.

### 3.1.6 Tools Used

#### 3.1.6.1. Brill Tagger – A parts of speech tagger

Brill Tagger is used to tag the parts of speech of each word appearing in the problem descriptive scenario that is given as input to the system ***Analyst Aid***. Brill Tagger is a rule-based tagger. In the training phase, this tagger makes an initial hypothesis about the correct tags. In an iterative fashion it then betters its performance with regard to the training corpus by postulating context dependent tag rewrite rules. Brill's tagger is written in C. The accuracy of the part-of-speech tagger should be around 95-97% (i.e. 95-97% of the word tokens in arbitrary English text receive the correct tag).

Brill Tagger is a parts of speech tagger, which tags each word in the input text with the appropriate part of speech. Tagging is done in two stages. Every word is assigned its most likely tag in isolation. Each word in the tagged training corpus has a lexical entry consisting of a partially ordered list of tags, indicating the most likely tag for that word, as well as all other tags seen with that word (in no particular order). A list of transformations is provided for determining the most likely tag for words not in the lexicon. Unknown words are first assumed to be nouns (proper nouns if capitalized), and then cues based upon prefixes, suffixes, infixes, and adjacent word co-occurrence are used to change the guess of most likely tag. Next, contextual transformations are used to improve accuracy.

To compile the programs, type (in the tagger base directory):

**make**

To execute the program, type:

**tagger** LEXICON YOUR-CORPUS BIGRAMS LEXICALRULEFULE CONTEXTUALRULEFILE where YOUR-CORPUS is the file name of the (currently untagged) corpus you wish to have tagged, and the other files are all provided with the tagger. Options (which are typed after all the file names) are:

| | |
|---|---|
| **-h** | :: help |
| **-w wordlist** | :: provide an extra set of words beyond those in LEXICON. |
| **-i filename** | :: writes intermediate results from start state tagger. into filename |
| **-s number** | :: processes the corpus to be tagged "number" lines at a time. This should be specified if memory problems result from trying to process to large a corpus at once. |
| **-S** | :: use start state tagger only. |
| **-F** | :: use final state tagger only. In this case, YOUR-CORPUS is a tagged corpus, whose tagging will be changed according to the final-state-tagger contextual rules. YOUR-CORPUS should be tagged corpus ONLY when using this option. |

The tagger writes to standard output. The output can be also be redirected to some file.

**Information on training files**

**LEXICON**

A list of (word tag 1 tag2 ...tagn), where tag 1 is the most likely tag for "word" in the training corpus, and tag2...tagn are other taggings of "word" seen in the training corpus (in no particular order). There are three different lexica provided with Brill Tagger.

LEXICON. BROWN. AND.WSJ was derived from the Penn Treebank tagging of the WSJ, roughly 3 million words, and the Brown Corpus. LEXICON.BROWN was derived from the Penn Treebank tagging of the Brown corpus only. LEXICON.WSJ was derived from only the WSJ. (LEXICON is a link to LEXICON.BROWN.AND.WSJ) Which lexicon you choose to use will depend on the type of corpus you wish to tag.

**CORPUS: the corpus you wish to tag**

Should be one sentence per line, with punctuation (and anything else appropriate) tokenized. Words can be pretagged by tagging with two slashes: H's the winner (at least, that's what I was told). The boy//NN said: “I am here”.

**BIGRAMS: a list of adjacent word pairs seen in the training corpus**

Used to apply transformations such as: “change the tag from X to Y if word Z ever appears to the right”. In this distribution, this file is just set to a dummy list, as a place holder. This is because “BIGRAMS” is only used for unknown words, and there is no Brown or WSJ text in the Penn Treebank that is not tagged.

**LEXICALRULEFILE: list of transformations used for initial tagging of words not in the lexicon**

There are two lexical rule files provided with the Brill Tagger. LEXICALRULEFILE.BROWN was derived from roughly 300,000 words of tagged text from the Brown Corpus. LEXICALRULEFILE.WSJ was derived from roughly 300,000 words of tagged text from the WSJ. (LEXICALRULEFILE is a link to LEXICALRULEFILE.WSJ)

**CONTEXTUALRULEFILE: list of contextually triggered transformations**

CONTEXTUALRULEFILE.BROWN was derived from roughly 600,000 words of tagged text from the Brown Corpus. CONTEXTUALRULEFILE.WSJ was derived from roughly 600,000 words of tagged text from the WSJ CONTEXTUALRULEFILE.WSJ.NOLEX was derived from roughly 600,000 words of tagged text from the WSJ, disallowing all transformations that make reference to words. (CONTEXTUALRULEFILE is a link to CONTEXTUALRULEFILE.WSJ)

## How the Part-of-Speech Works

In the first step, a lexical lookup module assigns exactly one tag to each occurrence of a word (usually the most frequent tag for that word form), disregarding context.

The lexicon has been generated from tagged corpora and contains more than 93,000 entries.

In the second step, words not in the lexicon are handled separately, by the guesser. The guesser starts by assigning tag NNP to unknown capitalised words, NN to others. This is the relevant rule:

```
replace (pos 'unknown' 'NNP' 'NN') # [isCap#[0]]
```

Then replacement rules are applied that may change these tags on the basis of a simple suffix analysis. Here is a guessing rule:

```
replace (pos 'NN' 'JJ') # [suffix#less#[0]]
```

The rule means "replace tag NN with JJ if the word in question ends in "less".

```
replace (pos 'VB' 'NN') # [canHave# 'NN' #[0] pos#
'DT' #[~1]]
```

The rule means "replace tag VB with NN if the word in question can have tag NN (according to the lexicon) and if the pervious word is tagged DT".

The present system uses around 50 guessing rules and nearly 300 context rules. Both kinds of rules have been induced from tagged corpora by means of Transformation Based Learning (TBL). At the directory where the tagger is installed the list of files that are generated are given below.

```
[root@AA/Bin_and_Data]#ls
BIGRAMS                              LEXICALRULEFILE.BROWN
CONTEXTUALRULEFILE                   LEXICALRULEFILE.WSJ
CONTEXTUALRULEFILE.BROWN             LEXICON
CONTEXTUALRULEFILE.WSJ               LEXICON.BROWN
CONTEXTUALRULEFILE.WSJ.NOLEX         LEXICON.BROWN.AND.WSJ
contextual-rule-learn                LEXICON.WSJ.Z
final-state-tagger                   NBEST-RULES
fix-kbest-rule-learn                 nbest-tagger
kbest-contextual-rule-learn          start-state-tagger
LEXICALRULEFILE                      tagger
```

When an input file named untagged-input-file is tagged using the following command the tagger gives the following output.

```
[root@AA/Bin_and_Data]#tagger LEXICON untagged-input-file BIGRAMS
LEXICALRULEFILE CONTEXTUALRULEFILE | MORE
START STATE TAGGER::LEXICONREAD
START STATE TAGGER::CORPUSREAD
START STATE TAGGER::RULEFILEREAD
START STATE TAGGER::BIGRAMS READ
ssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
ssssssssssssssssssssssssssssssssssssssssssssssssssssss

FINAL STATE TAGGER :: READ IN OUTPUT FROM START
FINAL STATE TAGGER :: READ IN LEXICON
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fffffffffffffffffffffffffffffffffffffffffffffffffffffff

The/DT September/NNP October/NNP term/NN jury/NN had/VBD been/VBN
charged/VBN by/IN Fulton/NNP Superior_Court_Judge_Durwood_Pye/NNP
to/TO investigate/VB reports/NNS of/IN possible/JJ"/" irregularities/NNS "/"
in/IN the/DT hard-fought/JJ primary/JJ which/WDT was/VBD won/VBN
by/IN Mayor-nominate_Ivan_Allen_Jr./NNP./. "/" Only/RB a/DT relative/JJ
handful/NN of/In such/JJ reports/NNS was/VBD received/VBN "/",/, the/DT
jury/NN said/VBD,/,"/" considering/VBG the/DT widespread/JJ interest/NN
in/IN the/DT election/NN,/, the/DT number/NN of/IN voters/NNS and/CC
the/DT size/NN of/IN this/DT city/NN '"/"./.
[root@AA/Bin_and_Data]#
```

The accuracy of the parts-of-speech tagger should be around 95-97% (i.e., 95-97% of the word tokens in arbitrary English text receive the correct tag).

The tag set used in Brill Tagger.

| POS Tag | Description | Example |
|---|---|---|
| CC | coordinating conjunction | and |
| CD | cardinal number | 1, third |
| DT | Determiner | the |
| EX | existential there | there is |
| FW | foreign word | d'hoevre |
| IN | preposition/subordinating conjunction | in, of, like |
| JJ | Adjective | green |
| JJR | adjective, comparative | greener |
| JJS | adjective, superlative | greenest |
| MD | Modal | could, will |
| NN | noun, singular or mass | table |
| NNS | noun plural | tables |
| NNP | proper noun, singular | john |
| NNPS | proper noun, plural | vikings |
| PRP | personal pronoun | i, he, it |
| PRPO | possessive pronoun | my, his |
| RB | Adverb | however |
| RBR | adverb, comparative | better |
| RBS | adverb, superlative | best |
| RP | Particle | give up |
| TO | To | to go, to him |
| UH | Interjection | uhhuhhuhh |
| VB | verb, base form | take |
| VBD | verb, past tense | took |
| VBG | verb, gerund/present participle | taking |
| VBN | verb, past participle | taken |
| VBP | verb, sing. present, non-3d | take |
| VBZ | verb, 3rd person sing. present | takes |
| WDT | wh-determiner | which |
| WP | wh-pronoun | who, what |
| WPO | possessive wh-pronoun | whose |
| WRB | wh-abverb | where, when |

**Table 1**

### 3.1.6.2. Flex – Fast lexical analyzer generator

`flex` is a tool for generating **scanners** – programs which recognized lexical patterns in text. `flex` reads the given inputs files, or its standard input if no file names are given, for a description of a scanner to generate. The description is in the form of pairs of regular expressions and C code, called **rules**. `flex` generates as output a C source file, `'lex.yy.c'`, which defines a routine `'yylex()'`. This file is complied and linked with the `'-lfl'` library to produce an executable file. When the executable is run, it analyses its input for occurrences of the regular expressions. Whenever it finds one, it executes the corresponding C code.

**Syntax**

flex [-bcdfhilnpstvwBFILTV78+? –C[aefFmr] -ooutput-

Pprefix –Sskeleton]

[--help –version]  [filename….]

**Format of the input file**

The `flex` input file consists of three sections, separated by a line with just `'%%'` in it:

```
definitions
%%
rules
%%
user code
```

The **definitions** section contains declarations of simple **name** definitions to simplify the scanner specification, and declarations of **start conditions**. Name definitions have the form:

```
name definition
```

The "name" is a word beginning with a letter or an underscore ('_') followed by zero or more letters, digits, '_', or '-' (dash). The definition is taken to begin at the first non-white-space character following the name and continuing to the end of the line. The definition can subsequently be referred to using "{name}", which will expand to "(definition)".

The *rules* section of the `flex` input contains a series of rules of the form:

```
pattern action
```

where the pattern must be unindented and the action must begin on the same line.

Finally, the user code section is simply coded to `'lex.yy.c'` verbatim. It is used for companion routines, which call or are called by the scanner. The presence of this section is optional; if it is missing, the second `'%%'` in the input file may be skipped, too.

In the definitions and rules sections, any indented text or text enclosed in `'%{'`and `'%}'` is copied verbatim to the output (with the `'%{}'` 's removed). The `'%{}'` 's must appear unindented on lines by themselves.

In the rules section, any indented or %{} text appearing before the first rule may be used to declare variables which are local to the scanning routine and (after the declarations) code which is to be executed whenever the scanning routine is entered. Other indented or %{} text in the rule section is still copied to the output, but its meaning is not well-defined and it may well cause compile-time errors.

In the definitions section (but not in the rules section), an unindented comment (i.e., a line beginning with "/*") is also copied verbatim to the output up to the next "*/".

**How the input is matched**

When the generated scanner is run, it analyses its input looking for strings which match any of its patterns. If it finds more than one match, it takes the one matching the most text (for trailing context rules, this includes the length of the trailing part, even though it will then be returned to the input). If it finds two or more matches of the same length, the rule listed first in the `flex` input file is chosen.

Once the match is determined, the text corresponding to the match (called the *token*) is made available in the global character pointer `yytext`, and its length in the global integer `yyleng`. The *action* corresponding to the matched pattern is then executed (a more detailed description of actions follows), and then the remaining input is scanned for another match.

If no match is found, then the **default rule** is executed: the next character in the input is considered matched and copied to the standard output. Thus, the simplest legal `flex` input is: %% which generates a scanner that simply copies its input (one character at a time) to its output.

**Generated scanner**

The output of `flex` is the file `'lex.yy.c'`, which contains the scanning routine `'yylex()'`, a number of tables used by it for matching tokens, and a number of auxiliary routines and macros. By default, `'yylex()'` is declared as follows:

```
Int yylex()
     {
     ...various definitions and the actions in here...
```

Whenever `'yylex()'` is called, it scans tokens from the global input file `yyin` (which defaults to stdin). It continues until it either reaches an end-of-file (at which point it returns the value 0) or one of its actions executes a return statement.

**Interfacing with Bison**

One of the main uses of `flex` is as a companion to the `bison` parser generator. Bison parsers expect to call a routine named '`yylex()`' to find the next input token. The routine is supposed to return the type of the next token as well as putting any associated value in the global `yylval`. To use `flex` with `bison`, one specifies the '`-d`' option to `bison` to instruct it to generate the file '`y.tab.h`' containing definitions of all the '`%tokens`' appearing in the `bison` input. This file is then included in the `flex` scanner.

### 3.1.6.3. Bison – A parser generator

Bison is a general-purpose parser generator that converts a grammar description for an LALR (1) context-free grammar into a C program to parse that grammar. Bison is upward compatible with Yacc: all properly-written Yacc grammars ought to work with Bison with no change. Anyone familiar with Yacc should be able to use Bison with little trouble. One needs to be fluent in C programming in order to use Bison.

Bison was written primarily by Robert Corbett; Richard Stallman made it Yacc-compatible. Wilfred Hansen of Carnegie Mellon University added multicharacter string literals and other features.

In order for Bison to parse a language, it must be described by a *context-free grammar*. This means that specifying one or more *syntactic groupings* gives rules for constructing them from their parts. For example, in the C language, one kind of grouping is called an 'expression'. One rule for making an expression might be, "An expression can be made of a minus sign and another expression". Another would be, "An expression can be an integer". Rules are often recursive, but there must be at least one rule, which leads out of the recursion.

The most common formal system for presenting such rules for humans to read is *Backus-Naur Form* or "BNF", which was developed in order to specify the language Algol 60. Any grammar expressed in BNF is a context-free grammar. The input to Bison is essentially machine-readable BNF.

Not all context-free languages can be handled by Bison, only those that are LALR (1). In brief, this means that it must be possible to tell how to parse any portion of an input string with just a single token of look-ahead. Strictly speaking, that is a description of an LR (1) grammar, and LALR (1) involves additional restrictions that are hard to explain simply; but it is rare in actual practice to find an LR (1) grammar that fails to be LALR (1).

**Language and Context free grammar**

In the formal grammatical rules for a language, each kind of syntactic unit or grouping is named by a *symbol*. Those, which are built by grouping smaller constructs according to grammatical rules, are called *nonterminal symbols*; those which can't be subdivided are called *terminal symbols* or *token types*. We call a piece of input corresponding to a single terminal symbol a *token*, and a piece corresponding to a single nonterminal symbol a *grouping*. One nonterminal symbol must be distinguished as the special one, which defines a complete utterance in the language. It is called the *start symbol*. In a compiler, this means a complete input program.

The Bison parser reads a sequence of tokens as its input, and groups the tokens using the grammar rules. If the input is valid, the end result is that the entire token sequence reduces to a single grouping whose symbol is the grammar's start symbol. If we use a grammar for C, the entire input must be a 'sequence of definitions and declarations'. If not, the parser reports a syntax error.

**Overall Layout of a Bison Grammar**

The input file for the Bison utility is a **Bison grammar file**. The general form of a Bison grammar file is as follows:

```
%{
C declarations
%}
```

```
%%
Grammar rules
%%
Additional C code
```

The `'%%'`, `'%{'` and `'%}'` are punctuation that appears in every Bison grammar file to separate the sections.

The C declarations may define types and variables used in the actions. Pre-processor commands can also be used to define the macros used there, and use "`#include`", to include header files that do any of these things.

The Bison declarations declare the names of the terminal and non-terminal symbols, and may also describe operator precedence and the data types of semantic values of various symbols. The grammar rules define how to construct each non-terminal symbol from its parts. The additional C code can contain any C code to be used. Often the definition of the lexical analyzer `yylex` goes here, plus subroutines called by the actions in the grammar rules. In a simple program, all the rest of the program can go here.

# REQUIREMENT ENGINEERING

# 4. REQUIREMENTS ENGINEERING

## 4.1 Requirements Analysis

### 4.1.1 Problem Recognition

TATA Consultancy Services (TCS) had developed a new vision called ***Concept to Code***. This says that you are given a problem or a concept, there will be a tool which will convert this concept to its equivalent hard code through a series of steps. The project entitled ***Analyst Aid*** is done as a first step towards ***Concept to Code***. The first step is to structure and understand the problem. The results of this will be given to UML diagram generator, whose output could be converted to code using the already available tool called Master Craft.

The problem is find an answer to the question, "How can you understand a problem well?"

Understanding a problem is the primary step involved in solving it. Particularly in finding software solutions to problems, which involve a lot of effort and resources, the lost will be enormous if mistakes are made at this initial stage. Wrong understanding can occur because of many reasons. Gap in communication is one such reason, the user will be describing an aspect and the software vendor/consultant may understand it in another way and he may provide a controversial solution for the problem. There can be cases where even though the user specifies his requirements hard on paper, the software vendor/consultant is not be able to extract all the required essentialities from that document and provide the user with the solution. Here, the analyst fails because of the defective way of understanding the problem. Unless the analyst is able to correctly structure the problem and understand the right problem in the right way he will definitely fail to realize the apt solution for it. So this work is an attempt to develop a tool that will assist the analyst throughout the understanding procedure.

Understanding the problem means going through the problem, identifying the factors that are important in the system and factors that are not. Also we should be able to connect these factors together in some way which will help us in reaching the objective of the system.

Two important problem structuring methods are followed here; the SNAC (Stakeholders, Needs, Alterables and Constraints) analysis and the Cybernetic Influence Diagram (CID) analysis.

Here SNAC analysis is used because we are concentrating in developing a software which assist in analyzing typical application oriented problems. Application oriented problems are those on which we can perform a view point oriented analysis i.e. in those systems we can get different view points if we view the system on behalf of the different stakeholders appearing in the system. These stakeholders with varying needs will be connected with various activities and in doing these activities there could be numerous constraints. Thus for analyzing an application oriented problem SNAC is best.

In SNAC analysis, the key components found in typical problem, i.e. the stakeholders, needs, alterables and constraints are extracted. This is a natural language processing problem. From the lexicographical clues obtained from the document the analyst should be able to trace the probable stakeholders evolved in the system, their needs, and the constrains and alterables that come into existence while solving their needs. In CID analysis, a diagram connecting the key elements of the problem is drawn which helps the analyst to structure the problem. Now the diagram is analyzed for loops, merge points and burst points. Finding the loops in the diagram and studying their behavior is a method of structuring the problem. The resultant polarities of these loops are also found.

***Analyst Aid*** was developed as part of an attempt to develop a tool that will assist an analyst to understand any problem, typically a consultancy problem.

The system initially focuses on attaining the following objectives

- ✓ Find the key elements in the document and perform SNAC analysis of the document.

- ✓ Enable the user to draw the CID of a problem using a graphical editor and to the merge point, burst point and loop analysis of the diagram.

In the manual process of understanding a problem, the analyst goes through a problem descriptive scenario, which is a document that covers all the requirements of the user in solving the problem. The analyst then notes the important key elements in that document by underlining that portion of text. From these key elements identified, the analyst tries to draw a relationship between them. This approach will help the analyst to structure the problem and get a clear idea about the problem. The same idea is followed in Analyst Aid also.

The system gets a problem descriptive scenario as its input. With the lexicographical clues, it identifies a set of key elements in them. The phrases or patterns that usually fall under the key element category are verb and noun phrases in the document. So we have to find the noun and verb phrases in the document to get the key elements. Finding the SNAC components also requires a lexicographical analysis. A stakeholder will always be a noun, this is the greatest clue in finding a stakeholder. A need usually occur in some regular pattern say ***to + implement + actions***, so these patterns could be found from the document and marked as needs. The constraints and alterables eventhough could be found with the help of some lexicographical clues, these will not be identified by the system. This is because this will require a component of artificial intelligence in the system, which is avoided inorder to reduce the complexity of the system. So the user is left to find the constraints and alterables in the document.

For performing the CID analysis of a problem the analyst has to draw its CID. CID is a structured representation of the perceptions, beliefs or values of the various stakeholders that define the problem.

Analysis of the CID facilities identification of a small number of elements crucial for the problem situation. The system Analysis Aid provides a graphical editor which will enable the user to draw the CID and also perform loop analysis, merge point and burst point analysis of the diagram. These analysis results are stored in a matrix form for future use.

### 4.1.2 Problem evaluation and solution synthesis

The objective of the system can be redefined as, ***Analyst Aid*** is a software tool, which assists an analyst through out the understanding phase of the software development life cycle. The thing from which we have to start our work is the problem descriptive scenario of any problem. Thus ***Analyst Aid*** should include the following functionality.

a. Identify the key elements from a problem descriptive scenario

b. Perform the SNAC analysis of the document

c. Specify the relationships between the SNAC components and store it in a matrix

d. Provide a graphical editor which facilitates to draw a CID of the problem

e. Perform the analysis of the CID, which includes the loop analysis, merge point and burst point analysis

f. Represent the CID in a matrix form

#### Solution synthesis of the key element identification function

The first thing that we have to take into consideration is that identifying key elements is a natural language processing problem. For any kind of NLP problem which need to deal with the grammar, the initial step is to find the part of speech of each word or token occurring in the document. So we have to specify to which part of speech each word belongs. The corresponding part of speech should be attached to each word so that the words could be manipulated after this. This leads to the idea of tagging each word with its parts of speech.

A parts of speech tagger named Brill Tagger was found as a solution to this. This tagger is a context sensitive rule based tagger that makes its predictions almost 95-97% accurate for the parts of speech that the word is likely to be. Now as the tagging is accurate to such a high percentage the further processes which is based on this will also will not be less error prone.

