National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file   Votre référence

Our file   Notre référence

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

Canada

# COMPUTER SUPPORTED COOPERATIVE WORK

# IN

# SPECIFYING SOFTWARE REQUIREMENTS

K.S.N. UDUPA

A Major Report

in

The Department

of

Computer Science

Presented in Partial Fulfilment of the Requirements

For the Degree of Master Of Computer Science

Montreal, Quebec, Canada

December 1992

©K.S.N. Udupa,1992

National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-84695-X

Canada

# ABSTRACT

**Computer Supported Cooperative Work in Specifying Software Requirements.**

**K.S.N. Udupa**

This major report presents a practical study, in which the principles of Computer Supported Cooperative Work (CSCW) are applied to elicitation of software requirements. For the sake of practical work, a commercially available software system called, GTCS (Groupware Tele Communication Software) is used. The practical implementation is not fully complete, but it demonstrates the feasibility of using GTCS for the purpose of this project

Two major areas of research are summarised in this report. One is software requirement specification and the other is Computer Supported Cooperative Work. In eliciting the requirements of proposed software system, we identify three categories of participants : client, software developer, and typical intended end users of the software system. While developing the software requirement document, these three people will interact with each other. It is proposed that a group of co-workers will organise themselves into several small teams. In order to facilitate their inter working, the work space on the screen will be based on three windows, one for the private use, one for the local use within a team, and third for the global view within the group. Support for the groupwork based on three windows is discussed in this report for the above purposes.

As part of this report, the GTCS software system was implemented in a PC environment. Its application to the software requirement elicitation problem is demonstrated through an example. This example is a simplified problem for the design of a real time controller for a Gas Turbine Engine.

Limitation of GTCS in its application of software engineering are analyzed and presented for the purposes of future product enhancements. Applications of Computer Supported Cooperative work to software engineering is a problem with too large a scope for a single project like this. Thus, potential avenues for further investigations are also given in this report.

*To the memory of my father*

# ACKNOWLEDGEMENTS

*I wish to express my appreciation to Dr Radhakrishnan for his professional assistance and insightful guidance during the course of this project.*

*The financial assistance provided to Dr Radhakrishnan by Bell Northern Research(BNR), Nuns Island, in support of GTCS was indispensable and is thankfully acknowledged.*

*I am greatly indebted to my wife, Ranjani for her unflagging support throughout the project. Without her encouragement, especially during the initial stages when I needed her support the most, this project would not come to fruition.*

*For putting up with my absences when she needed me at home, I thank my daughter, Sharmila. May she learn about the joys of sacrifice rewarded.*

*And finally, I remember with respect my venerable uncle Shivaramanna, who wanted me to complete this more than anyone else, but is not here to see this.*

# TABLE OF CONTENTS

CHAPTER I

CHAPTER II

CHAPTER III

CHAPTER IV

CHAPTER 5

# LIST OF FIGURES

# CHAPTER 1

# CSCW DESIGN AND IMPLEMENTATION ISSUES

## 1.1 Introduction

A large software project involves several people for its specification, design, development and maintenance. Quite often these people are not co located in the same place. This situation offers a good potential for using the techniques of Computer Supported Cooperative Work (CSCW) in the various stages of a software life cycle. In this major report, we present some preliminary investigations towards the use of CSCW techniques for acquiring, analyzing and specifying software requirements.
Based upon our literature research and study, first we present the various design and implementation aspects of the CSCW. In the following chapter we present the various aspects of software requirements elicitation.

Groupware Tele Communications Software (GTCS) is a commercial groupware tool. It is a multimedia multipoint teleconferencing software. It provides basic facilities for CSCW environment. GTCS offers a simple and efficient technique for controlling the flow of information among participants, and it can be configured on a wide area or local area computer networks. We believe that commercial tools like the GTCS, must be used to achieve goals such as ours. Because of its ease of installation, minimal user training requirements and flexibility of configuration, GTCS is a good tool for CSCW application. With this in mind, we have acquired GTCS for a PC platform and implemented it on a network of PCs. In the third chapter of this report we study the suitability of GTCS and we make use of a case study involving the specification of software requirements for a real time control system. In the fourth chapter of this report we address the shortcomings of the GTCS and suggest further enhancements for making it suitable for a wide variety of CSCW applications.

The work presented in this major report is only a ground work for the exploration of applying CSCW techniques in software requirements elicitation. We have focused our discussions on software requirements stage, although the CSCW techniques are applicable to other stages of software development, because it is the least supported stages in software life cycle. Also, people with different backgrounds participate in this stage and they use documents that are usually available to them in multiple media such as text, images, and drawings. The informal discussions held among the participants in this stage make the CSCW and multimedia applications to be natural and very useful.

## 1.2 CSCW - History and Definitions

Personal computers and computer work stations have penetrated large segments of our population at work and at home, making us aware of their values as well as their limitations. The end goal of these developments is to provide support for people to work with each other, using computers to facilitate, mediate, and enhance our skills and capabilities. Personal computers, as the name implies, are designed with an objective to support individual work in a fixed environment. Even with advances in the lap top computer models, toting a lap top to a business meeting is still cumbersome if not impossible. Most of us revert to the paper and pencil model for note taking and for storing and recalling facts. Furthermore, at the boundary between individual and group work the limits of current technology are most visible. Advances in computer communication and networking have succeeded in connecting personal computers together to adapt them to a group work environment. Computer Supported Group Work is based on the expectation that far more than better note taking can accrue from bringing work stations into group settings.

Over the last half a dozen years, Computer Supported Cooperative Work (CSCW) has emerged as an identifiable research field focused on the role of the computer in group work [1,3]. The issues being discussed relate to all aspects of how large and small groups can collaborate in group work. How should people plan to work together, to take advantage of this powerful medium? What kind of software should be developed? How will group work be defined and redefined to tap the potential of people and technology? The answer will come from research across a range of disciplines. CSCW is in the centre of this research.

CSCW research is examining ways of designing systems - people and computer systems - that will have profound implications on the way we work. It is a young field, drawing on a diverse set of more established disciplines It has a history going back at least forty years.

The technology of CSCW has some distinctive characteristics resulting from its focus on people and their working relationships. CSCW is not an electronic mail system. Computer conferencing can be closely related to CSCW application because of the use of the shared databases of messages, with access to the messages based on the roles of people in a group. The most fundamental aspect of group work is the coordination among participants. A CSCW environment must take into account the coordination technologies utilized in transaction oriented databases for concurrency control and access control with a view of having the participants build something together, not just preventing them from inadvertently corrupting the data in a shared space.

A CSCW system must take into consideration the ways in which coordination and collaboration among participants are managed. In addition to common data formats for information exchange among the participants, and common user-interface abstractions for general purpose operations, the CSCW environment requires another dimension for the integrated work station: There will now have to be common ways for users to deal with people-related aspects of their work across all their application systems.

The software that supports the CSCW application is called Groupware. Groupware refers to software where the user is a "group". More specifically the groupware can be defined as a software system that supports two or more, possibly simultaneous, users working on a common task and that provides an interface to a shared environment. It is software designed to take group work into account in an integral way. Groupware products range from the electronic mail like coordination tools to forms tracking systems and document commenting process managers What they have in common is that they put coordination technology into the hands of the group members, giving them access to the positive aspects of coordination - not just preventing collisions, but enabling collaboration. Groupware will be made commonplace by evolving understanding of what the key coordination technologies are, how they should appear to end users, and what the software libraries are that embody this understanding.

As a research field, CSCW is distinct from any of the fields on which it draws The fields from which we borrow methodologies - sociology, anthropology and organizational design- specialize on computer related phenomena but do not focus on interactions between computers and people. CSCW is primarily concentrated in the interaction between computers and people; how computers can be used in interaction between people.

## . *Why CSCW?*

As Vannevar Bush foresaw [16], it is not possible for us to manage all the information we collect. People are now more willing to master a new technology or change the way we work in order to gain the needed aid to manage the information we require. Perhaps most importantly, the computer is becoming less and less a novelty that distracts us from our "*real business*". It is simply a device, a communication medium, a part of the process.

As computers offer more and more computing power and capabilities, the requirement for computer services is also growing in the same proportion, if not by a bigger magnitude. We are finding more ways of utilizing the computer, to suit our application needs. However, all the computers, including super computers and personal computers, are a kind of centralized problem solving architecture, which requires people to work together in a central place to solve a particular problem. Interactions between various people working on a common problem will have to take place during a meeting or discussions, which requires physical presence of the individuals concerned, at one geographical location.

Developments in the communication field have essentially made the world a much smaller place than it used to be. The concept of a centralized work place is not a requirement any more in a non-manufacturing activity. Many of the design problems could be worked on without one having to get all the individuals concerned at a central place. Distributed work place goes along well with the concept of distributed architecture. If people can somehow electronically be coupled to each other so that they can collectively contribute to the problem solving, then the requirement for people to meet in a central place, such as an office, to work, will be reduced by a great deal.

Computer Supported Collaborative Work is a name attributed to a process wherein people at different geographical locations work together collectively towards some task. The various people resources required to solve a problem or for achieving a certain design can be scattered, and they do not have to commute to a common place to meet and discuss about the problem. As developments in this field progresses, people will be sitting in their homes and working together on a common problem through the computer medium. One can just think about all the good things that may turn out if we did not have to commute to work every day.

Another advantage of collaborative work method is the features it offers for resource collection. The resources that are required to solve a problem, may be scattered all over the world. A collaborative process would result in the increased productivity, reduced overhead, and most of all a goal of achieving a scenario where the whole world can be thought about as a centralized work place.

## 1.3 CSCW Design Issues

A central concern in computer supported cooperative work is coordinated access to shared information [17]. Data Base Management System (DBMS) technology provides data modelling and transaction management, well suited to business data processing but not adequate for applications such as computer aided design or software development. These technologies are primarily concerned in preventing the accidental data corruption by individual users. Group design applications mainly deal with structured sets of data objects, such as design drawings and software modules, and complex relationships among data, people, and schedules. Advanced programming languages do provide excellent abstraction facilities for those classes of data but provide little support for data storage and sharing. Typically, application programmers have to write their own data management functions in terms of files and operating system calls.

A fundamental requirement for a co-operative work is that it provides a coordinated interface for all the participants. Such an interface is intended to let coordinating participants interact with each other easily and efficiently through the communicating media. The coordinated access to shared information must include support for object linking, inheritance, associative access, versions and triggers to manage communication through shared data that is required during the collaborative process. Access control, concurrency control and consistency are the main issues to be addressed in the data management area in a CSCW environment.