Key elements are formed by the noun and verb phrases in the document. Now we have to identify these phrases. For that grammar rules for these phrases are written and let a parser to detect these grammars. The parser goes through the document and finds these phrases and writes it into a file. Thus the system outputs a set of noun and verb phrases that can be the key elements. The user has to select the required ones from the list of keywords. ***Analyst Aid*** should provide an interface to the user to select and deselect the list of key words from the document.

**Solution synthèsis of SNAC analysis**

This also involves the lexicographical analysis of the document. In finding the stakeholders and needs the document itself provides some clues. The nouns in the document correspond to the stakeholders. So we need to extract these nouns from the document to get the stakeholders. But what has to be taken into consideration is that there can be many unimportant nouns in the list. The system should be able to minimize this stakeholders list to some extent.

**Solution synthesis of relationship representation**

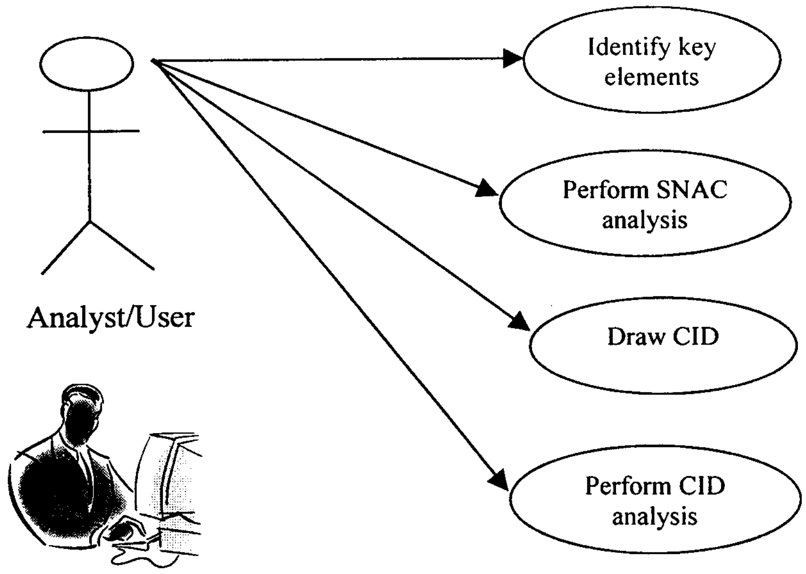
The SNAC components identified from the document should be analyzed to understand the relationship between them. That is, if there are a set of stakeholders, then we have to find which stakeholder is connected to which need, in fulfilling which need a constraint or an alterable comes etc. Finding these relationships helps the analyst to understand the problem clearly. The system cannot find the relationship on its own, but it can provide an interface to the user to specify the relationships. These relationships between the SNAC components are to be stored in a matrix for future use.

**Solution synthesis of the CID editor**

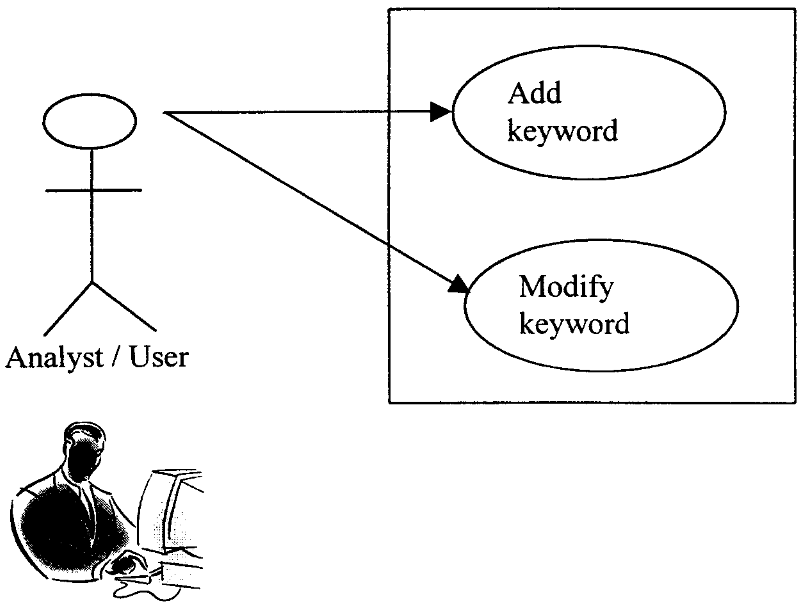
Another method for analyzing the problem is CID. Drawing the CID of a problem helps the analyst to structure the problem and also identify the critical factors involved in solving the problem. Developing a graphical editor plus an analyzer that will do the analysis of the graph drawn can facilitate this. The graphical editor should have a good user interface. As Linux is the platform chosen for doing the project, and C++ the language for programming, using Qt library functions for designing the user interface will have robustness to the system. Qt supports many functions for a good user interface design.

## 4.2 USE CASES

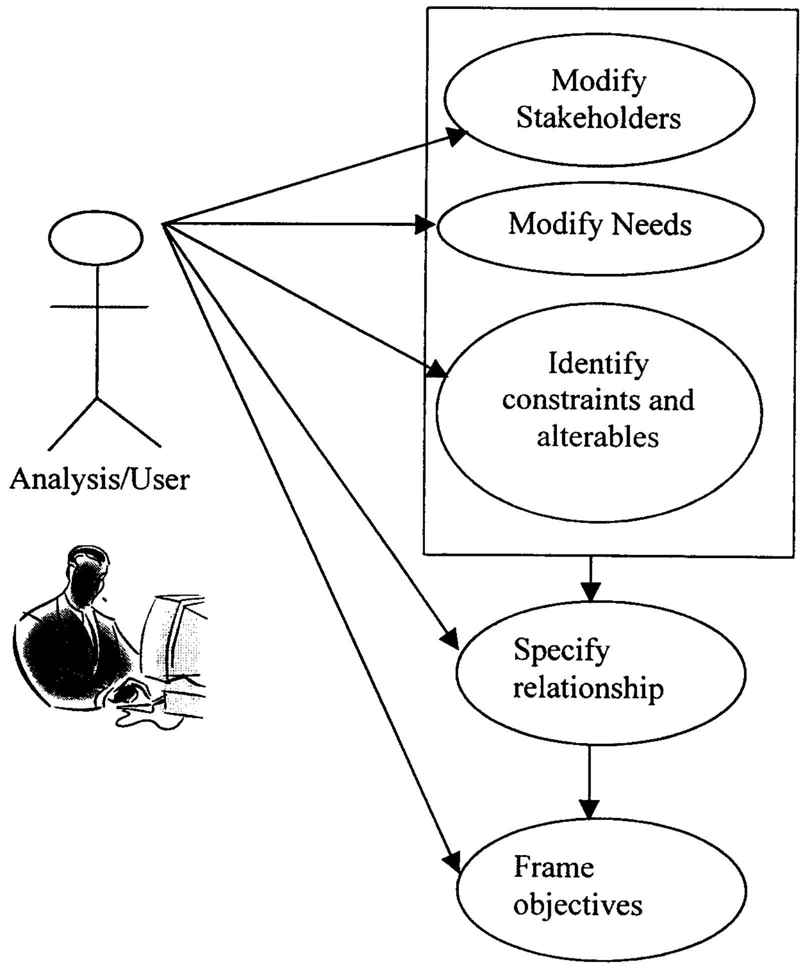
*Analyst Aid*



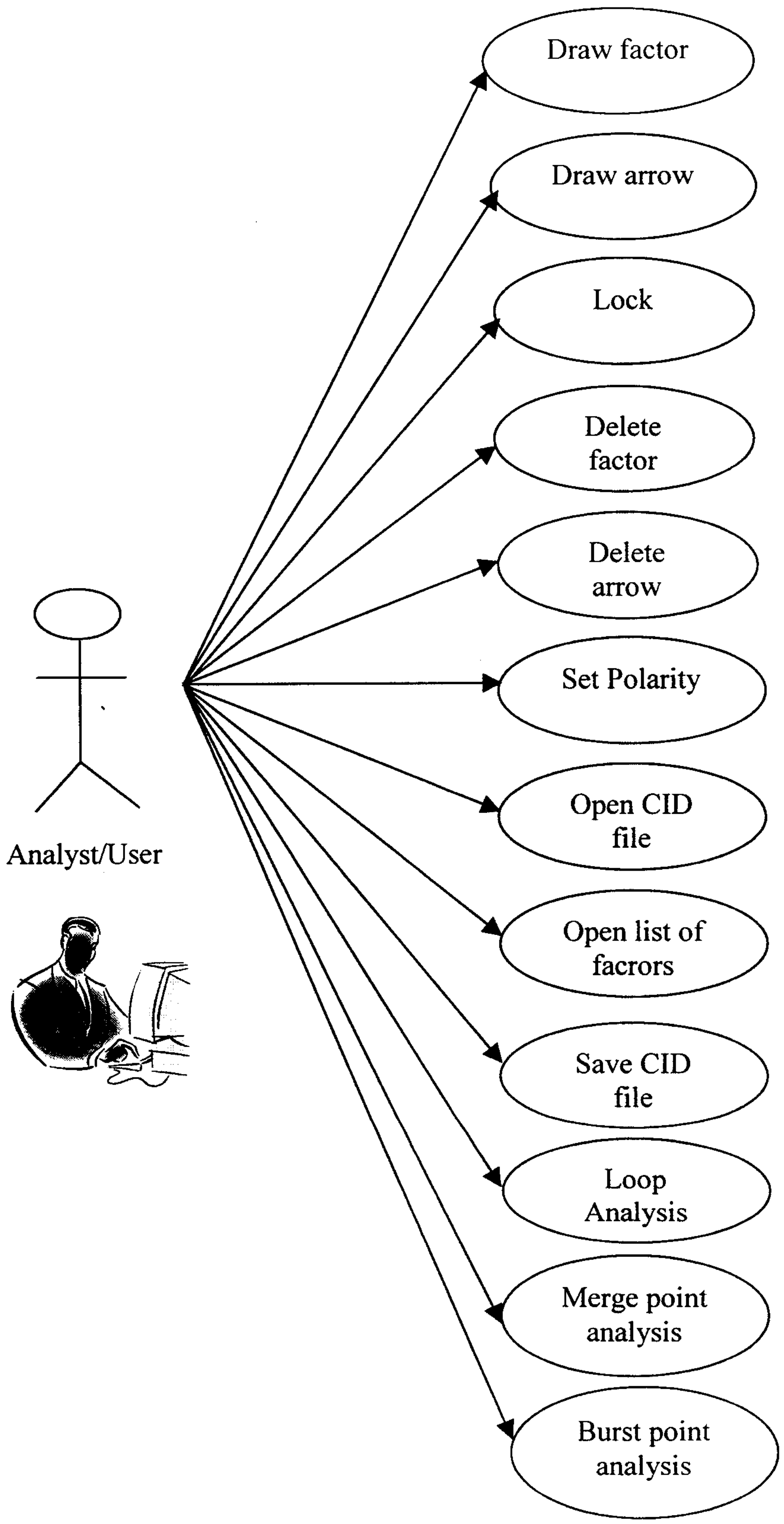
*Identifying of key elements*



***SNAC analysis***



## *CID analysis*



## 4.3 Requirement Definition

A software requirement definition is an abstract specification of the services, which the system should provide, and the constraints under which the system must operate. The requirement definition of the system Analysis Aid can be defined as follows.

The software should enable the user for a better understanding of the consultancy problem to be solved with the help of SNAC analysis and CID analysis. By performing the SNAC analysis the user is able to define the inter-linkages between the SNAC components. These inter linkages help the user to set the objectives to solve the problem. CID is a problem structuring method. The structured representation helps the user to find the perceptions, beliefs of values of the various stakeholders that define the problem.

## 4.4 Requirements Specification

The specifications of the software are described in detail in requirement specification. There are different methods for specifying requirements like structural language specification, graphical notations, mathematical specifications etc. The specifications of the functions that the software should support are obtained from the analysis of the problem. Thus the functions of the system can be listed as below.

### 1. Identification of key elements

The key elements are the words or phrases that have significant relevance to the situation described in the problem. Each element represents a set of ideas, keeping in view the context within which these elements are used. The system should be capable of listing the potential key elements from the lexicographical clues obtained from the document. The list of key elements should not contain redundant data. The user may be allowed to add new keywords from the document to the system-generated list. There should also be a provision to delete the unwanted keywords from the list. The generated list of keywords may be stored for later use.

## 2. Performing SNAC analysis of the document

SNAC analysis process requires classification of the SNAC components and then specifying the relationships between them. The classification is done with the guidance of the user. The stakeholders are found by performing lexicographical analysis of the document. The system should be able to guess the probable stakeholders with the help of some dictionary, which gets enriched intelligently. Some of the needs can also be identified from the document using lexicographical analysis. The user has to mark the alterables and constrains in the document. The user has a role in finalizing the component list. So he should be able to modify this list according to his need.

To understand the system more, the user needs to analyze the relationships between the SNAC components. For this a cross-interaction matrix connecting these components is to be prepared. The relationships that are to be specified are between Stakeholder and Need (S-N), Need and Alterable (N-A), Need and Constraint (N-C), an finally Constraint and Alterable (C-A). For each set a matrix is generated showing the interaction. The structure of the matrix is as given below. Here the interaction between S-N is taken into consideration.

|  | N1 | N2 | N3 | .........upto Nm |
|---|---|---|---|---|
| S1 | 1 | 0 | 1 |  |
| S2 | 0 | 0 | 0 |  |
| S3 | 1 | 1 | 1 |  |
| Upto Sn |  |  |  |  |

This cross-interaction matrix represents that the stakeholder S1 is connected to needs N1 and N3. The stakeholder S2 has no needs and the stakeholder s3 is connected to all the needs listed here. The matrix can be of order m x n matrix where m is the total number of needs and n total number of stakeholders.

The user is the entity who specifies the relationships between the components. Hence the system should provide an interactive interface for recording these relationships.

### 3. Provide graphical editor for drawing the CID

The CID is an efficient method for structuring a problem. The graphical editor allows the user to represent the problem graphically. The user can also load the list of key elements derived from SNAC analysis phase. The graphical editor is capable of performing graph analysis. The graph analysis includes loop identification, merge point analysis and burst point analysis.

The graphical editor should include the following functionally to facilitate CID drawing/analysis:

a. Draw a factor or add a factor to the existing graph.
b. Modify the name and position of the factor.
c. Delete the factor from the graph
d. Draw an arrow between two selected factors.
e. Delete an arrow between two selected factors.
f. Load a CID file to the editor.
g. Create a new instance of the editor.
h. Load the list of factors from the SNAC analysis phase.
i. Save the CID to a file
j. Perform loop analysis of the graph.
k. Identify Merge points and Burst points.
l. Set the polarity of the arrows.

Drawing the CID with the factors obtained the SNAC analysis phase helps the user in verifying the validity of the objectives derived from the system.

The functions of the system are represented using the standard functional specification method.

### 4.4.1. Functional specifications of SNAC module

1.

| | |
|---|---|
| **Function** | Identify key elements |
| **Description** | Key elements relevant in the document are identified and listed. The key elements are the noun phrases and verb phases in the text. Requires lexical and syntactical analyzer to find these phrases. |
| **Inputs** | Text file describing the consultancy problem. |
| **Outputs** | List of potential key elements |
| **Destination** | The user selects the required key elements from the list of potential key elements. |
| **Requires** | The user should click the identify button. |
| **Pre-condition** | The user should open the text describing the problem onto the application's text editor. |
| **Post-condition** | Nil |
| **Side-effects** | None |

2.

| | |
|---|---|
| **Function** | Perform SNAC analysis |
| **Description** | Finds out the S, N, A and C components in the document. Involves several sub activities in doing SNAC analysis. |
| **Inputs** | Key elements identified from the document. |
| **Outputs** | Objectives to be realized to solve the problem. |
| **Destination** | Specifies the relationship between the SNAC components. This helps the analyst in understanding the system. |
| **Requires** | The user to invoke the SNAC button. |
| **Pre-condition** | The user should open the text describing the problem onto the application's text editor. |
| **Post-condition** | The components are listed to the user to specify relationships. |
| **Side-effects** | None |

3.

| | |
|---|---|
| **Function** | Extract nouns |
| **Description** | Extracts the nouns/group of nouns occurring consecutively throughout the document and writes it to a file. |
| **Inputs** | Text file with parts of speech tagged. |
| **Outputs** | File containing the superset of stakeholders/nouns. |
| **Destination** | The component stakeholder is obtained from this function. The user can add new stakeholder or delete existing stakeholders from the list. |
| **Requires** | The user should invoke the SNAC analysis function of the application. |
| **Pre-condition** | The parts of speech should be tagged. |
| **Post-condition** | Nil |
| **Side-effects** | None |

| | | |
|---|---|---|
| 4. | **Function** | Prune stakeholders. |
| | **Description** | The initial list of stakeholders obtained by extracting nouns from the document contains many irrelevant stakeholder types. This function uses stakeholder dictionary and an efficient search algorithm to prune this list of stakeholders. Redundant stakeholders are also eliminated. |
| | **Inputs** | File containing list of stakeholders. |
| | **Outputs** | Pruned list of stakeholders. |
| | **Destination** | Displays the list of stakeholders to the user to make a final decision about its importance. |
| | **Requires** | Stakeholder dictionary. |
| | **Pre-condition** | The user should invoke the SNAC analysis function of the application. Also grouping and extracting of nouns have to be done. |
| | **Post-condition** | The user should be able to add/delete the stakeholders from this list. |
| | **Side-effects** | Stakeholder dictionary can get updated. |
| 5. | **Function** | Extract needs |
| | **Description** | Identifies and extracts the need patterns in the document. |
| | **Inputs** | Problem description text file that is tagged. |
| | **Outputs** | File containing the list of needs. |
| | **Destination** | Displays the list of needs to the user to make a final decision about its importance. |
| | **Requires** | A scanner that identifies the need pattern. |
| | **Pre-condition** | The user should invoke the SNAC analysis function of the application. Also grouping of nouns has to be done. |
| | **Post-condition** | The user should be able to add/delete need patterns from this list. |
| | **Side-effects** | None |
| 6. | **Function** | Identify alterables and constraints. |
| | **Description** | Alterables and constraints cannot be differentiated by the lexicographical clues. Therefore the user has to select and mark the pattern in the document as an alterable or constraint. Already existing patterns are not allowed to mark. |
| | **Inputs** | Plain problem description file. |
| | **Outputs** | List of alterables and constraints. |
| | **Destination** | Displays the list of alterables and constraints to the user to make a final decision about its importance. |
| | **Requires** | User intervention to mark the alterables and constraints. |
| | **Pre-condition** | The user should invoke the SNAC analysis function of the application. Also grouping of nouns has to be done. |
| | **Post-condition** | The user should be able to add/delete patterns from this list. |
| | **Side-effects** | None |

| | | |
|---|---|---|
| 7. | **Function** | Specific relationship between SNAC components. |
| | **Description** | The user with an interactive interface specifies relationship between S-N, N-C, N-A and C-A. Generates a cross-interaction matrix of these components. |
| | **Inputs** | Files containing list of SNAC components. |
| | **Outputs** | A matrix representation of the relationships. |
| | **Destination** | The user can frame objectives of the organization from these relationships. |
| | **Requires** | User intervention to make the relationship between the different components. |
| | **Pre-condition** | The user should invoke the SNAC analysis function and mark relationship function of the application. |
| | **Post-condition** | The user should be able to load new problem description text and continue with the analysis process right from the beginning. |
| | **Side-effects** | None |

### 4.4.2. Functional Specification of CID Module

| | | |
|---|---|---|
| 1. | **Function** | Draw factor |
| | **Description** | Adds a factor to the CID editor screen. The user should specify the position, name, font size and font style of the factor. When added to the diagram, the factor becomes the current selection. The user chooses the factor position by moving the cursor to the area where the factor is to be added. |
| | **Inputs** | The name, position, font size, font style of the factor. |
| | **Outputs** | Modified CID with factors added. |
| | **Destination** | Parameters of the CID is stored in a data structure. |
| | **Requires** | The user to invoke the draw factor function. |
| | **Pre-condition** | The CID editor should be in draw mode. |
| | **Post-condition** | The parameters of the CID should be updated. |
| | **Side-effects** | None |
| 2. | **Function** | Add arrow |
| | **Description** | Adds an arrow between two factors. The user adds arrow by selecting the from-factor and to-factor of the arrow. The polarity of the arrow is also be specified. |
| | **Inputs** | The from-factor and the to-factor of the arrow to be drawn. |
| | **Outputs** | Modified CID with arrows added. |
| | **Destination** | The application stores parameters of the arrow into a data structure. |
| | **Requires** | The user to invoke add arrow function. |
| | **Pre-condition** | The CID editor should be in draw mode. |
| | **Post-condition** | The parameters of the CID should be updated |
| | **Side-effects** | None |
| 3. | **Function** | Lock screen |
| | **Description** | Locks the screen thereby preventing the user to invoke all |

| | | |
|---|---|---|
| | | other functionality. |
| | **Inputs** | Nil |
| | **Outputs** | Nil |
| | **Destination** | Disables all the draw functionality. |
| | **Requires** | The user to invoke lock function. |
| | **Pre-condition** | The editor should be in draw mode. |
| | **Post-condition** | The parameters of the CID should be updated. |
| | **Side-effects** | None |
| 4. | **Function** | Delete factor |
| | **Description** | Deletes the factor from the current location in the diagram. |
| | **Inputs** | Factor that is to be deleted. |
| | **Outputs** | The modified CID |
| | **Destination** | Parameters of the CID are updated and stored in a data structure. |
| | **Requires** | The user to invoke delete factor function. |
| | **Pre-condition** | The editor should be in draw mode. |
| | **Post-condition** | The parameters of the CID should be updated. |
| | **Side-effects** | None |
| 5. | **Function** | Delete arrow |
| | **Description** | Deletes the arrow between two factors. The user deletes the arrow by selecting the from-factor and to-factor of the arrow. |
| | **Inputs** | The from-factor and to-factor of the arrow to be deleted. |
| | **Outputs** | Modified CID |
| | **Destination** | Parameters of the CID are updated and stored in a data structure. |
| | **Requires** | The user to invoke delete arrow function. |
| | **Pre-condition** | The editor should be in draw mode. |
| | **Post-condition** | The parameters of the CID should be updated. |
| | **Side-effects** | None |
| 6. | **Function** | Identify merge points |
| | **Description** | Finds the merge points in the diagram and highlights them in the CID |
| | **Inputs** | Matrix representation of the CID |
| | **Outputs** | CID with merge points highlighted. |
| | **Destination** | The user gets an idea about the importance of the factor and thereby helps to find the complexity of the diagram. |
| | **Requires** | The user to invoke merge point analysis function. |
| | **Pre-condition** | The editor should be in draw mode. |
| | **Post-condition** | Nil |
| | **Side-effects** | None |
| 7. | **Function** | Identify burst points |
| | **Description** | Finds the burst points in the diagram and highlights them in the CID. |
| | **Inputs** | Matrix representation of the CID. |

| | | |
|---|---|---|
| | **Outputs** | CID with burst points highlighted. |
| | **Destination** | The user gets an idea about the importance of the factor and thereby helps to find the complexity of the diagram. |
| | **Requires** | The user to invoke burst point analysis function. |
| | **Pre-condition** | The editor should be in draw mode. |
| | **Post-condition** | Nil |
| | **Side-effects** | None |
| **8.** | **Function** | Perform loop analysis |
| | **Description** | Identifies the loops in the diagram and perform loop analysis, and finds the resultant polarity of the loop. |
| | **Inputs** | Matrix representation of the CID |
| | **Outputs** | Displays the total number of loops found, each loop and its resultant polarity. |
| | **Destination** | Gives an idea of the complexity of the diagram. |
| | **Requires** | The user to invoke loop analysis function. |
| | **Pre-condition** | The editor should be in draw mode. |
| | **Post-condition** | Nil |
| | **Side-effects** | None |
| **9.** | **Function** | Load an existing CID file |
| | **Description** | Loads a CID file with '.cid' extension in the graphical editor. |
| | **Inputs** | CID file |
| | **Outputs** | CID of the '.cid' file loaded. |
| | **Destination** | Stores CID analysis information. |
| | **Requires** | A file with '.cid' extension |
| | **Pre-condition** | The file with the '.cid' extension should strictly be in the format for which the application is designed. |
| | **Post-condition** | Nil |
| | **Side-effects** | None |
| **10.** | **Function** | Load list |
| | **Description** | The user can open the key element list to a combo box in the application. |
| | **Inputs** | A text file containing key elements |
| | **Outputs** | The populated combo box. |
| | **Destination** | The user can assign the name of the factor as the key element loaded in the combo box of the application. |
| | **Requires** | A text file containing key elements. |
| | **Pre-condition** | Identification of key elements. |
| | **Post-condition** | The combo box is populated. |
| | **Side-effects** | None |

# SOFTWARE DESIGN

# 5. SOFTWARE DESIGN

Software design mainly focuses on data, architecture, interfaces and components design. Each of the analysis models provides information that is necessary to create the design models required for a complete specification of design. Thus the design task produces a data design, architectural design, interface design and component design.