### . *Access Control*

Access control in the absence of concurrency will be discussed first, i.e., assuming users perform actions on the shared information sequentially one at a time. The rights of a user to perform operations on an object will primarily depend upon how contextual attributes of the users' computation match the access rights associated with the object and operation. A simple approach would be to have read, write, and execute permissions set for the owner of the section of the database or file. Extensions to the above access, such as appending information, which is less dangerous than modifying a given section, may be provided to users who do not have write access to the file. However, such an extension may not be sufficient for some applications, which have a more abstract model of information than a simple file type object. In a cooperative work, access control such as viewing and modifying certain portions of data is required for each participant. The problem with attempting to control access in terms of abstract operations on shared information is that the underlying operating system must be able to support this requirement. If the operating system is only able to provide very primitive type of access control (read/write), then the required type of access control must be implemented by the application program itself. Under such circumstances the access controls on the files used to store the information (i.e., read/write, etc.,) must be weak enough not to preclude any of the operations allowed by the application to different participants. This means that files must be left completely unprotected at the operating system level, because it is difficult to use stronger protection and keep it consistent with the applications' more complex requirements. A simple way to achieve this is to keep all sections of the file publicly modifiable by all the users. This approach is not safe in that the files are then vulnerable to accidental modifications and destruction by the users This may be alleviated by having the application program temporarily change its users ID from that of the actual user, to a special user ID allocated for this purpose, and the files in question can be made accessible and modifiable only by this special user ID.

Even with the ability to control access to abstract operations, the use of conventional access criteria based on owner, group and public is rarely sufficient to implement the generous policy desired in cooperative work. The following properties may be utilized to determine access to the file or database by the cooperative users

a) Roles: In a CSCW environment participants may assume different roles with regards to the group work considered, i.e, owner of a section of the design, the user, or the client in a Computer System Configuration Item (CSCI) definition. The role is an attribute of the user's computation that enters into the determination of whether or not to allow a given operation. This reflects the idea that the same users may have different privileges depending on what roles they are currently playing. Privileges may be enabled or disabled depending upon the current roles of the participants.

b) Extensible user-valued attributes: Permissions may be granted to only specific users to modify the given portions of a file. The access check for a modify/cancel operation must examine the user access before granting the permission.

It is useful to have different variants of the above access control to be provided in the system. In a multiple user environment, when a user attempts to modify the contents within a section, the access may be permitted as a confirmed modification or only treated as a request for modification pending confirmation, depending upon the role definition in effect. A trigger or a flag may be invoked in such a case when the requested change may be brought to the notice of the user who has access control for the requested change.

## *. Concurrency Control*

In a group environment different participants will be working on their own section of the file or database, which may result in concurrent modification of the database or the file. The conventional approach to ensuring database consistency in the face of concurrent access, is to ensure that each transaction on its own preserves consistency, and that the effect of running multiple transactions must be the same as if they had been executed one at a time. This can be ensured by using concurrency control methods such as locking, and time stamps on each access and change. This is acceptable as long as the transactions are short and the conflicts are resolved quickly. However, long user interactions cannot be managed in the same way. If one of the users holds the lock for a long period on a given segment of the file, he may prevent other users from modifying that section for a long time. This problem becomes very serious if the system crashes, or there is a network breakdown, making it impossible to determine whether a transaction was committed or aborted.

Concurrency can in some cases, be improved by exploiting the semantics of the data and abstract operations. However, these also face the same problems when transactions are long. If serializability is to be enforced for long interactive transactions, concurrency cannot in general be increased without incurring a higher probability of transaction aborts. Optimistic concurrency control methods do not lockout transactions but instead check at the end of a transaction, whether it saw a consistent state; if not, the transaction is aborted. The checking can be done in many of several ways: by analyzing the conflicts among transactions that read and wrote the same data items, by recording the version identifiers of data read, and checking that they have not changed, or by checking the assumptions made by a transaction about the data it read. Whatever the test, it must be made automatically at transaction commit time, using locks or some other appropriate mechanism for that brief duration. Long transactions are also vulnerable to aborting because of system failures. This can be alleviated by supporting save points within a transaction, such that the effect of the transaction up to the save point is made permanent even if the transaction subsequently aborts. If save points are not built into the underlying transaction mechanism, they can often be supported on top by the operating system. As an example, the transaction commit can have a continue option which, after committing the changes of the current transaction, creates a new transaction with the same state, holding the same locks. Another alternative approach will be to periodically save a small set of changes made by the participant, until such time as the participant terminates his editing session.

Most concurrency control algorithms have a fixed and inflexible way of deciding which transaction should wait (or abort) when a conflict occurs. More flexible policies are needed for interactive transactions controlled by users.

In the co-operative environment, it is also useful to inform a user which other user has the lock. The users then can informally negotiate and perhaps agree to take turns. This concept is used for real time conferencing applications. A floor passing mechanism is a simple form of this technique which is utilized in some implementations [RTCAL,5]. A self lock mechanism, where a user can set a lock on the data file is another approach. The user who sets such a lock can later inspect an associated log, to see if other users have accessed or modified the object, and may enter into a negotiation (outside the system) with the other users. Such support for user controlled coordination is analogous to distributed systems that support file sharing among cooperating rather than competing processes. As an example, on receiving a lock request that can not immediately be granted, the system can inform the process currently holding the lock; the process can release the lock, or refuse to do so, or specify that it will release the lock in a short period of time after completing the necessary internal cleanup operations

A long transaction or activity against a shared data need not be limited to a single groupwork session. It may be interrupted by a crash, or the user may leave his work unfinished at the end of the day with the intentions of continuing at some later time. By saving the necessary state information, such activities can be made to last days or weeks or longer. This scheme has been used in a CSCW work in the design of large database system [18]. In these systems it is up to the users to pick up where they left off and continue the groupwork. In other scenarios, the actions that need to be performed may be known in advance and can be recorded in a script, so as to provide a guarantee of execution. The system performs the actions in the background as and when the data becomes available.

### . *Consistency*

Consistency issues play an important and critical role in the collaborative design process. The consistency issues can be identified into two main areas:

a) Global Consistency: Global consistency can be defined as a situation, where each participant works with the same information. Global consistency issues ensure that when the various works are merged, a clean end product is obtained.

b) Local Consistency: Local consistency is defined as the situation where any modifications made by the participant are recorded only on the participant's local workspace.

The global and local consistency issues may be resolved with little effort if the collaborative process is treated as a single master and multiple observer process. In such an environment all the changes and modifications on the database or files will be executed by a single participant at a time, which will be the master.

In order to increase concurrency among participants, without increasing the frequency of aborts due to conflicts, the consistency constraints that the transactions are intended to preserve must be examined. The basic premise underlying most concurrency control mechanisms is that if a transaction saw a different value for a data item that it read, it would have performed a different update or external action. For interactive transactions involving a user decision, there may not be a very strong link between what data users see and what actions they perform as a result. It may therefore be acceptable to present users with data that are not guaranteed to be most up-to-date. In such a case a weak read operation may not see all of the effects of update, a transaction had already executed.

In a CSCW environment the participants are geographically distributed. Any access to a data object by a participant and subsequent modifications to that object must be now communicated to all participants and also to a centralized database if it is implemented. This process may involve in the transmission of the transaction over a physical medium to maintain the integrity of the data object, which results in communication delays before an update can be performed. It is extremely important that the communication delays associated with different work stations be minimal, in order for the participants to be able to work or modify with latest versions of the data. It is also very useful to have a "**notify**" [19] mechanism, which notifies the cooperating participants whenever a data item specification is changed. When the participants learn that the data they are looking at have changed, they can then compensate for their actions, for example, by bringing back old versions or performing further changes. Some simple operations can be compensated for automatically. In others, the participants' intervention is needed to determine the right course of action.

A data consistency requirement may represent invariants that must always be true in order for operations on the data to work correctly. On the other hand many requirements may simply be desirable but need not be required to hold at all the times.

In certain CSCW applications participants may be allowed to make changes on the file concurrently without taking any conflicts into consideration. The transaction may continue with a conflict notification later on. This form of consistency is referred to as a weak consistency and is utilized in MPCAL [17]. The weak consistency may arise in applications such as design of software, where inconsistent states can be allowed to persist until a time of the users' choosing. Because individual transactions do not have to enforce consistency, the checking of the constraints need not get in the way of concurrent activity or cause conflicting transactions to block or abort. Rather the inconsistency can be detected and compensated for, after that.

Since users acting concurrently may not be aware of conflicting actions, some mechanism is needed for recognizing situations that require compensation. Appropriate users can then be notified about the changes. The consistency checks can be made by several methods [17].

- Pre-assigning an ordering, and seeing if any of the users read an object version older than the one most recently updated.

- Constructing a conflict graph on the basis of read sets and write sets and checking for cycles of certain kinds.

- Retaining information with each object as to what other objects were read when this object was updated.

Inconsistencies may also be prevented by use of a warning mechanism. The collaborative participants will typically be warned not to concurrently update the data checked out. The users may have the option of ignoring the warning and working concurrently, but they will do so with the full knowledge that they will have to deal with the current changes at some later time. The probability of inconsistency can also be controlled using optional locks with automatic time outs. However, in a multi user environment, because of communication delays and failures in the network, two or more users could believe they hold the same or conflicting locks. It has to be ensured that such a situation does not arise, by the application level protocols.

## 1.4 Database Models

To maintain the CSCW database, several types of data base models may be utilized. The **centralized model**, the **centralized-lock model**, the **cooperative model**, the **dependency-detection model**, and the **roving-locks model** are some of the models that can be effectively used for managing the collaborative process [13]. The main property of the database model is its ability to support a distributed computing environment.

The following sections discuss the various database models that can be effectively used in a CSCW application.

### . *Centralized Model*

The centralized approach has been used successfully in many applications [13 Colab]. It ensures that all participants use the same data. There is only one copy of the database, and concurrency control is straightforward. To coordinate simultaneous changes, database transaction mechanisms must be used. The centralized model could take a long time to retrieve data, as the data may have to be retrieved and stored from a centralized location.

### . *Centralized-Lock Model*

This model corrects the slow retrieval problem of the centralized model by having the copy of the data on each of the work stations. Data for the visual display are always fetched directly from the local memory of each of the participant, and are updated whenever changes are received. In this model, each computer has a copy of the database, but cannot make changes to an item until it obtains ownership of that item. There can be one lock for the whole database, or separate locks for different parts of it. With multiple locks, changes to different parts of the database can proceed in parallel.