## 5.1 Architectural Design

The architectural design defines the relationship between major structural elements of the software. The architectural design representation can be derived from they system specification, analysis model, and the interaction of subsystems defined within the analysis model. An object oriented design approach is used in designing the architecture of the system. The overall architecture specifies that there are two modules in the system; the SNAC analysis module and the CD analysis module. These two subsystems interact with each other to reach the objectives of the system.

The system ***Analyst Aid*** mainly comprises of two subsystems – Stakeholders, Needs, Alterables, Constraints (SNAC) analysis module and Cybernetic Influence Diagram (CID) analysis module. In the SNAC analysis module the user/analyst can perform the SNAC analysis of the problem descriptive scenario of the system to be solved. The SNAC analysis results in a cross-interaction matrix between the SNAC components from which the user can frame the objectives of the problem whose solution is to be given by the analyst. In the CID analysis module he can draw the CID connecting the key factors/components in the problem. Drawing the CID enables the analyst to understand the problem and helps him to find the complexity of the problem. The CID module also helps the analyst in performing the merge-point, burst-point and loop analysis of the CID. The analyst can use also use the application as a validation tool. That is the can find the SNAC components from the SNAC phase and frame a set of objectives. Then he can draw a CID connecting these components to realize the objectives framed. This will give the analyst a visualization of the problem. Visualization of the key factors in the problem influencing each other, helps in reducing the gaps in perceiving and understanding the problem. So if the analyst feels that all components and relationships are not identified in the primary step he can go back to the SNAC module and repeat his analysis.

The following diagram can depicts the structure of the system.

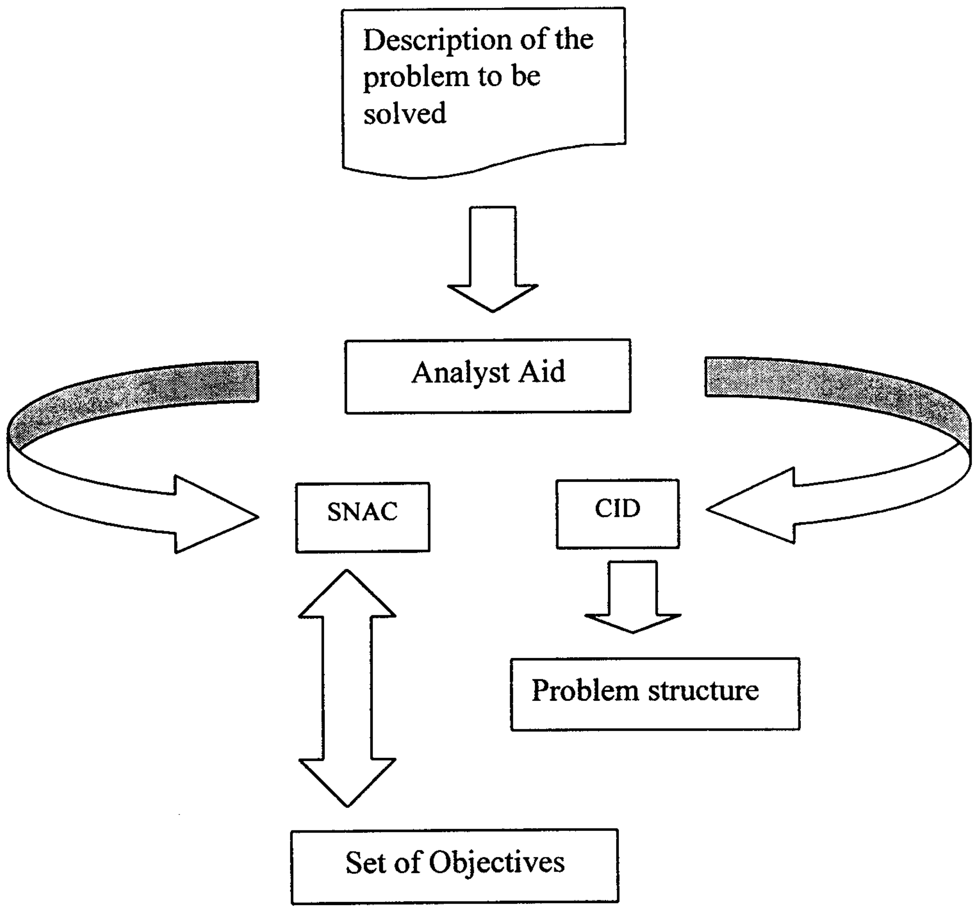


***Fig. 1***

The objectives of the SNAC module is to perform SNAC analysis of the document which describes the consultancy problem. The SNAC analysis comprises a series of steps involving lexicographical analysis of the problem description. The steps involved are shown in

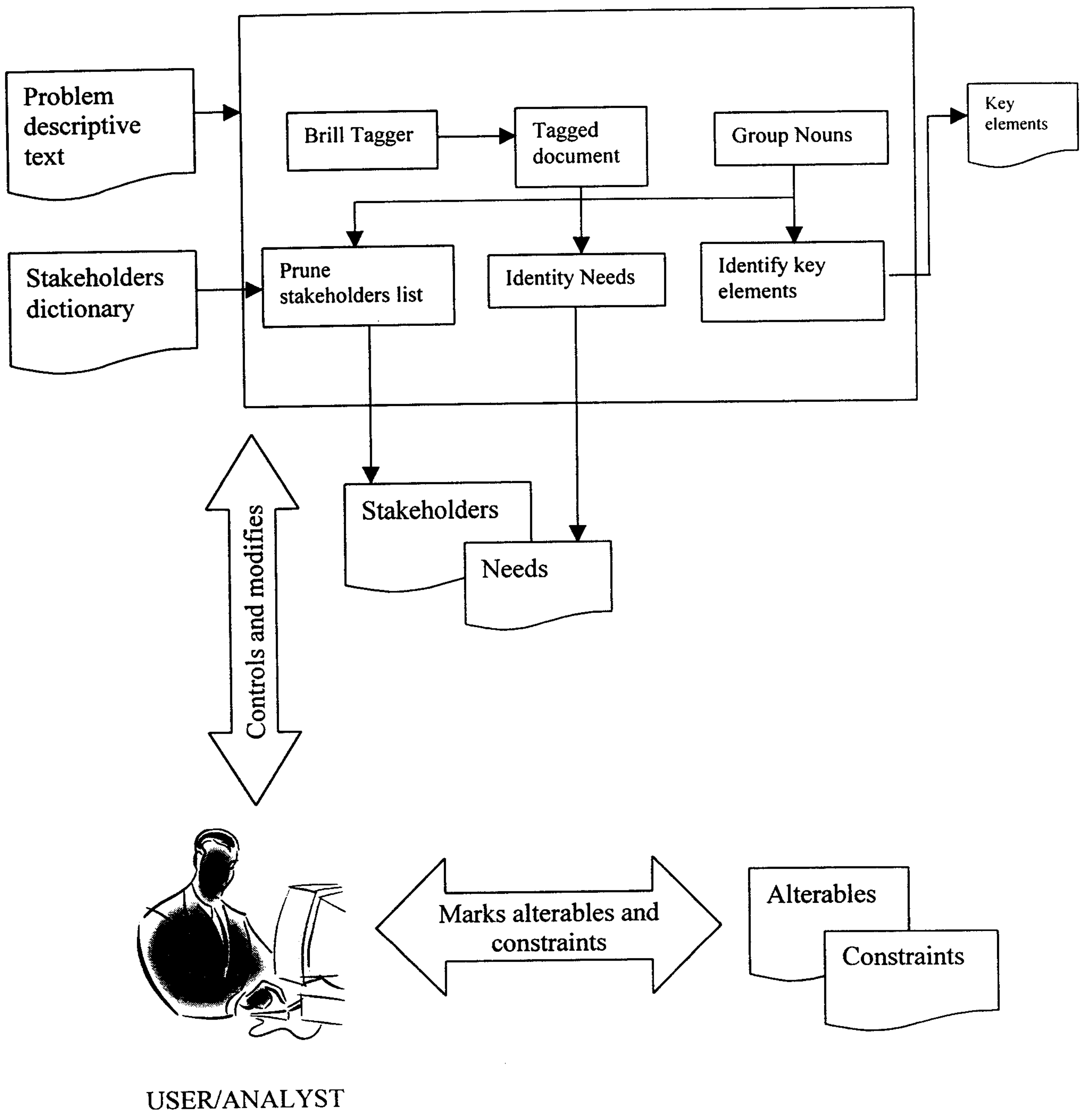
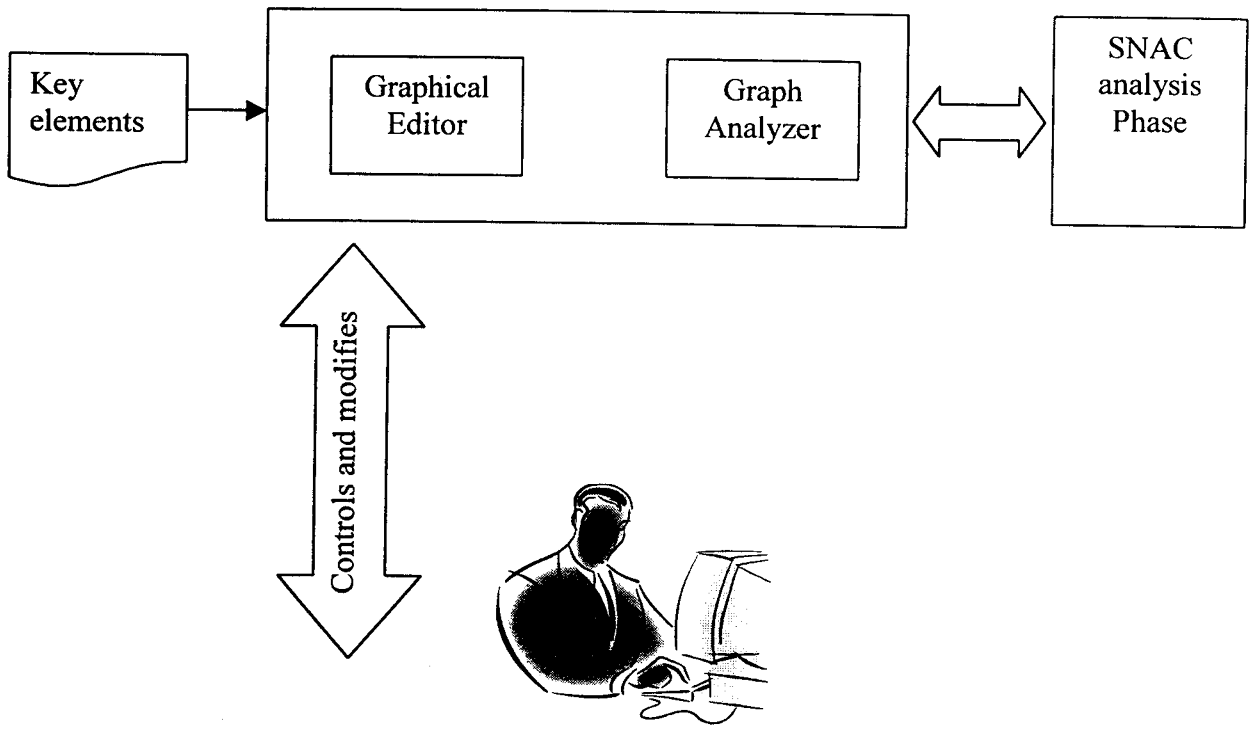

Problem descriptive text
Stakeholders dictionary
Brill Tagger
Tagged document
Group Nouns
Key elements
Prune stakeholders list
Identity Needs
Identify key elements
Controls and modifies
Stakeholders
Needs
Marks alterables and constraints
Alterables
Constraints
USER/ANALYST

*Fig 2*

The CID module contains a graphical editor which helps in drawing the CID and also an analysis part which does the analysis of the diagram. The structure of the CID module can be depicted using the diagram.



USER/ANALYST

***Fig 3***

### 5.1.1 Abstract specification of the SNAC module

To the SNAC phase, the problem descriptive text is given as the input. The output of this phase is a set of objectives needed to sole the problem. The user of the system plays an important role in transforming the input into output.

The system needs to go through the text describing the problem and hence requires a lexicographical analyser. The first step in doing the lexicographical analysis is, to identify the parts of speech of each word in the document. There are several techniques to identify this. One can use a set of parts of speech dictionaries which consist of all the common words that occur in English language. That is a noun dictionary consisting of all the nouns, a preposition dictionary with prepositions, an adjective dictionary with adjectives and like wise. The program can take each word in the descriptive text and tag it with the appropriate tag. But there can be situations where no parts of speech dictionary contain the match. In that case the word can be treated as noun. The problem with this method is that sometimes a word can occur in more than one parts of speech form. The decision has to be made depending on the previous one or two words occurring in the sentence. Thus the decision is context sensitive. For e.g. the word ***look*** can occur both as a noun and a verb. As this program does not take the context of tagging into consideration, an alternate tagging technique is adopted. This tagging technique uses an already available tagger named ***Brill Tagger*** which is context sensitive.

This tagger makes use a file LEXICON, which is a dictionary consisting of almost 93696 different words with the tag of the word specified on its side, the word can have more than one tags. The tag is given according to their frequency of occurance. The most likely tag is written first and the tags following have less frequency of occurance. Brill Tagger is a rule based tagger. It consists of a set of rules to be applied to the word depending on the context. Brill Tagger is found to be highly efficient and makes its prediction 95-97% accurate.

Once tagging is done, the next important module in SNAC phase is the one which identifies key elements in the document. Key elements are the words and phrases that have significant relevance to the situation mentioned in the descriptive scenario. The key elements are assumed to be the noun phrases and verb phrases occuring in the document. To identify the key elements a parser can be used. The needs of the stakeholders can also be found with the help of the parser, since needs occur in a specific pattern. The needs are the verb phrases in the document.

Eg of a need pattern ***to + implement + actions***

The parser will identify the occurrence of such patterns in the input text.

The alterables and constraints cannot be differentiated using the lexicographical clues in the document and it is the analyst/user who has to identify them. They system should provide an interactive interface to specify the relationships between these SNAC components and thereby arriving at the objectives of the organisation.

Functional design specification of some of the modules in the SNAC phase is given below:

1.

| | |
|---|---|
| **Function** | Format the problem description text file for the tagger. |
| **Description** | The input text is to be formatted in such a way that each line should contain only one sentence. The punctuation marks should be separated with a space. |
| **Inputs** | Problem description text. |
| **Outputs** | Formatted text is written onto a temporary file. |
| **Destination** | The tagger requires its input in this specified format. |
| **Requires** | A scanner that formats input text to the tagger's format. |
| **Pre-condition** | The user should invoke the identifying key elements function or the SNAC analysis function of the application. |
| **Pot-condition** | The input text content remains unchanged. |
| **Side-effects** | None |

2.

| | |
|---|---|
| **Function** | Remove spaces |
| **Description** | The scanner is likely to insert unwanted spaces in its output file, which is formatted for the tagger. These spaces are removed. |
| **Inputs** | The output temporary text file from the scanner that formats the text for the tagger. |
| **Outputs** | Spaces removed text is written onto a temporary file. |
| **Destination** | The tagger requires its input in this specified format. |
| **Requires** | A scanner that detects and removes extra spaces. |
| **Pre-condition** | The use should invoke the identifying key elements function or the SNAC analysis function of the application. |
| **Post-condition** | Te input text content remains unchanged. |
| **Side-effects** | None |

3.

| | |
|---|---|
| **Function** | Tag the input text with parts of speech. |
| **Description** | Tags each word in the input text with the corresponding parts of speech tag. Also keeps track of the context under which each word is tagged. |
| **Inputs** | Formatted text file |
| **Outputs** | Tagged file |
| **Destination** | The parts of speech each word in the input text is to be known for performing lexicographical analysis. |

| | | |
|---|---|---|
| | **Requires** | Brill Tagger and supporting files. |
| | **Pre-condition** | Input to the tagger should in strict format. The user should invoke the identifying key elements function or the SNAC analysis function of the application. |
| | **Post-condition** | Nil |
| | **Side-effects** | None |
| **4.** | **Function** | Group nouns |
| | **Description** | Groups consecutively occurring nouns, *noun group* preceded by one or two adjectives, *noun group* with the preposition 'of' in between. |
| | **Inputs** | Text file with parts of speech tag attached to each word. |
| | **Outputs** | Text file containing grouped nouns. |
| | **Destination** | Grouping nouns facilitates identifying stakeholders and also writing grammar for the parser generator. |
| | **Requires** | Scanner that identifies the patterns to be grouped |
| | **Pre-condition** | The user should invoke the identifying key elements function or the SNAC analysis function of the application. |
| | **Post-condition** | Content of the text file should remain intact. |
| | **Side-effects** | None |

### 5.1.2 Abstract specification of the CID module

The system CID consist of 2 sub modules, drawing the CID and performing the analysis of the diagram. As every graphical editor, the CID editor also contains facilities to draw/delete a factor in the CID, draw/delete an arrow and move the factors/arrow in the diagram with the help of a mouse.

The analysis part of the CID performs different types of analysis like

- Merge point analysis
- Burst point analysis and,
- Loop analysis

Merge points are those nodes where convergence of many arrows takes place while burst points are the nodes from which a lot of arrows diverge. These elements are potent because they have ramifications for a large number of other elements.

Merge points are to be found out depending on the condition that number of incoming arrow should be greater than a threshold number. The burst point is also found using this method but here the outgoing arrows are taken into consideration. Identification of the feedback loops also facilitates review of the cause and effect as perceived by the stakeholder.

The user interface of CID editor is designed to be highly interactive. It shows a Windows like interface and gives appropriate messages to the user, thereby making his job easy. The user interface is designed keeping Qt – a cross-platform C++ GUI application framework in mind which gives flexibility in adding a wide range of functionality to the system.

## 5.2 Data Design

The data design transforms the information domain model created during analysis into the data structures that will be required to implement the software.

The most important data structure used in the system ***Analysis Aid*** is the data structure file. Almost all the files used are text files in which each byte represents one character according to the ASCII code. This in contrast with a binary file, in which there is no one-to-one mapping between bytes and characters. ASCII files are sometimes called plain txt files. The language used for developing the system is C, and C++. Both the languages are rich with file manipulation functions. The Qt library also provides a set of functions facilitating file manipulation. The data files used are listed below.

### i. problem description file

This file is a normal text file describing the problem to be solved. The file is not allowed to contain any sort of formatting. The headings that usually come in these type of input texts are to be avoided. There is no limit in the length of the file.

**ii. file with 'for' extension**

This is the file, which comes as the output from the scanner that formats the input text file. This formatting is done in such a way that only one sentence is there in a line. Thus an EOL (end of line) character separates each sentence. The punctuation marks occurring in the text are also separated by a space. This formatting is done exclusively for the Brill Tagger.

For. e.g. If the input text is:

**John (I think) said: "Who are you?" He then gave me $10.**

would become:

**John (I think) said: "Who are you?"**

**He then gave me $ 10.**

**iii. file with '.rem' extension**

The scanner can introduce some extra blank spaces into the output document after the formatting is done. The file with '.rem' contains problem descriptive text in the tagger's format with unwanted spaces removed.

**iv. file with '.tag' extension**

This file is obtained as the output file from the Brill Tagger. The tagger will tag each word and punctuation marks in the document.

For e.g. If the input test is:

**This is not a program of socialized medicine.**

would become:

**This/DT is/VBZ not/RB a/DT program/NN of/IN socialized/VBN medicine/NN./.**

Here Dt, VBZ, RB, NN, IN, VBN are tags associated with different parts of speech.

v. **file with '.grp' extension**

The tagged document file is taken and the consecutively occurring nouns in that file are grouped into small nuggets in the <GP> .............. </GP> tags. The text with the grouped nouns is written into the file with .grp extension. The other words are kept intact in this tile.

vi. **file with '.txt' extension**

This system uses 4 files with .txt extension. Of which two of them output.txt and sneed.txt are outputs from the parser that identifies the noun and verb phrases and need patterns in the document. The contents in these files are in the format given below.

*output.txt:*

**{Insurance industry}<NPH> {undergoing dramatic changes} <VPH> {likely entry of foreign and private insurers} <NPH> {increased customer expectations} <VPH>.**

*sneed.txt:*

**{to implement the provisions} <ND>.**

**{providing social security benefits} <ND>.**

**{meeting certain requirements} <ND>.**

The other two files are **cachecat.txt** and **fullcat.txt**. These files serve as stakeholder dictionary. Cachecat.txt contains the most probable words that occur as stakeholders in typical application problems and fullcat.txt contains other set of words that have a less chance of being stakeholders. So some domain specific knowledge assumed while creating these dictionaries. The nouns groups in the text file that are enclosed in <GP>.....</GP> tags are the probable stakeholders. But all of them are not required. The list has to be pruned to get only the stakeholders that come in typical application orientation problems. The cachecat.txt file is searched first to increase the efficiency.

**vii. file with '.slist', '.need', '.alt', '.cons'**

These are the respective text files that contain the list of stakeholders, set of needs, alterables and constraints that are found in the document.

**viii. file with '.key' extension**

This file contains the key elements that are identified from the document.

**ix. file named LEXICON**

The file name LEXICON is the data file that the tagger uses to find the tag of the words in English language. Each record in the tagger is in the general format

**word tag1 tag2 tag3.......tagn**

Tag1 denotes the most likely tag of the word and the following tags denote the other probable tags of the word. The structure of the LEXICON has not been altered but the tags ending with '$' sing in the file have been changed.

For example PRP$ - possessive pronoun

WP$ - possessive wh-pronoun

The parser does not support $ in its input, the reason behind this being the fact that the symbol '$' is used to refer the value in the value stack used by the parser. Thus $$ shows the top of the stack, $2, the second element in the stack and so on. Hence the tag PRP$ needs to be changed to PRPO and WP$ to WPO.

As already mentioned the tagger is a rule-based tagger and consists of a set of rules to be followed in certain contexts. A sample set of rules in the rule file is given below.

**NNs fhassuf 1 NNS x**

**NN. fchar CD x**

**NN – fchar JJ x**

**NN ed fhassuf 2 VBN x**

**NN ing fhassuf 3 VBG x**

The first line denotes that if the tag is NN (i.e. noun) and the word has a suffix 's' with length 1 should be changed to NNS (denoting Nouns or noun in plural form).

a new set of rules which, identifies the date patterns in the document is added to the system. Only a specific format of date is supported i.e.dd-mm-yyyy or dd-mm-yy. For supporting this hyphenated version of date the rules added are

CD – fchar RG x

RG – fchar DAT x

If a cardinal number has a hyphen been it is marked as a RG, range and if an RG has again a hyphen then it will be marked as a DAT, indicating **date** tag. These were the changes made in the data files used by the tagger. Structure of the matrix showing inter linkages.

**x. A matrix, that represents the relationship between the SNAC components.**

The matrix is a cross-interaction matrix. There are four matrices representing the relationships between S-N, N-A, N-C and C-A.

**xi. Structure of the '.cid' file**

In the CID module a diagram that is saved using the editor is saved in a file with '.cid' extension. This file is preserved in a particular format. The file contains the following information.

The file starts with a header called CID header.

This header is followed by the description of the factors in the diagram. They have the format.

| | |
|---|---|
| ➢ factor id | - to identify the factor |
| ➢ in-arrow id | - to identify the in-arrow |
| ➢ out-arrow id | - to identify the out-arrow |
| ➢ x_pos | - x position on the screen of the factor |
| ➢ y_pos | - y position of the factor on the screen |
| ➢ box_width | - width of the box enclosing the factor |
| ➢ box_height | - width of the box enclosing the factor |

Following this, the source and the destination detail of all the arrows in the diagram are also stored as shown below.

| | |
|---|---|
| ➢ from_id | - id of the from factor |
| ➢ to_id | - id of the to factor |
| ➢ pol | - polarity of the arrow |
| ➢ xmiddle drawn | - x middle position of the factor box at which the arrow is |
| ➢ ymiddle drawn | - y middle position of the factor box at which the arrow is |

## 5.3 Algorithm Design

The software ***Analyst Aid*** uses three main algorithms.

a. An algorithm that helps in minimising the list of nouns extracted from the problem description text.

b. Loop analysis algorithm – which does the analysis of the CID.

c. Merge-Burst point analysis algorithm – which finds the merge points and burst points in the CID.

**Trimming the nouns list**

In this algorithm two dictionary filed are used. The first dictionary called ***cachecat.txt*** contains only key words that can occur as stakeholders in a typical application-oriented problem. As the name suggests this serves as a cache for the search, i.e. at first this entire dictionary file is searched and if a match occurs then that word is assumed to be a stakeholder. The match does not mean a dump mach. The word that is to be determined as a stakeholder is allowed to have its sub string in the cachecat. txt dictionary. Thus if a sub string of the word is found in the cachecat.txt dictionary it is considered as a stakeholder. If the cachecat.txt file does not contain any of the sub strings of the word it searches the next dictionary file named ***fullcat.txt***. The whole words, which is to be determined to be a stakeholder or not is searched as such in the fullcat.txt dictionary. The steps of the algorithm are given below.

```
1.  Open nouns list file
2.  While not end of nouns list
    2.1  take each noun
    2.2  open the cachecat.txt dictionary
    2.3  while not end of cachecat.txt file
         2.3.1  search for the word or a sub string of the word in the
                cachecat.txt file.
```

```
        2.3.2  if a match is found go to step 3.
        2.3.3  go to step 2.3.
        2.4    open the fullcat.txt dictionary
        2.5    while not end of fullcat.txt
               2.5.1  search for the full word in the file
               2.5.2  if full word is found go to step 2.6
               2.5.3  go to step 2.5
        2.6    go to step 2.
3.      If the non is not found in fullcat.txt or cachecat.txt go to step 5
4.      The word is assumed t be a stakeholder, go to step 6.
5.      The noun is not a stakeholder.
6.      Close all files.
```

**Loop analysis algorithm]**

This algorithm does in exhaustive search for identifying the loops in the CID. It reads a matrix that denotes the adjacency matrix of the diagram. The search is performed on the whole matrix until all the edges have been traced. The polarity of each loop is found by counting the number of positive edges and negative edges and subtracting them. The result indicates the polarity of the loop. The algorithm can be given as follows.

```
1.      begin
2.      Initializes total_visited_array, current_visited_array, current_path_array and
        path-ten to zero.
3.      Till the end of all the nodes
        3.1 If total visited array equals zero
        3.2 Search the graph
        3.3 If the node array is not empty
        3.4 from path_len-1 till the end
        3.5 if it reaches the same node
                3.6 to 3.3
        3.7 till the end of path_len o he loop stores the nodes in an array
```

```
    3.8  checks for duplicate loops and if not already present
    3.9  adds to the loop array and its polarity
    3.10 got to step 3
    3.11 sets the total_visited_array and node array to 1, increments the path_len
         and current_path to node.
    3.12 till the end of all the nodes
    3.13 checks if it is the same as node and if not same
    3.14 checks for 1 entry in the matrix and if present
    3.15 search for the loop
    3.16 sets the current visited node array to 0 and decrements the path len
4. end
```

The merge and burst point analysis algorithm

Every factor has an in-degree and an out-degree associated with it depending on the number of arrows going into and out of it. The difference between this in-degree and out-degree are calculated. If the in-degree is greater than a threshold value, or equal to the out-degree it is considered as a merge point and if it is the in-degree the point is a burst point. The algorithm for the merge and burst analysis is given below.

```
1.  While not end of the factor array
    1.1 read the in-degree and out-degree of the factor
    1.2 if temp=in-degree-out-degree>0 go to step 1.4.
    1.3 store the temp in the burst array.
    1.4 go to 1.6
    1.5 store the temp in merge array.
    1.6 go to 1.
2.  bubble sort merge array
3.  bubble sort burst array
4.  store the merge burst values in the an array with values 1, and 2 in it, where 1
    represents merge point and 2, burst points.
```

## 5.4 Interface Design

The interface design focuses on two areas of concern: a) the design of interfaces between software components, and c) the design of the interfaces between a human and the computer i.e. user interface design.

a. Interfacing between the software components

There are different software components in the system ***Analyst Aid*** that needs to communicate with each other to work as a whole. These components should be interfaced properly in-order to get the desired results. Design of inefficient interfaces could corrupt the data that is transferred between components. In our system most of the data transfer between components are through member functions of the objects forming these components. Special care is taken in designing member functions that need to interact with other objects and illegal type castings are avoided.

b. User Interface Design

The user interface design creates and effective communication between a person and a computer. The user interface design begins with the identification f user task and environmental requirements. Once the user tasks have been identified, user scenarios are created and analyzed to define a set of interface objects and actions. These form the basis for the creation of screen layouts that depicts graphical design and placement of icons, menus etc.

The user interface of the system ***Analyst Aid*** has a major role in realizing its objectives. The software needs to take its major decisions with the user guidance. So a good user interface should be maintained. The graphical user interface is designed for the system to ease the user effort. The system supports both menus and buttons that invoke the various functionality of the system. The interface mimics typical windows based application software to which users are familiar. The template of the user interface design is given below.

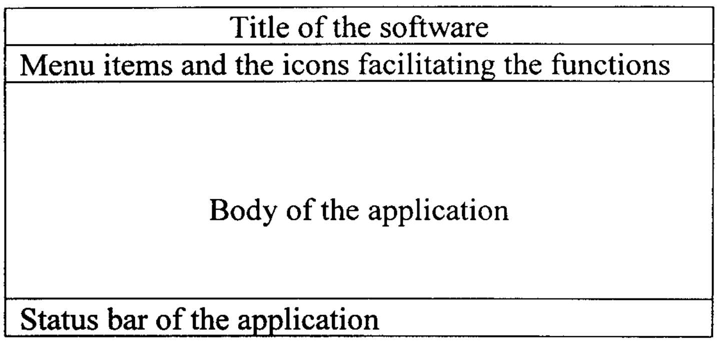

Title of the software
Menu items and the icons facilitating the functions
Body of the application
Status bar of the application

***Fig 4***

The user interface should meet the following requirements

- ✓ Menu selection for all major functions
- ✓ Icons showing the major functions
- ✓ Function short cut keys for easy navigation
- ✓ Visually appealing colors and design of the screen
- ✓ Messages guiding the user
- ✓ Mouse enabled
- ✓ Text and object selection and dragging
- ✓ Right click functionality to the mouse

In the SNAC module the application screen is divided into a 14 by 7 grid layout. The grid layout helps the different user-interaction components to be fixed at different positions in the application. Some of the user-interaction components that are needed in the application are a text editor, a text viewer-to view the text, a check list consisting of different key words found in the system, a list box showing the different SNAC components identified in the system.

The CID graphical editor provides a good interface for drawing the CID. It makes use of the mouse to draw and change the diagram. The editor is equipped with a large amount of facilities that help the user in analyzing the diagram and viewing the analysis result graphically. For example the editor shows the merge and burst points of the diagram highlighted after the merge and burst analysis.

# PROGRAM DESIGN LANGUAGE

# 6. PROGRAM DESIGN LANGUAGE (PDL)

## PDL for Analyst Aid Menu Options

```
Begin
        ⇒ If option selected is new_doc
                ⇒ Open a new application window.
        ⇒ Else if option selected is open_doc
                ⇒ Open the problem description text file to the editor of the application
        ⇒ Else if option selected is save_file
                ⇒ Save the document in the text editor to a file
        ⇒ Else if option selected is save As_file
                ⇒ Save the document in the text editor to a file with a new name
        ⇒ Else if option selected is print_file
                ⇒ Print the document in the text editor
        ⇒ Else if option selected is quit
                ⇒ Close the application
        ⇒ Else if option selected is what's_this
                ⇒ Enable the what is this function
        ⇒ Else if option selected is cut
                ⇒ Cut the selected text from the text editor
        ⇒ Else if option selected is copy
                ⇒ Copy the selected text from the text editor
        ⇒ Else if option selected is paste
                ⇒ Paste the selected text from the text editor
        ⇒ Else if option selected is identify_key_element
                ⇒ Do identify
        ⇒ Else if option selected is snac_analysis
                ⇒ Do snac
        ⇒ Else if option selected is matrix_analysis
                ⇒ Do matrix editor
        ⇒ Else if option selected is cid
                ⇒ Do cid and invoke graphical editor
        ⇒ End if
End;
```

## PDL for identify_key_elements

```
Begin
        ⇒ If text_of_editor edited is TRUE
                ⇒ Call save_file
        ⇒ Else
                ⇒ Format input text for tagger to get a. for file
                ⇒ Remove unwanted spaces in the .for to get .rem file
                ⇒ Tag the .rem file to .tag file
                ⇒ Group the noun patterns in the .tag file to get .grp file
                ⇒ Remove unwanted space in the .gap file to get .rem file
                        ⇒ Parse the .rem file to obtain the noun and verb
                            phrases and write to an output file
                ⇒ Extract the noun and verb phrases from this output file to .keys file
                ⇒ Set status message "Key elements identified"
                        ⇒ Highlight the noun and verb phrases identified and
                            display it in the text view of the application
                ⇒ If more_keywords is TRUE
                    ⇒ Set message "Select and right click to add a new keyword"
                    ⇒ Enable the popup_menu function of the text editor
                ⇒ End if
                ⇒ If modify clicked
                    ⇒ Write the modified key words to the file
                ⇒ Else if add clicked
                    ⇒ Prompt a text box to get the new keyword
                ⇒ End if
        ⇒ End if
End;
```

**PDL for SNAC analysis**

```
Begin
        ⇒ If text_of_editor edited is TRUE
            ⇒ Call save_file
        ⇒ End if
        ⇒ If identify_already_clicked is TRUE
            ⇒ Do noun_prune with the help of dictionary files
        ⇒ Else
            ⇒ Format input text for tagger to get a .for file
            ⇒ Remove unwanted spaces in the .for to get .rem file
            ⇒ Tag the .rem file to .tag file
            ⇒ Group the noun patterns in the .tag file to get .grp file
            ⇒ Remove unwanted space in the .grp file to get .rem file
            ⇒ Extract the nouns from the .grp file to get .slist file
            ⇒ Parse the .rem fle to get the need patterns in the .need file
            ⇒ List SN components in the list box of the application
            ⇒ Parse the .rem file to get the alterable patterns in the .alt file
                    ⇒ Enable user to select drag and drop the alterable and
                        constraint patterns from the document to the list
                        box of the application
            ⇒ List the AC components in the list box of the application
```

```
                ⇒ Set status message "SNAC components identified"
                ⇒ If mark_relation clicked is TRUE
                        ⇒ Do mark_relations
                ⇒ End if
        ⇒ End if
End;
```

**PDL for mark_relations**

```
Begin
        ⇒ If option selected is s_n
                ⇒ Load .slist file to the combo box
                ⇒ Load .need file to the checklist
        ⇒ Else if option selected is n_a
                ⇒ Load .need file to the combo box
                ⇒ Load .alt file to the checklist
        ⇒ Else if option selected is n_c
                ⇒ Load .need file to the combo box
                ⇒ Load .cons file to the checklist
        ⇒ Else if option selected is c_a
                ⇒ Load .cons file to the combo box
                ⇒ Load .alt file to the checklist
        ⇒ End if
        ⇒ If do_retister clicked
                ⇒ Generate matrix of the relationship.
                                ⇒ Write the component relationships one by one in a
                                        file for the user to frame objectives
                                ⇒ Show opened and created files.
                        ⇒ End if
End;
```

**PDL for CID menu options**

```
Begin
        ⇒ If option selected is new_doc
                ⇒ Open a new application window.
        ⇒ Else if option selected is open_file
                ⇒ Open the .cid file to the graphical editor of the application
        ⇒ Else if option selected is open_list
                ⇒ Open the key elements list file to the combo of the application
        ⇒ Else if option selected is save_file
                ⇒ Save the CID to a .cid file
        ⇒ Else if option selected is save As_file
```

```
            ⇒ Save the CID to a .cid file with a new name
    ⇒ Else if option selected is print_file
            ⇒ Print the document in the text editor
    ⇒ Else if option selected is quit
            ⇒ Close the application
    ⇒ Else if option selected is what's_this
            ⇒ Enable the what is this function
    ⇒ Else if option selected is draw_mode
            ⇒ Enable the draw functions in the editor
    ⇒ Else if option selected is lock_mode
            ⇒ Disable all the draw functions of the editor
    ⇒ Else if option selected is draw_factor
            ⇒ Draw a factor at the position where the mouse is clicked
    ⇒ Else if option selected is draw_arrow
            ⇒ Draw an arrow between the from and to factor selected
    ⇒ Else if option selected is delete_factor
            ⇒ Delete the factor which is selected
    ⇒ Else if option selected is delete_arrow
            ⇒ Delete the arrow connecting the from factor and the to factor
    ⇒ Else if option selected is mege_analysis
            ⇒ Do merge-point analysis and highlight the merge point in the graph
    ⇒ Else if option selected is burst_analysis
            ⇒ Do burst-point analysis and highlight the burst point in the graph
    ⇒ Else if option selected is loop_analysis
                    ⇒ Perform the loop analysis of the graph and show the
                        different loops existing in the diagram and their
                        resultant polarity
    ⇒ Else if option selected is about
            ⇒ Show the about dialog box
    ⇒ End if

End;
```

**PDL for the draw factor function**

```
Begin
    ⇒ If draw_mode is TRUE
                    ⇒ Initialize all its variables, no_in_arrows,
                        no_outarrows, x_y_position, text_of factor,
                        box_width, box_height, factor_id
            ⇒ Click on the editor window where the factor is to be drawn
            ⇒ Set the driver as its parent
```

```
                        ⇒ Registers this factor_name, its (x, y) position,
                            factor_id on to a driver which controls all the
                            activities
    ⇒ End if
End;
```

**PDL for draw arrow function**

```
Begin
    ⇒ If draw_mode is TRUE
        ⇒ Click on the form_factor and to_factor of the arrow
        ⇒ Set the polarity of the arrow as –ve, +ve or zero
        ⇒ Initialize all the arrow variables, arrow_id, from_factor, to_factor
        ⇒ Increment the no_of_arrow in the from_factor and to_factor
                ⇒ Find the middle points of the two factors to which
                    the arrow is connected
        ⇒ Register these details to the driver
    ⇒ End if
End;
```

# SOFTWARE TESTING

# 7. SOFTWARE TESTING

The system should be verified and validated at each stage of development. The verification involves checking that the program confirms to its specification, whereas validation involves checking the program as implemented meets the users expectations. The stages involved in testing are

- Unit Testing
- Module Testing
- Sub-system Testing
- System Testing

The testing process can be listed as

i. Requirement traceability – all the requirements are individually tested.

ii. Testing schedule

iii. Test recording procedure – the results of the test are recorded.

iv. Hardware and software requirements for testing

## 7.1 Unit Testing

The individual components are tested to ensure that they operate correctly. Some of the units in the system are taken into consideration and their test cases are described.

| Unit being tested | Test cases | Test Outputs |
|---|---|---|
| a. Formatting the input text for the tagger | i. Check whether more than one space is there between tokens | Single space between tokens |
|  | ii. All punctuation marks should be considered | Punctuation marks should be separated by a single space. |
|  | iii. Check whether program output miss any of the input tokens (if rules don't match this may happen) | Rules should cover all types of inputs |
|  | iv. Check the program output for different input files having varying patterns of text. | Program should support all patterns of input text |
| Grouping of tagged nouns | The groups are formed by<br>i. one or more nouns occurring consecutively<br>ii. patterns with the preposition "of" between nouns<br>iii. patterns with nouns preceded by one or two adjectives | All the noun groups coming under these category should be enclosed in <GP>....</GP> tags. |
| parser which identifies key words | check whether almost all verb and noun phrases are included | Can miss some of the verb phrases, but the entire noun p[phrases should be included |

| | | |
|---|---|---|
| Program which eliminates the duplicate elements from the list of stakeholders | Words can occur in plurals and singular form, only any one of the form should be preserved.<br><br>Words can occur in lower and upper cases, only one of the form is written.<br><br>Only noun groups with word whose starting letter is capital and has one or more "of" preposition should be taken.<br><br>For eg. *State Bank f India*<br><br>while *settlement of claims* should not be taken | List of stakeholders which don't contain duplicates |
| Parser which identifies need patters | Check if all the expected patterns are found. The parser takes the longest token if two grammars are matched. So it can miss some patterns | All the *need patterns* are needed |
| Generate matrix of the relationship between the SNAC components | Check the matrix do not store garbage values | Stores 1 if there is a relation and 0 if no relation |
| Merge point analysis | Checks if the merge points satisfies the threshold specified | Different type of complexity diagrams are drawn and there merge points are found to check the correctness of the algorithm |
| Loop analysis | Checks if all the loops in the diagram are found<br><br>Duplicates loops should not be allowed | The polarity of the loops, and the total number of loops in the diagram and the path of the loops are listed |
| Burst point analysis | Burst point also have a threshold value that should not be exceeded | The points which have out arrows less than the threshold value should be listed |

**Table 2**

## 7.2 Module Testing

Module interface is tested to ensure that information properly flows into and out of program unit under test. The local data structure is examined to ensure the data is stored temporarily maintains integrity during all steps. Boundary conditions are also tested. The unit tested modules are combined together to form a module and then the test cases for the modules are designed. Modules comprises the classes which come in the system. Using the debugger GNU gdb the classes can be executed one by one and the errors can be traced and corrected. The independent paths through the control structure are exercised to ensure that all the statements in the module are executed at least once. Finally all error handling paths are tested.

## 7.3 Structural testing

It is a white-box testing method. In white-box testing, the tester can analyze the code and use knowledge about the structure of a component to derive test data. Here an analysis of the code can be used to find how many test cases are needed to guarantee a given level of test coverage. For a module to be white box tested, a testing strategy called basis path testing an be used, whose objective is to exercise every independent path through the component. If every independent path is executed, then all statements in the program must have been executed at least once. All conditional statements are tested for both true and false values. The starting point for path testing is a program flow graph. A sample code PDL of one of the units in the system is given below.