### . *Cooperative Model*

In the cooperative model, each participant will maintain a copy of the database, and changes are installed by broadcasting the change without any synchronization. By itself, this approach entails the following inherent race conditions:

If two participants make changes to the same data simultaneously, there is a race to see which change will take effect first, and the results can be different on different machines. The participants must be aware of the race conditions so that they may use verbal cues ("voice locks") to coordinate their behaviour; under such a situation, it is rare that participants will change the same data at the same time.

As the changes are implemented by a verbal communication, the social aspects between the participants need to be considered. In this model it is necessary to coordinate the activities of the participants and support the social mechanisms for both partitioning work and reaching agreement that meetings by their nature rely on.

### . *Dependency-Detection Model*

The dependency-detection model corrects some of the shortcomings of the cooperative model by annotating data with a stamp describing the author and the time of the change. Every request to change data broadcasts several things: The new data, its stamp, and the stamp of the previous version of the data on the originating participant. When a station receives a message requesting a change, it first checks whether the previous stamp in the request is the same as the stamp in its database. If they are different, a "dependency conflict" is signalled. The conflict is to be resolved by a process that involves human intervention.

The advantage of the dependency-detection approach is its responsiveness. Changes to data do not first require serialization or the delay of obtaining the lock. The system assumes that a change can always be made, but it may have to fix things later if a conflict is detected.

Like the cooperative model, the dependency-detection model contains inherent race conditions, but it is able to detect them after the fact. If two participants change data at the same time, at least one of the machines will detect a dependency conflict as described above. However, it is possible to get "false alarms' if messages about changes to data from different sources arrive out of order; a dependency conflict would then be incorrectly signalled. Similarly, if two participants made a series of nearly simultaneous change to a datum, multiple false alarms might be signalled.

### . *Roving-Locks Model*

The roving-locks model tries to reduce the delay in obtaining locks that is incurred with the centralized-lock model, by distributing the lock-granting processes along with lock ownership. In this method, control of the data item is distributed, leading to a sort of working set for locks. In this scenario, a participant's machine would tend to acquire the set of locks for the subset of the database on which it is actively working. Most lock requests would require no communication with other machines. After the first access, delay in getting a lock would be significant only in those cases where the lock is on a remote machine, that is, when two or more participants are actually competing for the same part of the database.

## 1.5 User Interface Requirements for CSCW

An user interface for Computer Supported Co-operative Work (CSCW) requires different performance characteristics as compared to an interface for a single user environment. The CSCW user interface must concentrate upon how users should interact with the machine in a group environment. The user interface for a Groupware application presents challenging implementation problems, as the requirements for person to person, and person to machine interface are to be considered in the design and implementation. They are also particularly required to be well suited to the network supported workstations and networked window servers [7].

An user interface (UI) display for a CSCW application must invariably consist of a shared window type display. The display must as a minimum must include two windows. One of the windows must be a public window, the display on which will be the same as the display on all of the other participants. The other window, which we will call a "private space", will contain the display that is visible only to the local user. A CSCW user interface must also be able to support the following performance requirements [11].

. **Spontaneous interactions -** A CSCW user interface must be able to support spontaneous interactions. Its main objective is to minimize the conference start up overhead, to minimize the time to bring private windows in to the conference and to support participants who join the conference asynchronously.

Not all group interactions occur in the context of scheduled meetings in a CSCW environment. In fact, many interactions between collaborators are spontaneous and unplanned. This means that users cannot always predict who they will be interacting with, nor can they anticipate which windows need to be shared. For example, consider a collaborative debugging process. A user may want to say "Something went wrong. It is here on my screen now. I would like you to take a look at it and help me debug it". In order that shared window systems support these kind of interactions, initiating a conference must be a light weight operation, so as not to be a deterrent to spontaneity. In addition, it should be possible for latecomers to join an ongoing conference.

A shared window system should also allow users to pick up private windows and move them on to a different display (to allow coworkers to work on them asynchronously) or to bring private windows into a shared work space.

. **Work space management -** A shared window system must provide proper management of user windows associated with a conference. It has to ensure that instances of a shared window are kept consistent or that the user can easily distinguish private windows from shared windows, or all windows associated with the particular conference. A workspace manager is very much like a window manager. It controls the layout and size of a user window. In shared window management it is required to manage both the shared and private windows.

A shared workspace management must also allow conferees to distinguish a) shared window from private windows, b) all windows associated with a given conference, and c) which of several conferences a window is associated with.

These facilities can consist of simple visual cues to identify shared windows, or mechanisms for bringing all windows of a session to the top. Alternatively shared work space managers can provide mechanisms to organize shared windows into shared workspaces, separate from each user's private workspace. In a replicated environment, the use of private window management can lead not only to appearance problems, but to incorrect operation due to inconsistencies in application state. A way to distinguish private windows from shared windows is to employ grouping facilities. In this approach, a separate workspace is created for each shared session. The important thing is, the system must provide some facility for the users to lay out windows in the shared workspace independently from other participants.

A strict "WYSIWIS"[2] ( What You See Is What I See) approach workspace can make the collaborative process more effective by providing a common frame of reference to conference participants. This is important when trying to use telepointing facilities that span the entire shared workspace.

The WYSIWIS approach solves consistency and presentation problems that arise when private window managers control shared windows. However, the "WYSISWIS" approach has one important limitation from a user's point of view: as a result of sharing a single conference window manager, concurrent operation within the shared workspace is not possible; that is, it is impossible for different collaborators to work on different conference applications at the same time. An ideal collaboration aware window manager would allow each window manager to co-ordinate with other window managers in its conference to guarantee consistent sizing of shared windows.

This kind of window manager would provide visual cues to distinguish shared windows from private windows, and to perform stacking operations on all windows in the same session. The collaboration aware window managers could better support concurrent activities.

. **Floor control** - A CSCW user interface must assist in "moderating" a conference. It must be able to resolve conflicts. For example, if any of the participants is permitted to take control of a session, then a conflict will arise, whenever two participants want to take control of the session at the same time. It must also be able to adapt to increases in communication delay or number of participants, where running an open floor may not be acceptable. General guidelines as to who can communicate with whom must be addressed by the floor control mechanism. In a true multi-media conference, it is possible to distinguish a separate floor for each medium. A floor control policy may permit only one user at a time to control the entire shared workspace, or be like an "open floor" mechanism where any participant can generate any input to any window. The floor control policies can be characterized in the following way.

a. The number of floors - one for the entire conference, one per shared application, or one per window.

b. The number of people, who can hold the floor at the same time and,

c. How the control is passed between potential floor holders. This requires the potential use of auxiliary communication channels, specifically audio.

The first two of the above help resolve the issue of the amount of concurrent activity to be permitted. Separate floors correspond to sub-conferences and number of holders per floor corresponds to how the meeting is being run. With respect to how the floor is passed, this is where technology can clearly be used to improve upon traditional social protocols. One example will be the system can maintain the lists of people who have asked for the floor, and can optionally introduce hysteresis by selecting from this list at random, rather than from the head of the queue. In any event, design of a floor control mechanism that is inflexible is not desirable. For example, in a collaborative work when one of the sessions involves brain storming and another one person at a time activity, it is necessary to change the floor control mechanism as required.

Running of an open floor can be increasingly difficult when number of participants increases. Also as communication delay increases running an open floor mechanism becomes increasingly difficult. The availability of audio/visual links will significantly influence the floor control mechanism.

. **Telepointing and Annotation -** The CSCW user interface must provide tools for telepointing and annotations. Telepointing and annotations are the common features adopted in a group environment for attracting attention of the participants in the group discussions.

. **Performance** - A CSCW user interface system must be highly responsive. In particular, the user must notice no difference from running under the base system.

The workstation based network computing environments supporting teleconferencing provide a reasonably adequate interface for collaborative work [4]. These systems would enable geographically distributed individuals to collaborate in real time, using multiple media (audio, video, text, graphics and facsimile) and all available computer based tools, from their respective work places. The main goal for a user interface design for a CSCW application is to assume that most of the participants in the CSCW are novice computer users and are capable of working with the computers with minimal training.

### . *Interface Design Requirements*

How computers should be used to support meeting activities, and how users should interact with them are complex and as yet poorly understood problems. The main interface requirements for CSCW environment can be summarised as follows [12]:

- Easy transition between individual and group work;
- Privacy of individual work spaces;
- WYSIWIS (" What You See Is What I See ") paradigm for displaying shared information;
- The ability to keep pace with the meeting conversation;
- Maintenance of existing special structures among meeting participants;
- Minimal training requirements.

### . *Individual and Groupwork Transition*

CSCW usually consists of participants working together on a common problem However, the participants may at times be required to work on their own, to analyze a particular problem that may have arisen from the groupwork Furthermore, users may also utilize the same system for their individual work as well as groupwork. If people are comfortable performing their individual work in the collaborative system, then their individual work products will be part of the shared environment from the beginning. which may not be suitable. Also when the collaborative process extends for a prolonged session, the participants may be required to work alone, in their own work space from time to time. This requires that the system must be able to support the user requirement that they may be able to switch their system between groupwork and individual work.

Easy transition between individual work and groupwork is a mandatory requirement for a groupware user interface. Participants must be able to shift easily between working on a task privately on a personal computer, and working collectively, without major adjustments. The switching between phases of private work within the meeting and collaborative work must be supported with minimal effort and delay by the groupware interface.

### . *Privacy of work space*

By its very nature, the design of groupware systems and their associated interfaces focuses primarily on the collective efforts of the participants to work as a unit. In a groupwork however, the requirement for a private work must not be overlooked. One phenomenon in a public work space is that it creates a public showcase for user's poor typing, or their errors in using the system or software. Although a user might not be concerned about making mistakes when using a personal computer in the privacy of his or her office, making the same mistakes in a social setting might be intimidating and unacceptable. Furthermore, even in a CSCW environment, participants need to work privately. The private work will be required for each participant to make important notes or memos. Even extended work on one piece of the groups' task are common activities that participants engage in, during the course of a meeting. To support these activities, a computer supported meeting environment needs to provide the users with a private window on a shared computer so that they can work independently.

### . *WYSIWIS*

WYSIWIS or What You See Is What I See refers to the presentation of consistent images of shared information to all participants [2]. In a WYSIWIS system, all meeting participants see exactly the same thing and where others are pointing. In a meeting environment where each participant has a display monitor, the display of shared information (the information that all participants can see) can be accomplished in many ways. One approach is to present the shared information on each individuals' personal screen and allow each user to control its features (such as location or size) or how it is displayed. Such a method was used with early groupware tools such as cognoter in the colab by Xerox PARC [13]. However, it is also to be noted that the ability for individual users to customize their views of shared information causes referenced difficulties in conversation. People often use spatial descriptions to refer to items that others can see, such as the "The third line on the second paragraph" etc. If meeting participants can tailor the arrangement of shared items on their personal displays, such referencing methods can not be used. Consequently, for referencing, the shared information presented to all users by the computer system should be displayed in the same way for each user.