PDL of the program that eliminates duplicates in the stakeholders list.

```
Begin
        ⇒ 1. while stakeholders list end
            ⇒ 2. read one record
            ⇒ 3. if reading the list for the first time
                    ⇒ 4. write to the output file
                    ⇒ 5. else
                            ⇒ 6. search record for apostrophe symbol
                            ⇒ 7. if there is apostrophe
                                    ⇒ 8. set flag
            ⇒ 9. else
                    ⇒ 10. search the record for the word "of" in it
                    ⇒ 11. if there is "of"
                          ⇒ 12. count the words starting with lower
                          case and set flag
                    ⇒ 13. else
                          ⇒ 14. search record is present in the output
                                file and set flag
                          ⇒ 15. if not found in the output file
                                ⇒ 16. remove "s" from the record if
                                "s" is there and search for new record
                                ⇒ 17. set flag
                                ⇒ 18. if not found
                                ⇒ 19. remove "ies" from the record
                                    if "ies" is there and search for
                                    new record
                                ⇒ 20. set flag
                                ⇒ 21. if not found
                                ⇒ 22. change the upper case words
                            to lower and search and also vice versa
                                ⇒ 23. set flag
                          ⇒ 24. end if
                    ⇒ 25. end if
            ⇒ 26. end if
            ⇒ 27. if flag <2
                    ⇒ 28. write the record to the output file
            ⇒ 29. end if
      ⇒ 30. end if
      ⇒ 31. end while
⇒ 32. End;
```

# CONCLUSION

# 8. CONCLUSION

The system Engineering and Cybernetics Center (SECC), Pune, a division of TCS developed a multi-modeling framework as against traditional problem solving approach for consultancy process. This multi-modeling technique makes use of several process and techniques in performing the diagnosis and design phase of problem solving. Integrating these techniques in one framework creates a new vision called **Concept to Code**. The vision aims at producing the hard code equivalent to consultancy problem.

The software Analyst Aid is an attempt to create the initial phase of the **Concept to Code** realization project. It assists the analyst in initial problem discovery and diagnosis phases. The software can be further enhanced in the following way to proceed with the realization of **Concept to Code** project. The information architecture derived from this software can be transformed to UML notations. These notations, if interfaced with TCS like ADEX or Master Craft can produce the code equivalent to it.

# SCOPE FOR FUTURE DEVELOPMENT

# 9. SCOPE FOR FUTURE DEVELOPMENT

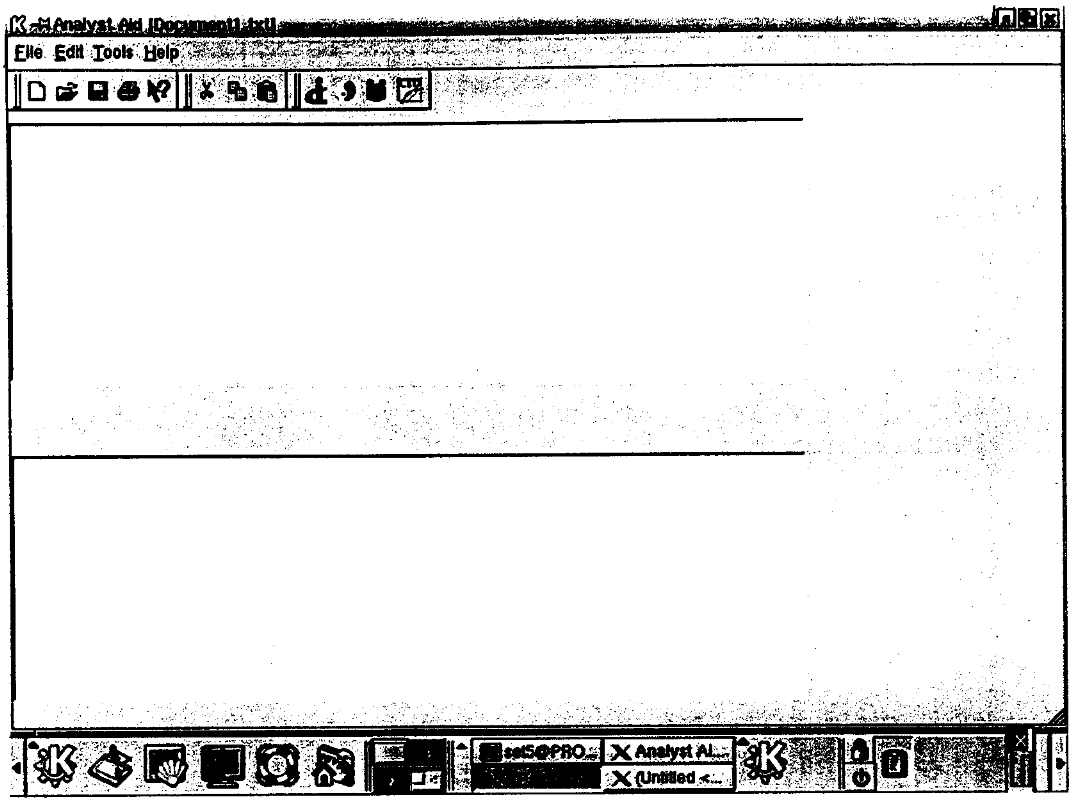
*Analyst Aid*, a software developed at TATA Consultancy Services, Technopark, Thiruvananthapuram, is a system which assists the requirement understanding phase in the software development life cycle. The system helps the user in determining the key factors in the consultancy problem and their relationships and thereby structuring the problem. We can do more things to the system to develop a perfect tool for an analyst. The following are the some of the future development of this project.

- It can be used to validate the CID module.
- Develop self - interaction matrix rather than cross interaction matrix.

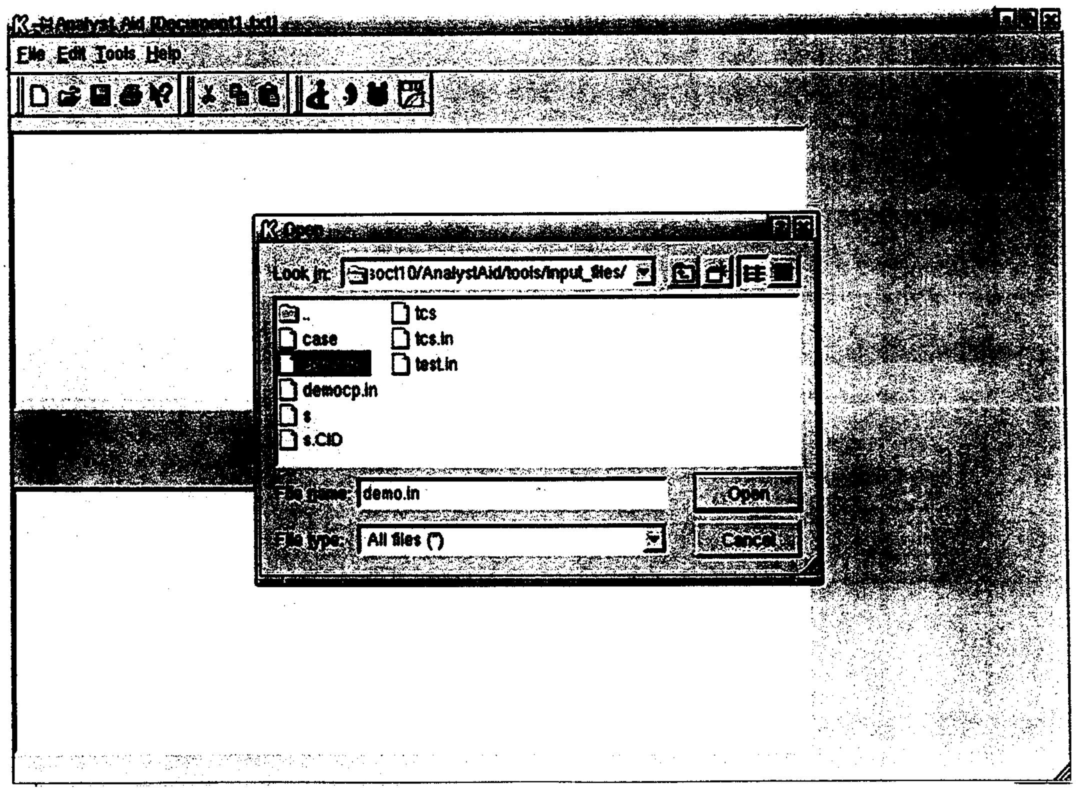
  The system will automatically generate the self interaction matrix using SNAC components and it will extract all the objectives of the system.
- Integrating with CID module in more effective way.
- We can add more rules to identify all stakeholder, need, alterables and constraints.
- Extract the overall object from the interaction matrix .

# APPENDIX

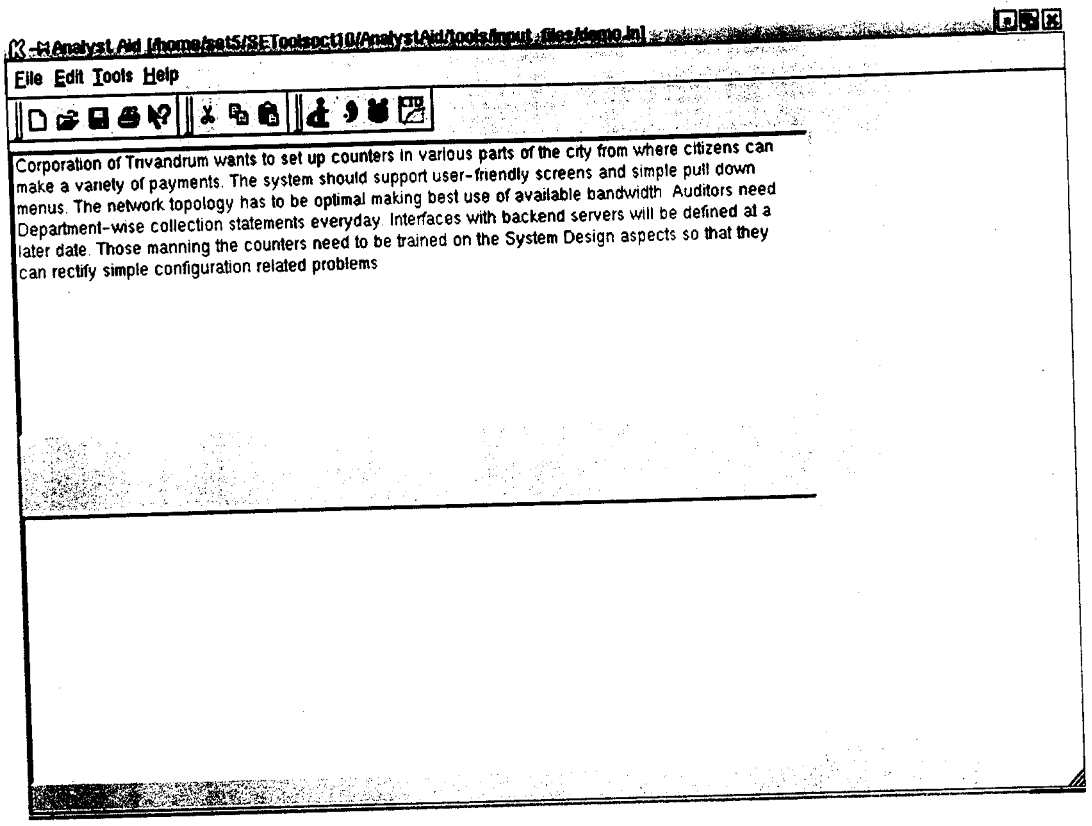
# SCREEN SHOTS



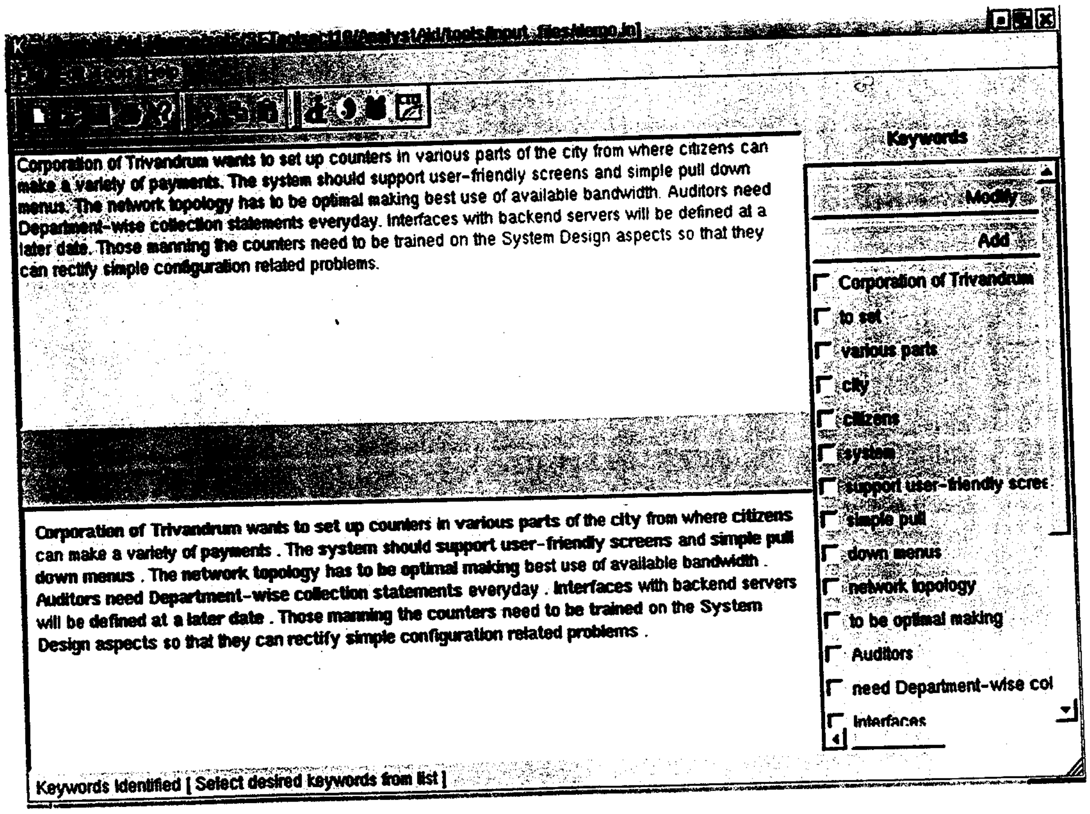
Analyst Aid [Document1.txt]
File Edit Tools Help
sst5@PRO...
Analyst Ai...
(Untitled <...



Analyst Aid [Document1.txt]
File Edit Tools Help
Open
Look in: 3oct10/AnalystAid/tools/input_files/
..
case
democp.in
s
s.CID
tcs
tcs.in
test.in
File name: demo.in
File type: All files (*)
Open
Cancel


Analyst Aid [/home/set5/SEToolsoct10/AnalystAid/tools/input_files/demo.in]
File Edit Tools Help

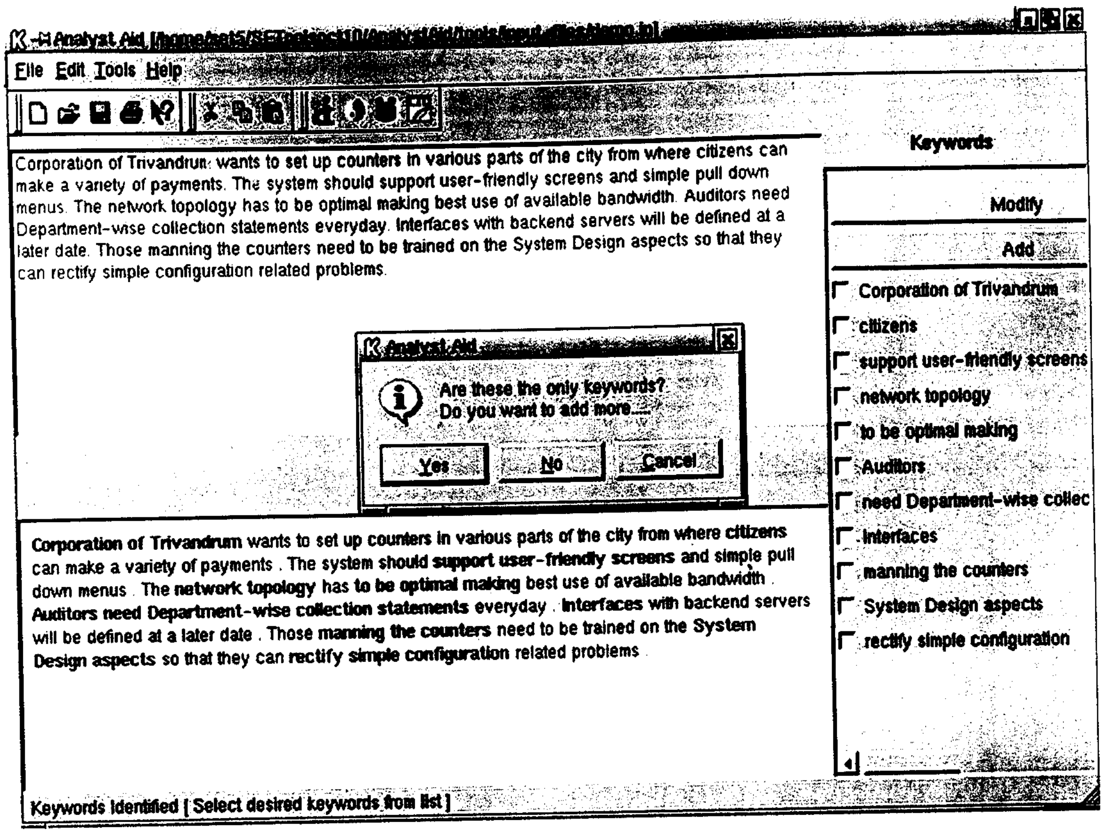
Corporation of Trivandrum wants to set up counters in various parts of the city from where citizens can make a variety of payments. The system should support user-friendly screens and simple pull down menus. The network topology has to be optimal making best use of available bandwidth. Auditors need Department-wise collection statements everyday. Interfaces with backend servers will be defined at a later date. Those manning the counters need to be trained on the System Design aspects so that they can rectify simple configuration related problems



Corporation of Trivandrum wants to set up counters in various parts of the city from where citizens can make a variety of payments. The system should support user-friendly screens and simple pull down menus. The network topology has to be optimal making best use of available bandwidth. Auditors need Department-wise collection statements everyday. Interfaces with backend servers will be defined at a later date. Those manning the counters need to be trained on the System Design aspects so that they can rectify simple configuration related problems.

Corporation of Trivandrum wants to set up counters in various parts of the city from where citizens can make a variety of payments . The system should support user-friendly screens and simple pull down menus . The network topology has to be optimal making best use of available bandwidth . Auditors need Department-wise collection statements everyday . Interfaces with backend servers will be defined at a later date . Those manning the counters need to be trained on the System Design aspects so that they can rectify simple configuration related problems .

Keywords Identified [ Select desired keywords from list ]

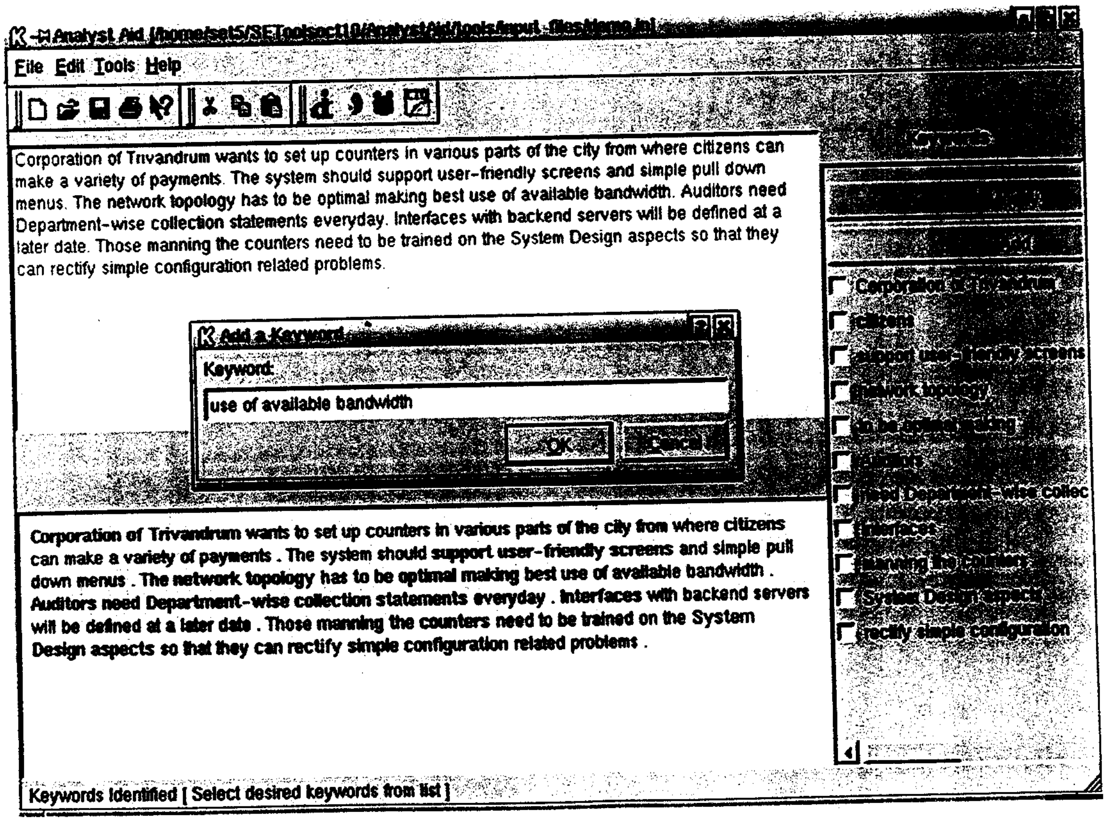


Corporation of Trivandrum wants to set up counters in various parts of the city from where citizens can make a variety of payments. The system should support user-friendly screens and simple pull down menus. The network topology has to be optimal making best use of available bandwidth. Auditors need Department-wise collection statements everyday. Interfaces with backend servers will be defined at a later date. Those manning the counters need to be trained on the System Design aspects so that they can rectify simple configuration related problems.

**Corporation of Trivandrum** wants to set up **counters in various parts of the city from where citizens** can make a variety of payments . The system **should support user-friendly screens** and simple pull down menus . The **network topology** has **to be optimal making** best use of available bandwidth . **Auditors need Department-wise collection statements** everyday . **Interfaces with** backend servers will be defined at a later date . Those **manning the counters** need to be trained on the **System Design aspects** so that they can **rectify simple configuration** related problems .

- Corporation of Trivandrum
- citizens
- support user-friendly screens
- network topology
- to be optimal making
- Auditors
- need Department-wise collec
- Interfaces
- manning the counters
- System Design aspects
- rectify simple configuration

Keywords Identified [ Select desired keywords from list ]



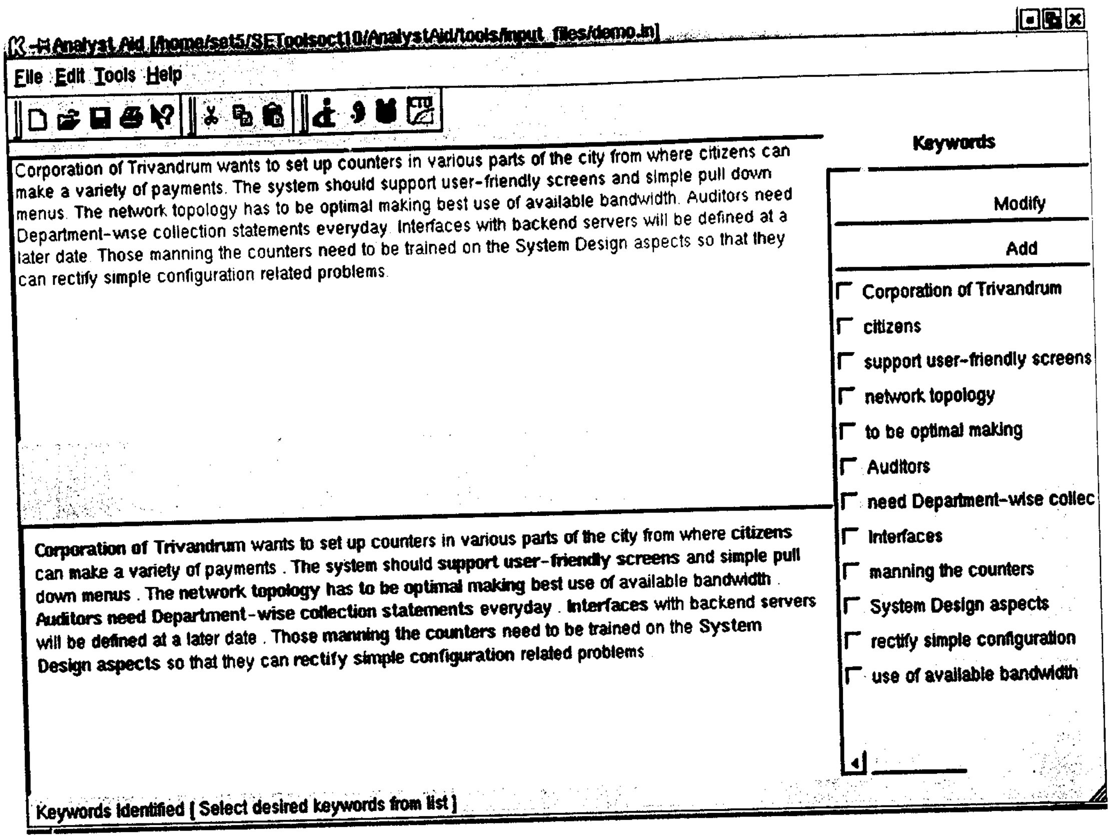
Corporation of Trivandrum wants to set up counters in various parts of the city from where citizens can make a variety of payments. The system should support user-friendly screens and simple pull down menus. The network topology has to be optimal making best use of available bandwidth. Auditors need Department-wise collection statements everyday. Interfaces with backend servers will be defined at a later date. Those manning the counters need to be trained on the System Design aspects so that they can rectify simple configuration related problems.

Keyword: use of available bandwidth

**Corporation of Trivandrum wants to set up** counters **in various parts of the city from where citizens** can make **a variety of payments . The system should support user-friendly screens** and simple pull down **menus . The network topology has to be optimal making best use of available** bandwidth . **Auditors need Department-wise collection statements everyday . Interfaces with** backend servers **will be defined at a later date . Those manning the counters need to be trained on the System Design aspects so that they can rectify simple configuration related problems** .

- Corporation of Trivandrum
- citizens
- support user-friendly screens
- network topology
- to be optimal making
- Auditors
- need Department-wise collec
- Interfaces
- manning the counters
- System Design aspects
- rectify simple configuration

Keywords Identified [ Select desired keywords from list ]

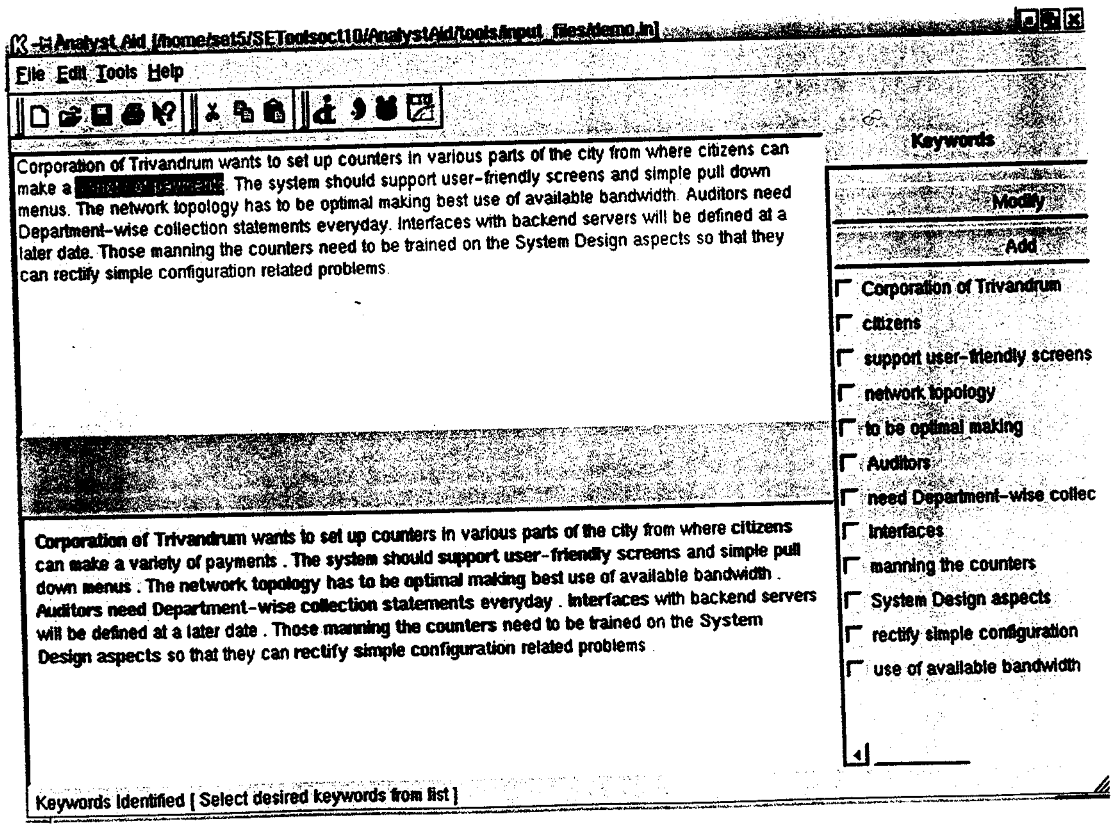

Analyst Aid [/home/set5/SEToolsoct10/AnalystAid/tools/input_files/demo.in]
File Edit Tools Help
Keywords
Modify
Add

Corporation of Trivandrum wants to set up counters in various parts of the city from where citizens can make a variety of payments. The system should support user-friendly screens and simple pull down menus. The network topology has to be optimal making best use of available bandwidth. Auditors need Department-wise collection statements everyday. Interfaces with backend servers will be defined at a later date. Those manning the counters need to be trained on the System Design aspects so that they can rectify simple configuration related problems.

**Corporation of Trivandrum** wants to set up counters in various parts of the city from where **citizens** can make a variety of payments . The system should **support user-friendly screens** and simple pull down menus . The **network topology** has **to be optimal making** best **use of available bandwidth** . **Auditors need Department-wise collection statements** everyday . **Interfaces** with backend servers will be defined at a later date . Those **manning the counters** need to be trained on the **System Design aspects** so that they can **rectify simple configuration** related problems .

- ☐ Corporation of Trivandrum
- ☐ citizens
- ☐ support user-friendly screens
- ☐ network topology
- ☐ to be optimal making
- ☐ Auditors
- ☐ need Department-wise collec
- ☐ Interfaces
- ☐ manning the counters
- ☐ System Design aspects
- ☐ rectify simple configuration
- ☐ use of available bandwidth

Keywords Identified [ Select desired keywords from list ]

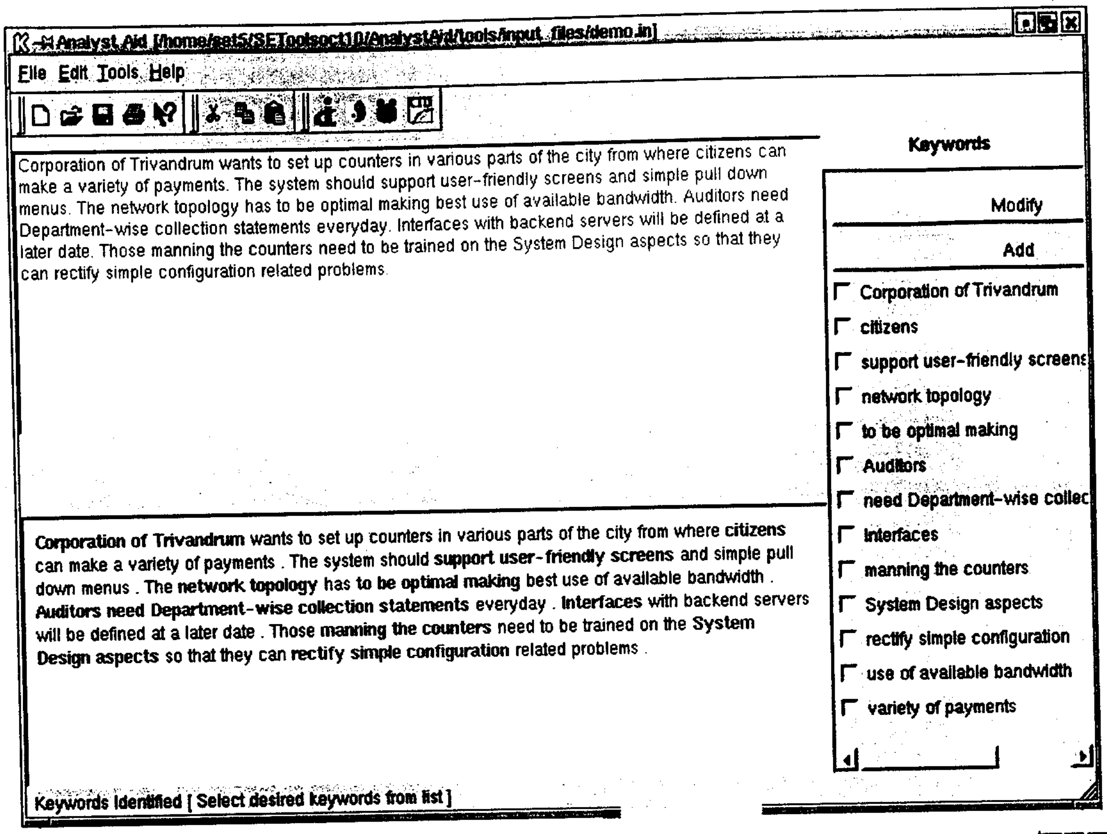

Analyst Aid [/home/set5/SEToolsoct10/AnalystAid/tools/input_files/demo.in]
File Edit Tools Help
Keywords
Modify
Add

Corporation of Trivandrum wants to set up counters in various parts of the city from where citizens can make a variety of payments. The system should support user-friendly screens and simple pull down menus. The network topology has to be optimal making best use of available bandwidth. Auditors need Department-wise collection statements everyday. Interfaces with backend servers will be defined at a later date. Those manning the counters need to be trained on the System Design aspects so that they can rectify simple configuration related problems.

**Corporation of Trivandrum** wants to set up counters in various parts of the city from where **citizens** can make a variety of payments . The system should **support user-friendly screens** and simple pull down menus . The **network topology** has **to be optimal making** best **use of available bandwidth** . **Auditors need Department-wise collection statements** everyday . **Interfaces** with backend servers will be defined at a later date . Those **manning the counters** need to be trained on the **System Design aspects** so that they can **rectify simple configuration** related problems .

- ☐ Corporation of Trivandrum
- ☐ citizens
- ☐ support user-friendly screens
- ☐ network topology
- ☐ to be optimal making
- ☐ Auditors
- ☐ need Department-wise collec
- ☐ Interfaces
- ☐ manning the counters
- ☐ System Design aspects
- ☐ rectify simple configuration
- ☐ use of available bandwidth

Keywords Identified [ Select desired keywords from list ]

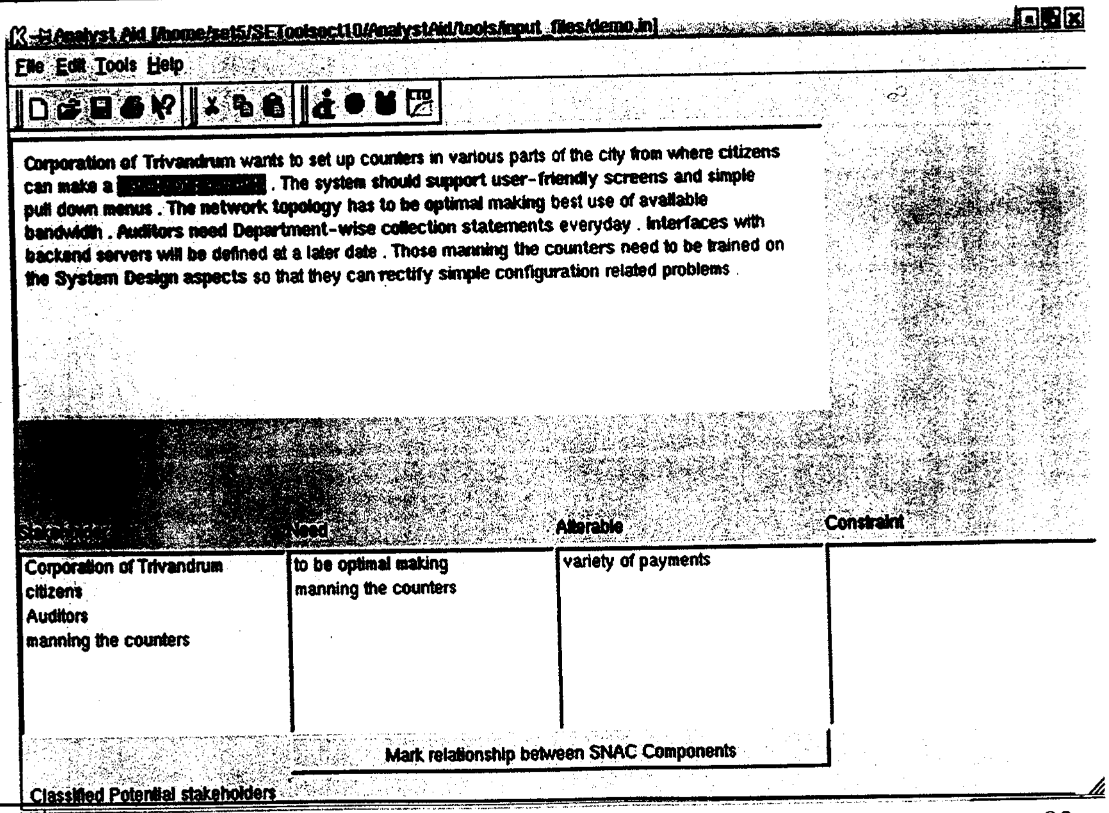

Analyst Aid [/home/set5/SEToolsoct10/AnalystAid/tools/input_files/demo.in]
File Edit Tools Help
Keywords
Modify
Add

Corporation of Trivandrum wants to set up counters in various parts of the city from where citizens can make a variety of payments. The system should support user-friendly screens and simple pull down menus. The network topology has to be optimal making best use of available bandwidth. Auditors need Department-wise collection statements everyday. Interfaces with backend servers will be defined at a later date. Those manning the counters need to be trained on the System Design aspects so that they can rectify simple configuration related problems.

**Corporation of Trivandrum** wants to set up counters in various parts of the city from where citizens can make a variety of payments . The system should **support user-friendly screens** and simple pull down menus . The **network topology** has **to be optimal making** best use of available bandwidth . **Auditors need Department-wise collection statements** everyday . **Interfaces** with backend servers will be defined at a later date . Those **manning the counters** need to be trained on the **System Design aspects** so that they can **rectify simple configuration** related problems .

- Corporation of Trivandrum
- citizens
- support user-friendly screens
- network topology
- to be optimal making
- Auditors
- need Department-wise collec
- Interfaces
- manning the counters
- System Design aspects
- rectify simple configuration
- use of available bandwidth
- variety of payments

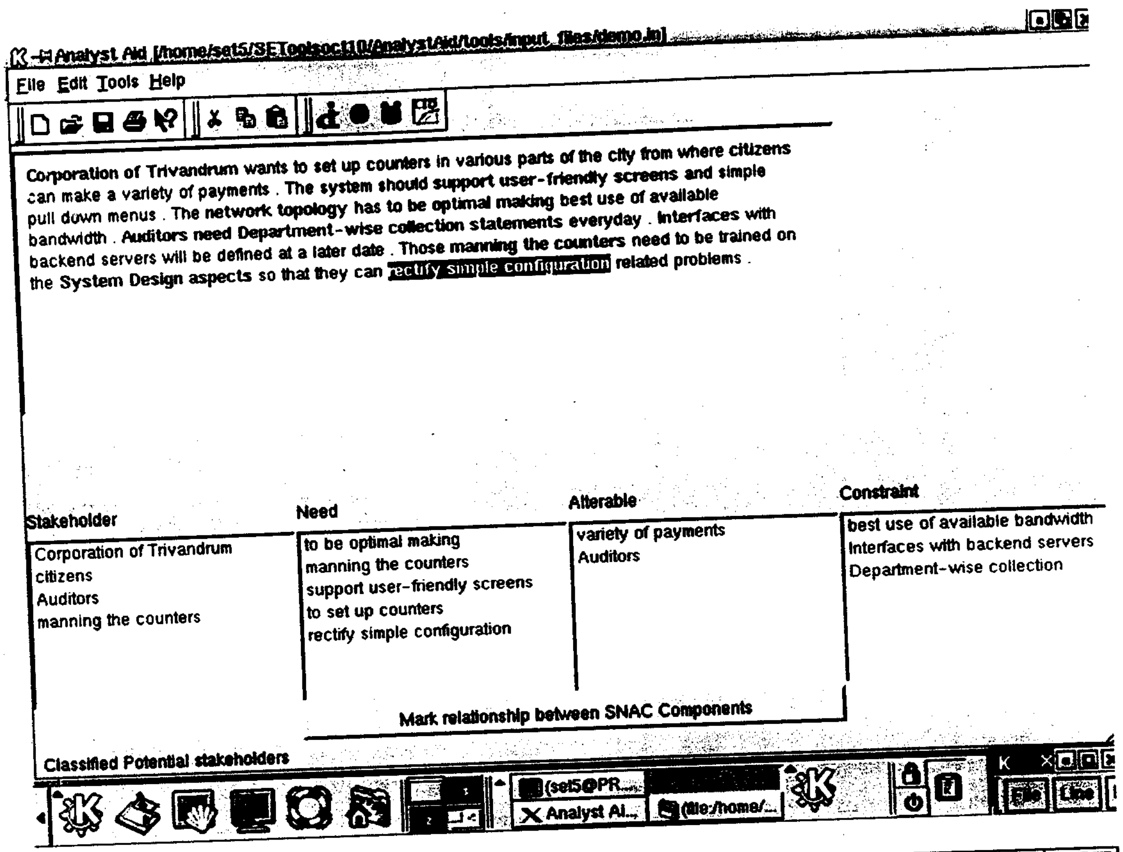
Keywords Identified [ Select desired keywords from list ]


Analyst Aid [/home/set5/SEToolsoct10/AnalystAid/tools/input_files/demo.in]
File Edit Tools Help
Mark relationship between SNAC Components

**Corporation of Trivandrum wants to set up counters in various parts of the city from where citizens can make a** [illegible] **. The system should support user-friendly screens and simple pull down menus . The network topology has to be optimal making best use of available bandwidth . Auditors need Department-wise collection statements everyday . Interfaces with backend servers will be defined at a later date .** Those **manning** the counters need to be trained on **the System Design aspects** so that they can rectify **simple** configuration related problems .

| Stakeholder | Need | Alterable | Constraint |
|---|---|---|---|
| Corporation of Trivandrum<br>citizens<br>Auditors<br>manning the counters | to be optimal making<br>manning the counters | variety of payments | |

Classified Potential stakeholders


Analyst Aid [/home/set5/SEToolsoct10/AnalystAid/tools/input_files/demo.in]