### . *Keeping pace with conversation*

The objective of an interface for a CSCW environment is to provide users with a scenario for a face to face meeting environment. People meet face to face because of the high communication bandwidth they can achieve with direct audio and visual channels. In this context computer-mediated communication is too slow to serve as a primary communication channel. This necessitates another important requirement for the user interface: it is necessary to keep pace with the conversation so as not to impede the primary flow of information. Specifically, the system should respond quickly to user commands, because users may need to quickly shift between documents, portions of a document, or even applications as the focus of the conversation shifts. This also means that users should be able to accomplish their tasks on the system quickly - if a user must focus attention on manipulating the computer system for more than a few seconds, the conversation may shift direction, lag, or otherwise sustain a loss of momentum.

The user interface for CSCW must permit the users to enhance their communication, for example, by displaying information that can be referred to, by providing a meeting record, or by providing capabilities the participants would not otherwise have. However, it should not be expected to replace the normal verbal and visual communication of a synchronous meeting.

### . *Support for protocols*

In a conventional meeting a protocol that dictates such issues as to who may speak or who has control over the meeting is an important factor. Social protocols and social structures are an integral component of all meetings. Good groupware must be able to offer similar aspects.

Several of the groupware tools developed include facilities that control the progress of collaborative work. The method utilized in cognoter [13], assumes that brainstorming and organizing ideas are independent activities that occur in sequence, and cannot be interleaved. A groupware system that assumes or imposes a particular meeting structure on a group can disrupt that group's familiar meeting protocol and actually impede their ability to hold a productive meeting.

### . *Minimal Training*

The user interface for a groupware system must be designed such that the users require minimum training to use it. This is desirable for any software system. It is particularly important for computer supported meeting environments where the users may use the room occasionally or do not have any incentive or the time to learn a complex system.

## 1.6 Communication and Networking Requirements for CSCW

The main motivation for the collaborative work is the need to share resources. The main resources that are often advantageous to share are the communication facilities, computer facilities and the information itself. The amount of information to be shared is directly influenced by the network that provides interconnections among systems. The two or more computers connected on the network should be able to control displays in certain areas of each other's screens, and processes in certain sectors of each other's computers. The systems must also provide each participant with a wide choice of media for communication, such as text, graphics, and speech.

A Multipoint Communication Service (MCS)[33] may be used to provide the communication among participating groups in a collaborative environment. MCS is a generic service designed to support interactive multimedia conferencing applications. It supports full duplex multipoint communication between participants connected on the network. Both asynchronous and synchronous communication access must be supported by the MCS. The MCS provides three forms of communications: one-to-many, many-to-one and many-to-many. One-to-many communication involves transmitting a file from a sender to many receivers. Many-to-many communication takes place during simultaneous annotations in the conferencing applications at different sites. Our implementation for the project provides a full duplex, point-to-point communication as the participants are limited to only two individuals.

### . *Communication Requirements* --

In collaborative work each participant will be seated in his own office at a workstation that may include a high resolution screen for computer output, a keyboard, a pointing device such as a mouse, a microphone, a speaker, and possibly a camera and a video monitor. Parts of each participant's screen are dedicated to displaying a *shared space* in which everyone sees the same information. A voice communication equipment when provided, may be used by the participants for discussion and negotiation during the group work. Collaborative work supported by video communication can add an illusion of physical presence by simulating a face-to-face meeting; and conversational references ("this paragraph" or "this line") can be clarified by pointing at the displayed information. The communication network must support the editing and processing of the displayed information, and saving and retrieval of new information that is relevant to the discussion. It is also desirable that participants have *"private spaces"* on their screens that allow them to view relevant private data or to compose and review information before submitting it to the shared space.

It is required that the minimum support provided by the communication network must include the following communication capabilities[31].

. ***Bidirectional transmission :***

Collaborative applications require two-way communication - if not the capability of sending and receiving simultaneously. Full duplex form of communication is desirable in a collaborative work.

. ***Freedom from Error :***

One wrong bit may completely change the meaning, especially if numerical data are represented non redundantly. The end to end communication must be made error free through the use of adequate error handling mechanisms and must be able to recover from errors in the communication channel.

. ***High connectivity :***

The source must be able to transmit to any one or more of many destinations. A destination may need to examine many sources. One to many, many to many, and many to one type of communications must be handled by the network.

. ***High information rate :***

The collaborative process must emulate as much as possible the good aspects of a real time face to face meeting environment. This need requires that the communication channel must have a very high bandwidth.

*.Speech capability :*

Present day speech circuits transmit alphanumeric information inefficiently, and most of the present day data networks are not designed to transmit speech. The network with the capability to integrate data with speech will provide a very desired environment. The Integrated Services Digital Network (ISDN)[35], which services a wide variety of user needs, can be used to provide speech and data transmission on the same communication link.

*. Picture transmission :*

The data transmission must be able to integrate graphs, charts, diagrams and simple sketches, and also high resolution pictures. A document or a design file that is usually utilized in a CSCW environment will consist of not only the text, but also complex graphics, and the network must be able to support the requirement.