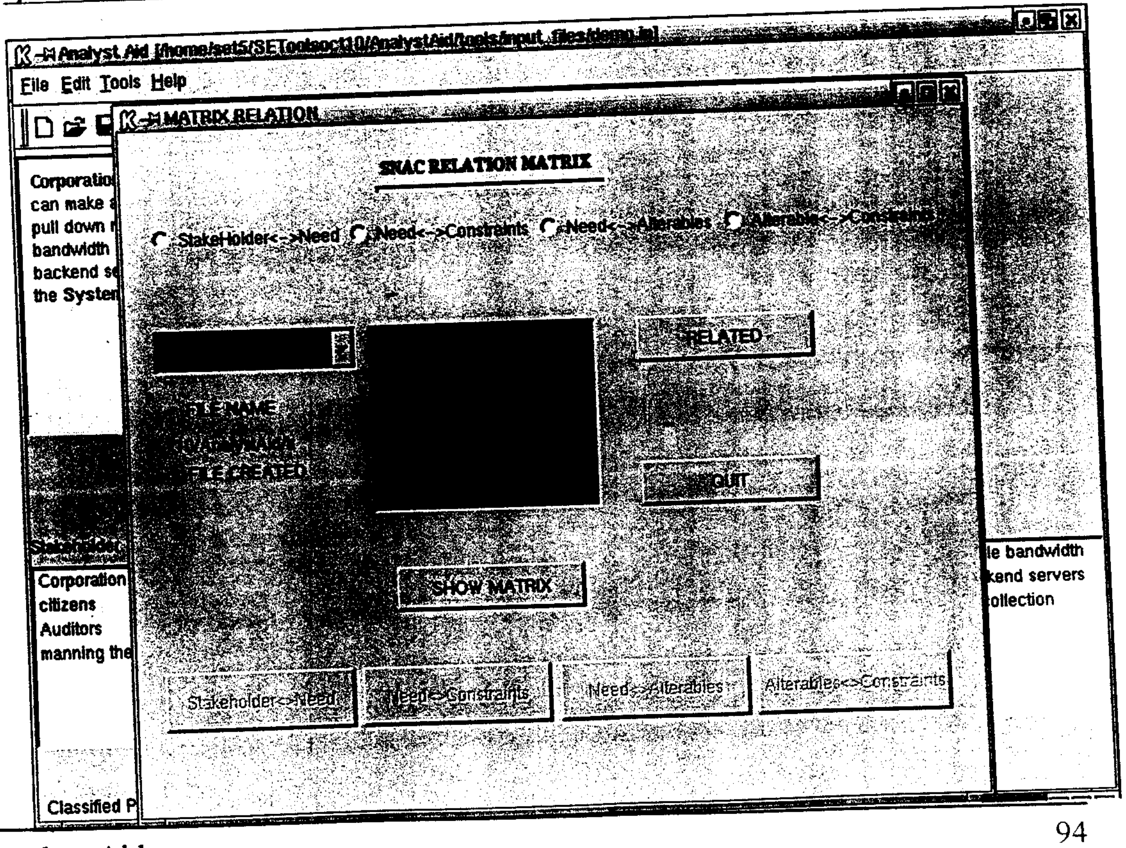
File Edit Tools Help

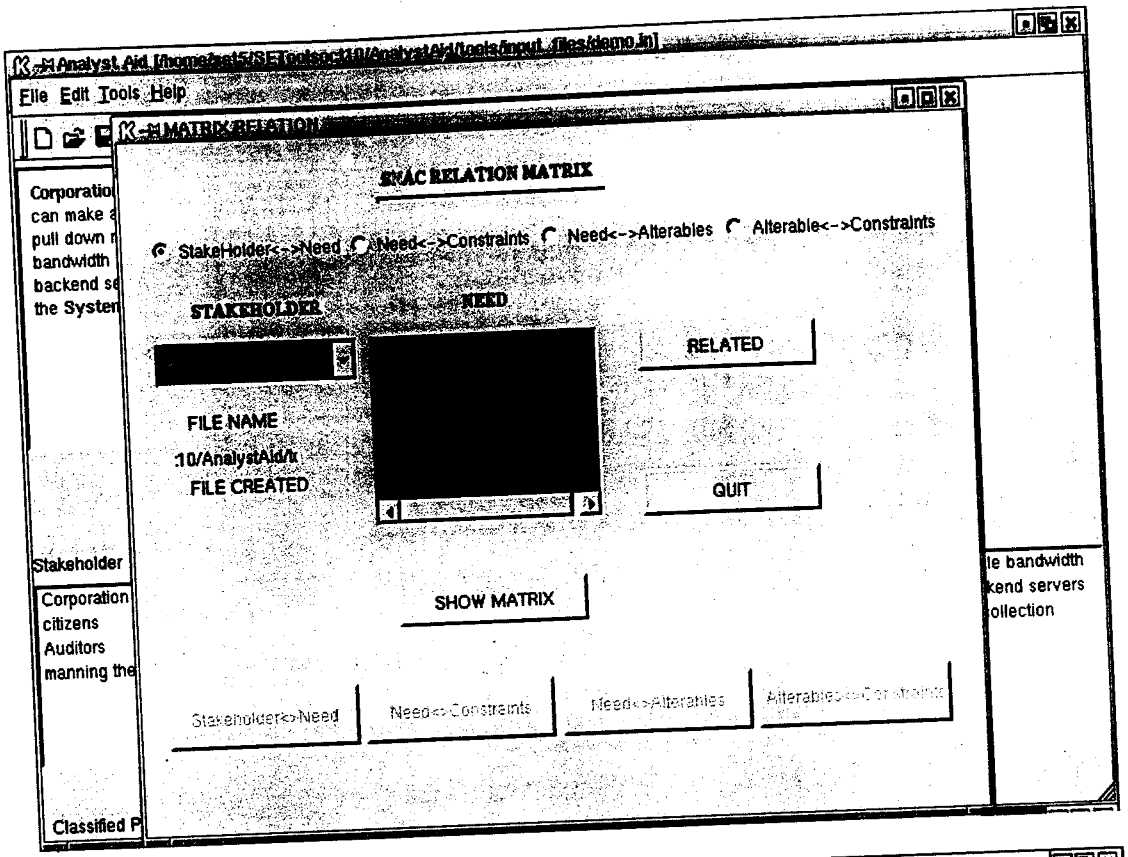
Corporation of Trivandrum wants to set up counters in various parts of the city from where citizens can make a variety of payments . The system should support user-friendly screens and simple pull down menus . The network topology has to be optimal making best use of available bandwidth . Auditors need Department-wise collection statements everyday . Interfaces with backend servers will be defined at a later date . Those manning the counters need to be trained on the System Design aspects so that they can rectify simple configuration related problems .

| Stakeholder | Need | Alterable | Constraint |
|---|---|---|---|
| Corporation of Trivandrum | to be optimal making | variety of payments | best use of available bandwidth |
| citizens | manning the counters | Auditors | Interfaces with backend servers |
| Auditors | support user-friendly screens | | Department-wise collection |
| manning the counters | to set up counters | | |
| | rectify simple configuration | | |

Mark relationship between SNAC Components

Classified Potential stakeholders





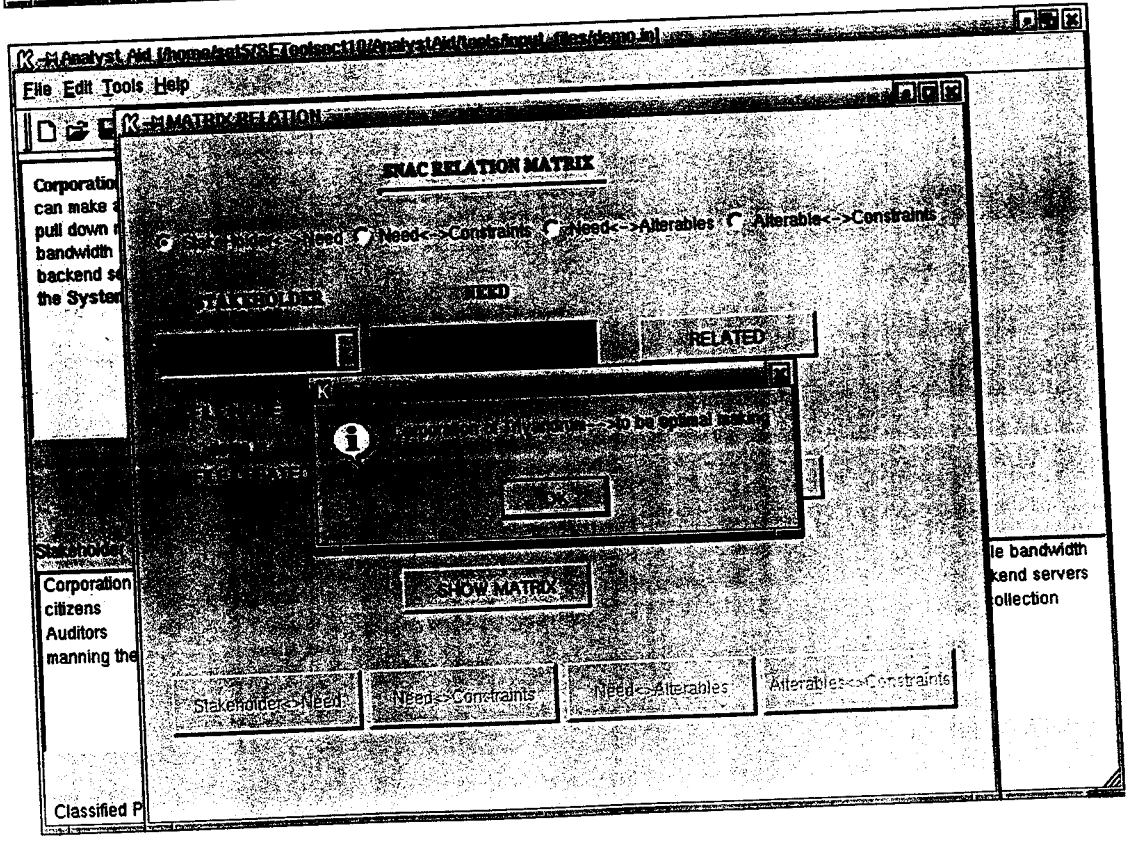
[✓ StakeHolder<-->Need] Need<-->Constraints Need<-->Alterables Alterable<-->Constraints

FILE NAME
:10/AnalystAid/t

Corporatio
can make
pull down
bandwidth
backend s
the Systen

Corporation
citizens
Auditors
manning the

le bandwidth
kend servers
ollection

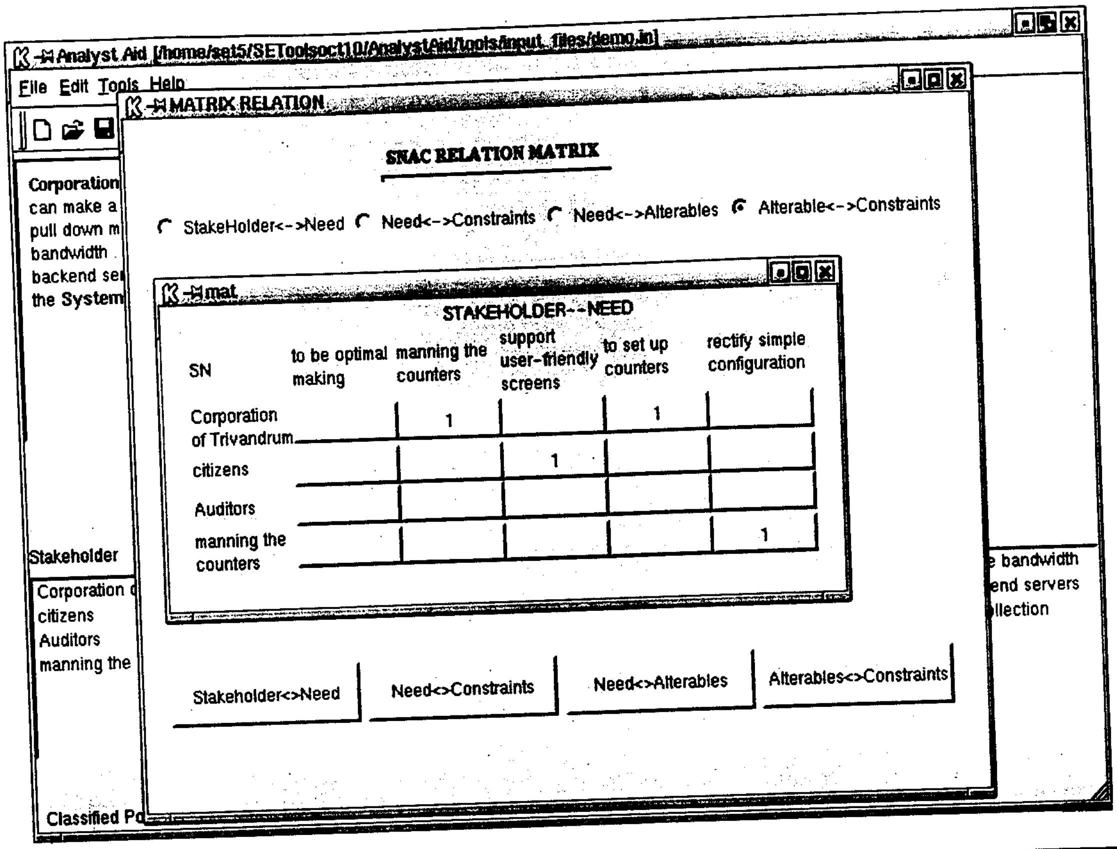


[illegible]-->to be optimal [illegible]

Corporatio
can make
pull down
bandwidth
backend s
the Systen

Corporation
citizens
Auditors
manning the

le bandwidth
kend servers
ollection


Analyst Aid [/home/set5/SEToolsoct10/AnalystAid/tools/input_files/demo.in]

File Edit Tools Help

MATRIX RELATION

**SNAC RELATION MATRIX**

( ) StakeHolder<->Need ( ) Need<->Constraints ( ) Need<->Alterables [✓ Alterable<->Constraints]

mat

STAKEHOLDER--NEED

| SN | to be optimal making | manning the counters | support user-friendly screens | to set up counters | rectify simple configuration |
|---|---|---|---|---|---|
| Corporation of Trivandrum |  | 1 |  | 1 |  |
| citizens |  |  | 1 |  |  |
| Auditors |  |  |  |  |  |
| manning the counters |  |  |  |  | 1 |

Stakeholder<>Need | Need<>Constraints | Need<>Alterables | Alterables<>Constraints

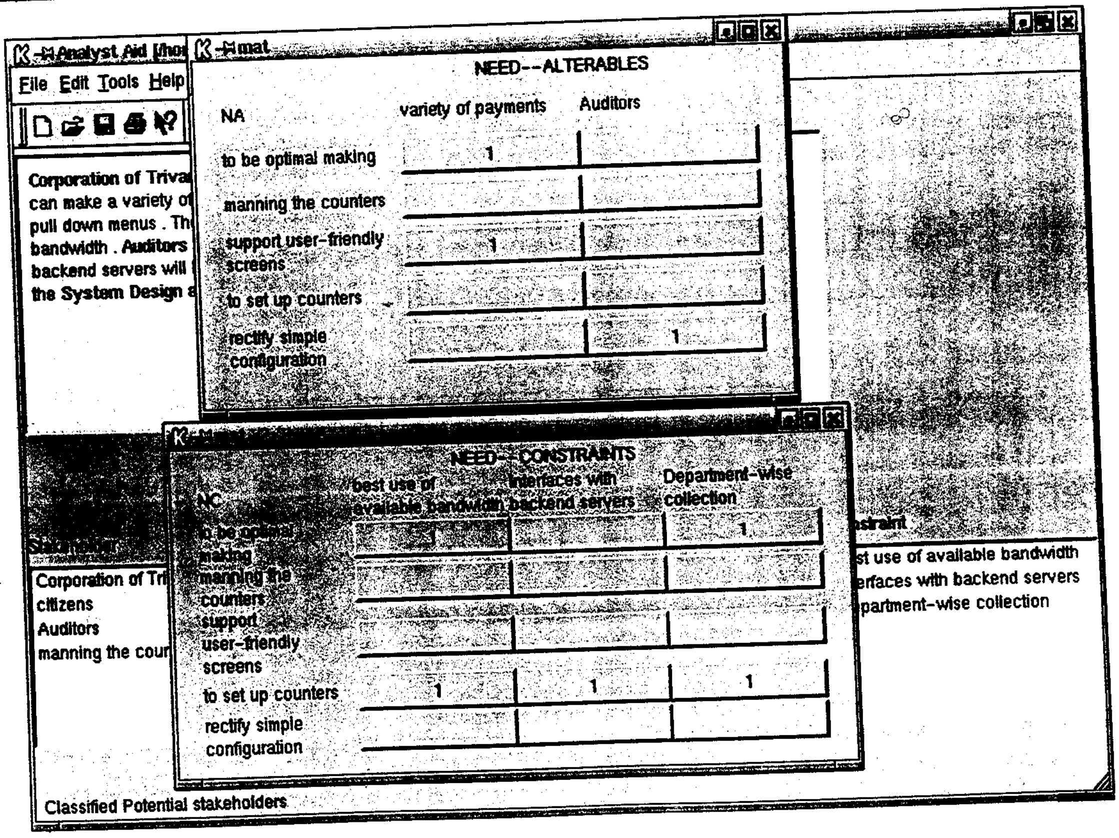
Corporation
can make a
pull down m
bandwidth .
backend se
the System

Stakeholder

Corporation
citizens
Auditors
manning the

Classified Po

bandwidth
end servers
llection


Analyst Aid [/hor

File Edit Tools Help

Corporation of Triva
can make a variety o
pull down menus . Th
bandwidth . Auditors
backend servers will
the System Design a

mat

NEED--ALTERABLES

| NA | variety of payments | Auditors |
|---|---|---|
| to be optimal making | 1 |  |
| manning the counters |  |  |
| support user-friendly screens | 1 |  |
| to set up counters |  |  |
| rectify simple configuration |  | 1 |

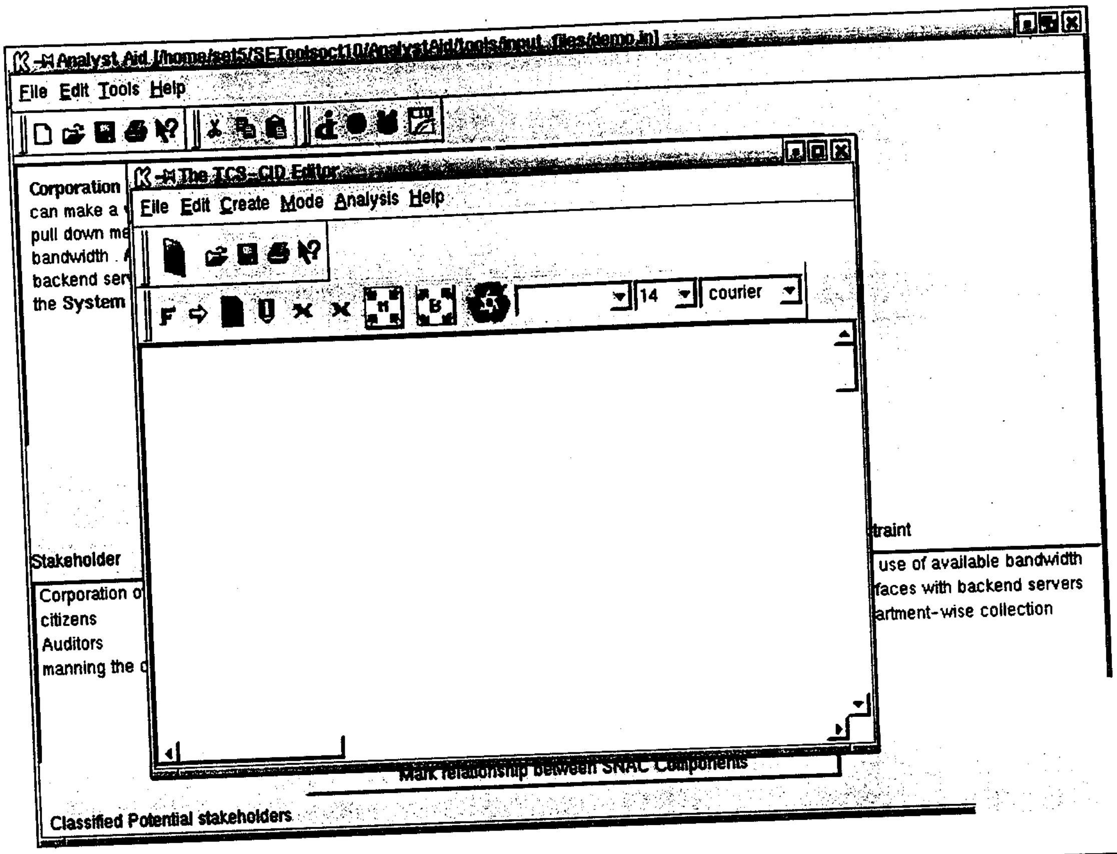
NEED--CONSTRAINTS

| NC | best use of available bandwidth | interfaces with backend servers | Department-wise collection |
|---|---|---|---|
| to be optimal making | 1 |  | 1 |
| manning the counters |  |  |  |
| support user-friendly screens |  |  |  |
| to set up counters | 1 | 1 | 1 |
| rectify simple configuration |  |  |  |

Corporation of Tri
citizens
Auditors
manning the cou

straint
st use of available bandwidth
erfaces with backend servers
partment-wise collection

Classified Potential stakeholders


Analyst Aid /home/set5/SEToolsoct10/AnalystAid/tools/input_files/demo.in
File Edit Tools Help
The TCS-CID Editor
File Edit Create Mode Analysis Help
14
courier
Mark relationship between SNAC Components

Corporation
can make a
pull down me
bandwidth .
backend ser
the System

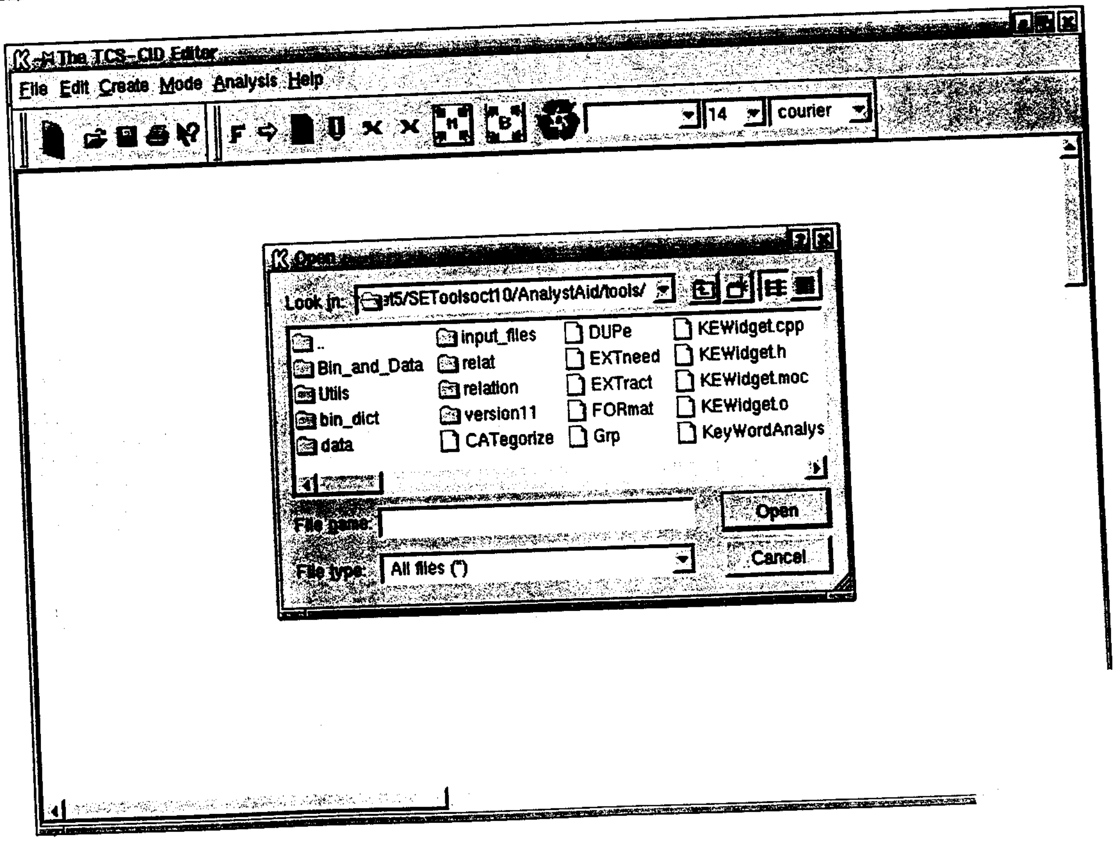
Stakeholder

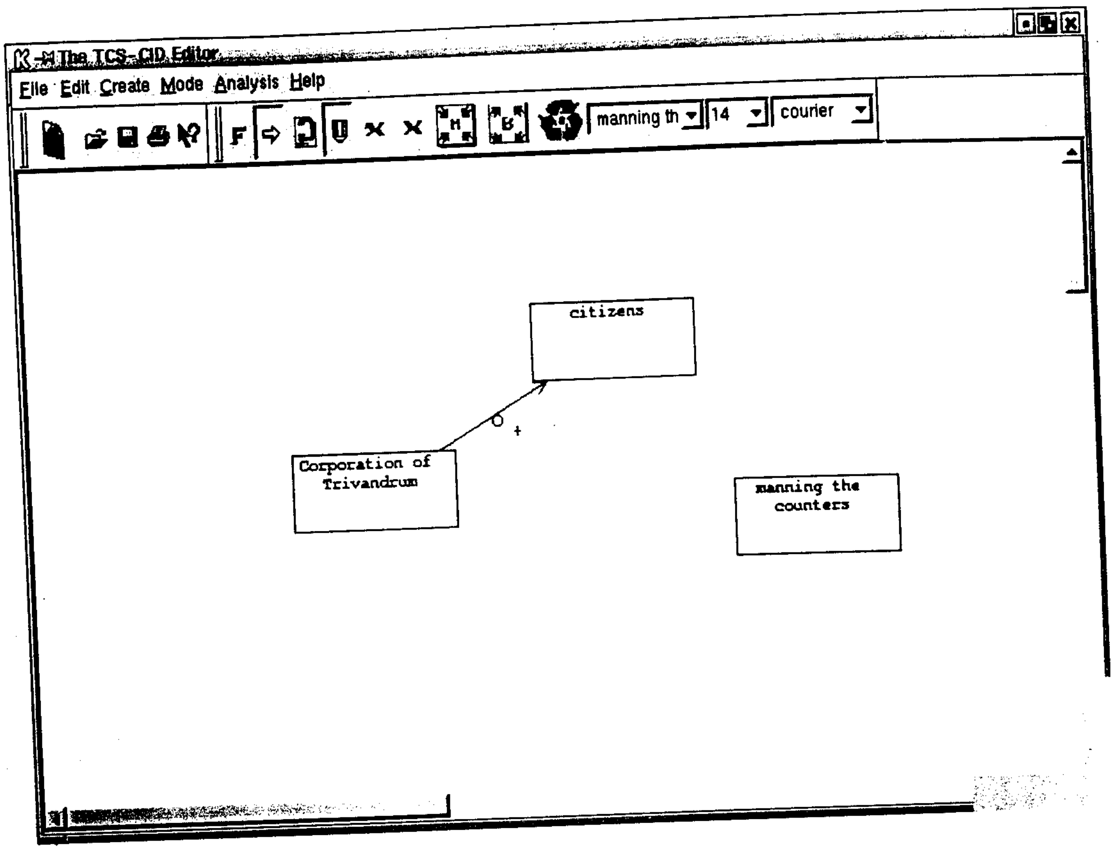
Corporation o
citizens
Auditors
manning the c

traint

use of available bandwidth
faces with backend servers
artment-wise collection

Classified Potential stakeholders

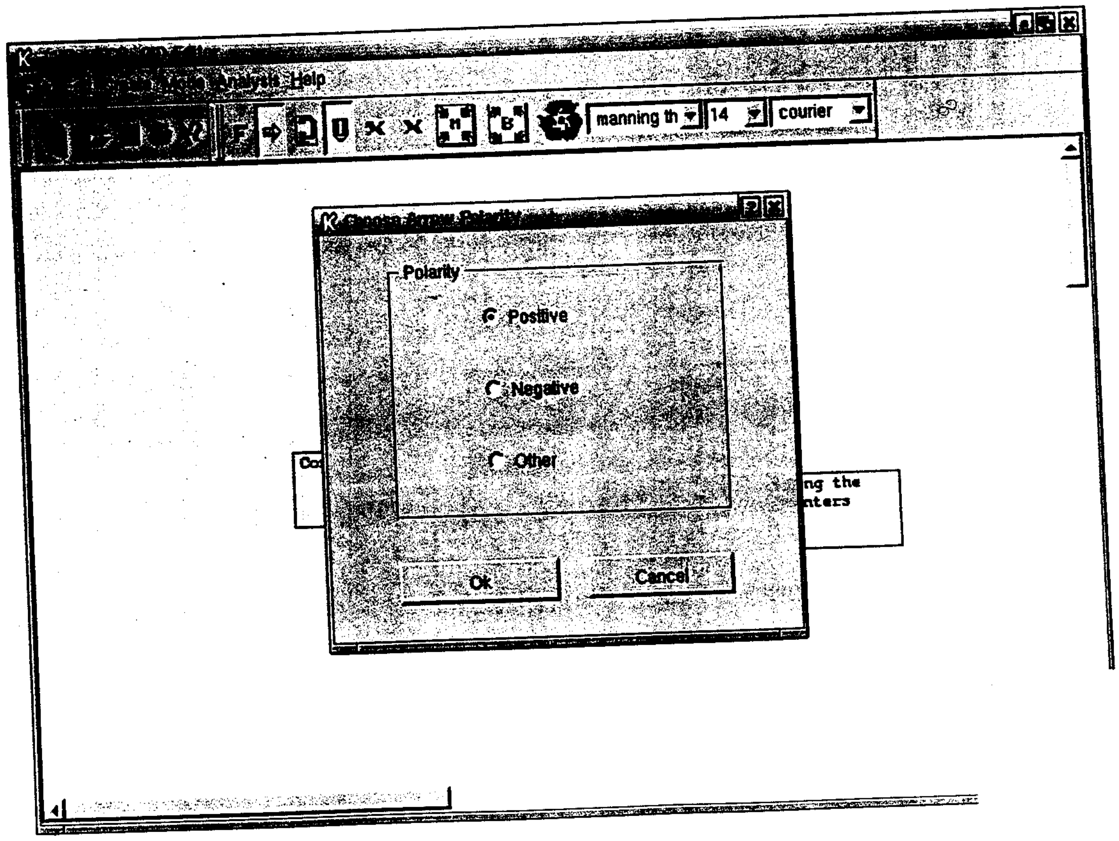




citizens

Corporation of Trivandrum

manning the counters

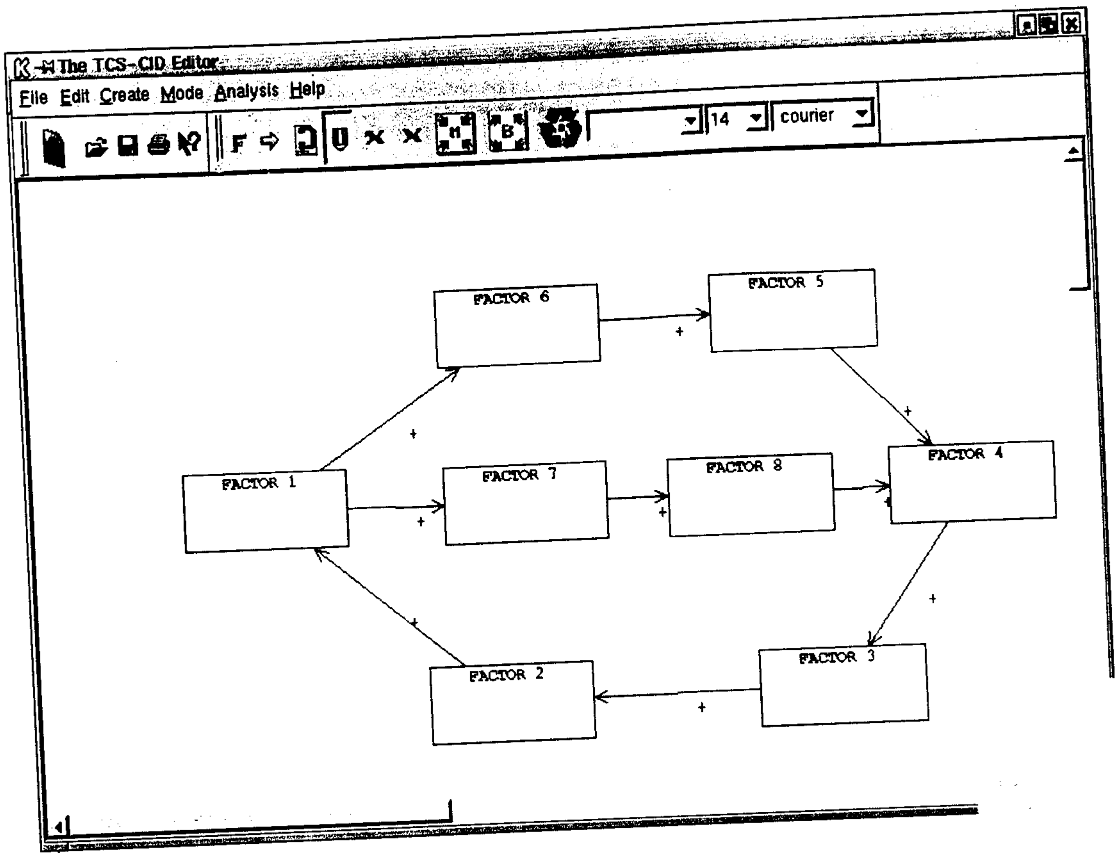


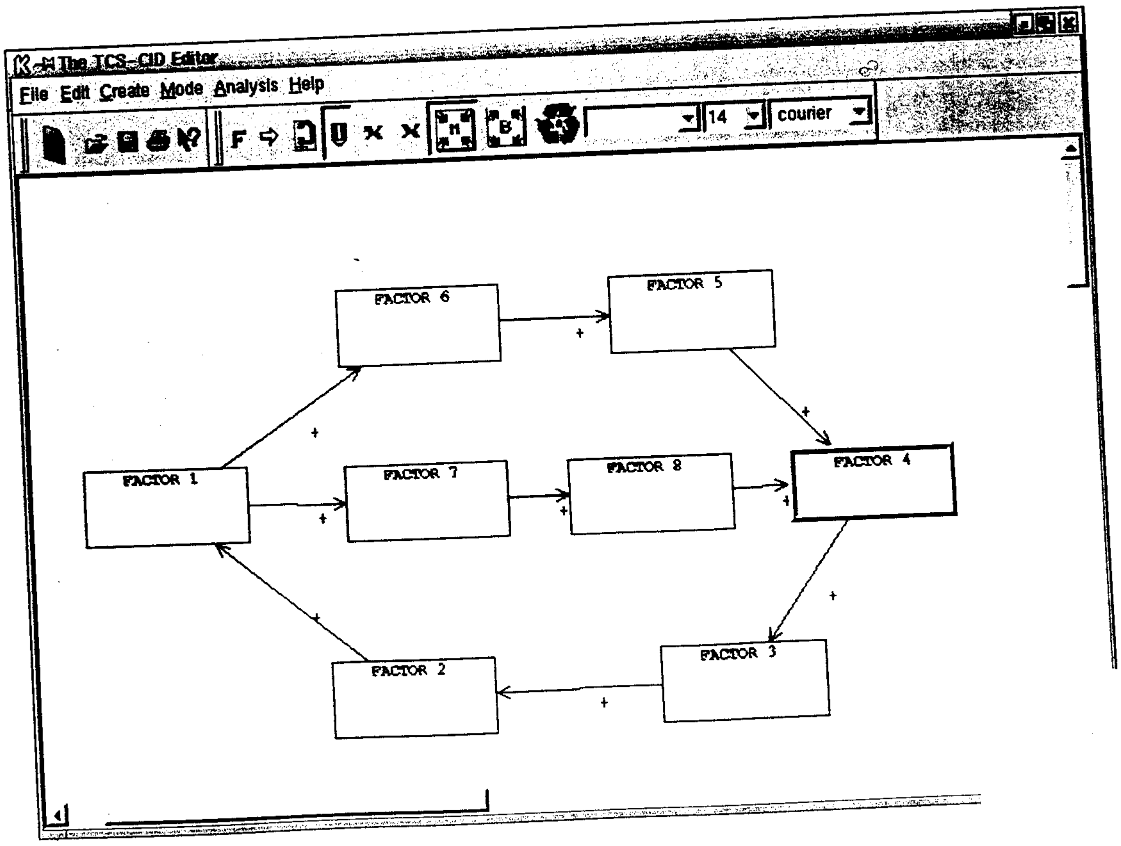
Polarity

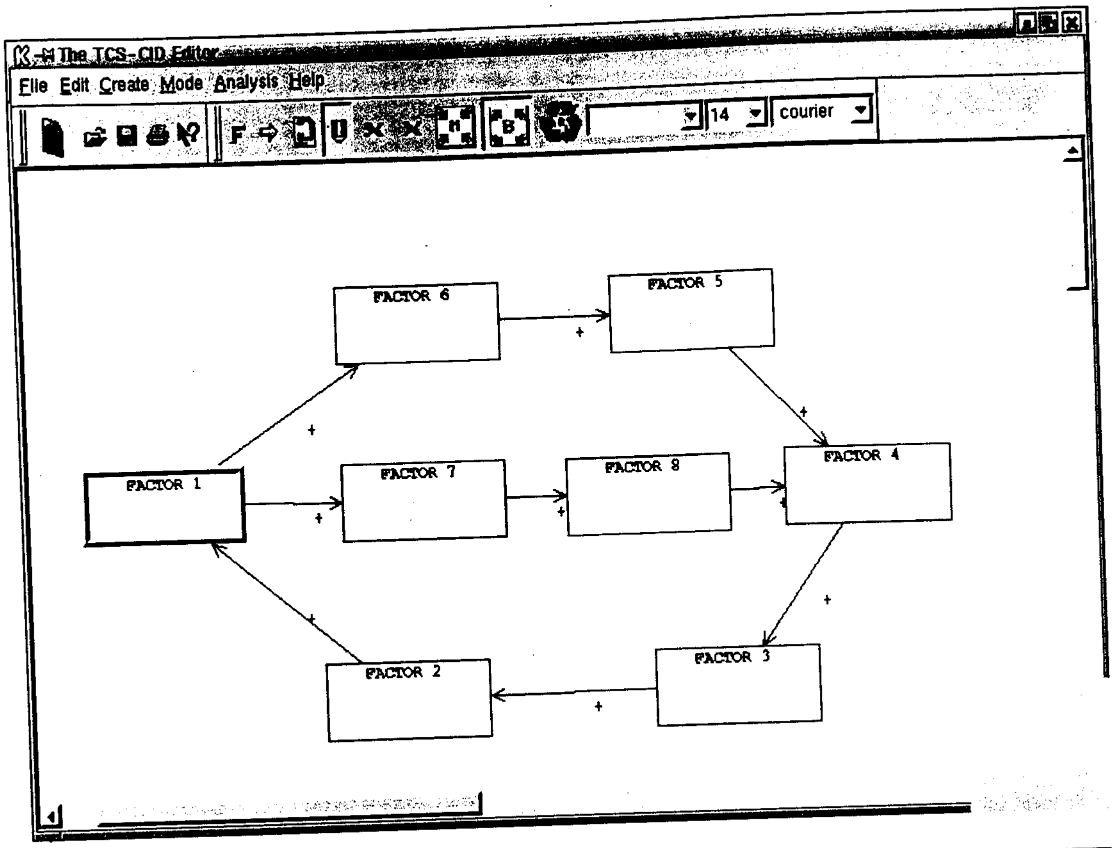
[✓ Positive]

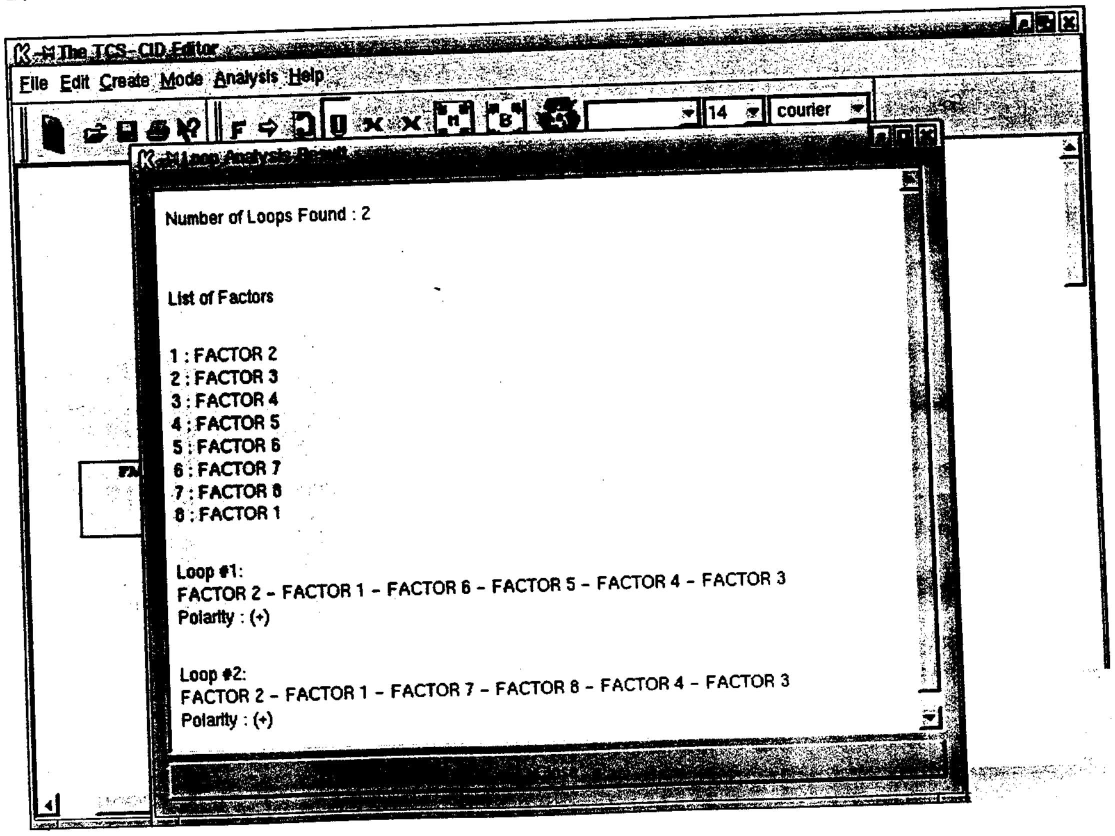
Negative

Other


The TCS-CID Editor
File Edit Create Mode Analysis Help
14
courier
FACTOR 6
FACTOR 5
FACTOR 1
FACTOR 7
FACTOR 8
FACTOR 4
FACTOR 2
FACTOR 3
+


The TCS-CID Editor
File Edit Create Mode Analysis Help
14
courier
FACTOR 6
FACTOR 5
FACTOR 1
FACTOR 7
FACTOR 8
FACTOR 4
FACTOR 2
FACTOR 3
+





Number of Loops Found : 2

List of Factors

1 : FACTOR 2
2 : FACTOR 3
3 : FACTOR 4
4 : FACTOR 5
5 : FACTOR 6
6 : FACTOR 7
7 : FACTOR 8
8 : FACTOR 1

Loop #1:
FACTOR 2 - FACTOR 1 - FACTOR 6 - FACTOR 5 - FACTOR 4 - FACTOR 3
Polarity : (+)

Loop #2:
FACTOR 2 - FACTOR 1 - FACTOR 7 - FACTOR 8 - FACTOR 4 - FACTOR 3
Polarity : (+)

# ACRONYMS

| | | |
|---|---|---|
| API | - | Application Program Interface |
| ASCII | - | American Standard Code for Information Interchange |
| BNF | - | Backus Naur Form |
| CID | - | Cybernetic Influence Diagram |
| DFD | - | Data Flow Diagram |
| DOM | - | Document Object Model |
| FTP | - | File Transfer Protocol |
| GPL | - | General Public License |
| GUI | - | Graphical User Interface |
| ISM | - | Interpretive Structural Modeling |
| ITP | - | Internal Training Program |
| LALR (1) | - | One token Look Ahead Left Recursive |
| LSB | - | Least Significant Bit |
| MSB | - | Most Significant Bit |
| MUD | - | Multi-User Dimension |
| ODBC | - | Open Database Connectivity |
| SAX | - | Simple API for XML |
| SECC | - | Systems Engineering and Cybernetics Center |
| SNAC | - | Stakeholders, Needs, Alterables and Constraints |
| SQL | - | Structured Query language |
| SSH | - | Secure Shell |
| STL | - | Standard Template Library |
| TBL | - | Transformation Based Learning |
| TCS | - | Tata Consultancy Services |
| UML | - | Unified Modeling Language |
| VMS | - | Vitual Memory System |
| XML | - | Extensible Markup Language |

# BIBLIOGRAPHY

1. Grady Booch, James Rumbaugh, Ivar Jacobson, *The Unified Modeling Language User Guide*, Addison-Wesely,1999.

2. Ian Sommerville, *Software Engineering*, Addison-Wesely, Fifth Edition, 1999.

3. Joseph O'Connor and Ian McDermott, *The Art of Systems Thinking* – Harper Collins Publishers, 1997.

4. John R. Levine, Tony Mason, Doug Brown, *Unix Programming Tool – Lex & Yacc*, O' Reilly, First Edition, 1995.

5. Roger S. Pressman, *Software Engineering – A Practitioners Approach*, McGRAW- HILL Guide, Addison-Wesely.

www.dictionary.com

doc.trolltech.com