*. Short transit time delay :*

The delay introduced by transmission from one station to another may slow down the operation of a multi participant collaborative environment. As described earlier, to create a real time environment the transit delay must be kept to a minimum.

## 1.7 Requirements and Issues for CSCW Session

The main issues in the cooperative work environment are:

- *Addressability of common topics*

- *Interactive dialogue*

- *Effect on session throughput with multiple participants*

- *Decision making*

- *Asynchronous and Real time interactions*

- *Shared Screen conferencing*

. ***Addressability of common topics :***

In a development environment where more than one person is required for the task, each of the developers must be able to address the topic simultaneously. In a Software Requirement Specification document production, the client and the analyst are required to discuss a particular aspect of the control or implementation regarding a common object. The common object is required to be displayed at the different geographical locations of the client and the analyst.

. ***Interactive dialogue***

Some kind of computer-augmented two person telephone communication is a requirement for a successful CSCW implementation. Studies on voice based interactive human communication have revealed that it results in rich information transfer and produces an environment of face to face meetings[32].

Another form of supporting communication between participants is the "mailbox" approach. In this method the messages are exchanged in text form, where a "mailbox" is utilized to support the data transmission and receipt. "Mailbox" method is a form of unidirectional communication. In such a situation the person who receives the message is a passive recipient of information. Speech based interactive communication is very effective for transfer of information among participants as compared to the "mailbox" approach. Furthermore, the message throughput in a mailbox type approach will be less, as compared to the speech based interactive dialogue. Quality of audio however, plays an important role in the speech based interactive communication. Inability to hear clearly what is being said will generally end up ruining the group meeting.

. *Session throughput with multiple participants*

Session throughput is defined as the number of decisions that will be made, during a given session. With reference to the software requirement specification, the decisions may correspond to issues regarding the person machine interface designs, or rationalization of certain control algorithms, etc. Interactive dialogues will have a profound effect on the session throughput. Issues that invoke dialogues, even though delaying the decision making process, will result in properly verified decisions. A dialogue cycle, which is defined as a query and answer sequence, will result during discussions. A cycle of dialogues may result, before a decision is made on an issue. In the case of multiple participants, the dialogue cycles may result in initiating many independent dialogue cycles, which can cause reduced system throughput. A typical dialogue cycle is shown in figure 1.1.

. *Decision making*

Certain issues that cannot be unanimously resolved in group meeting are required to be solved by a voting process. The decision making process normally involves consensus among participants and may also require negotiations. Sometimes a decision will be made from a set of choices. The larger the number of choices, the larger the number of negotiations or dialogue cycles that will be needed. A voting process may be chosen for some choices, if all of the participants have a basic agreement on some of the multiple choices.

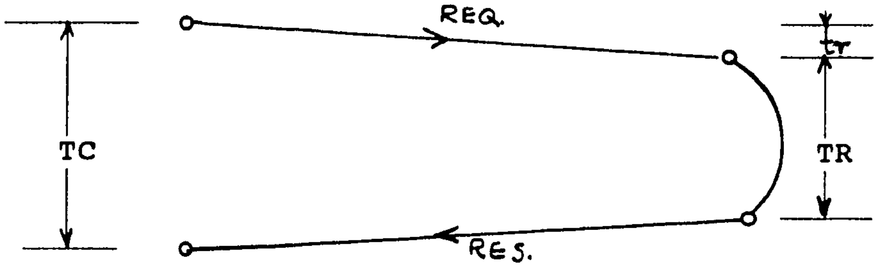


**Figure 1.1 - Dialogue Cycles**

TC (the cycle time) = TR (response time)

+ tr (transmission time).

To be effective, the transmission time, tr, must be much smaller than TR, the response time.

. *Asynchronous and real-time interactions*

Interactions are asynchronous when participants are allowed to participate in the work at their own speed. Real-time interaction is referred to as when all the participants are joining the collaborative work together and leave it at the same time Both kinds of interactions are useful for collaborative work. A small summary window indicating as to what the conference is about and which participants are present, (and which were invited to attend and which are unavailable or have not yet responded), and which participant has the control of the current session, is desirable to support asynchronous and real-time interactions. A one line events window, which displays important changes in status, such as when a participant is leaving, or joining, or passing control, will also be required on the participant's display screen.

## 1.8 A Proposed CSCW environment

This section defines the various terms and processes that are developed as part of my project, wherein the CSCW techniques are applied to the specification of a software requirement. The three participants in this specification will be the client, the analyst and the users. The client is the owner of the project, for which a software specification is developed. The users are people from the client organization who will be the end users of the software product. The analyst is the person responsible for the development of the software product.Each of the participating groups may consist of more than one person at any time. Each of the participating groups is designated by a node. A node will consist of a PC based work station, which is connected to other nodes via a communication link.

### . *Terms and definitions*

The following definitions apply to the proposed collaborative work. The definitions are made with respect to the participants for this project, which are the client, the user and the analyst.

**Node** : A node refers to a workstation that is part of the collaborative group. The node will be connected to other node(s) via a communication link. A node may be owned by a single person, or may be shared by a group of people.

**Active node** : A node which is involved in the present group work. Only an active node can be an observer node or master node. An active node is owned by a single person or shared by a group.

**Inactive node** : An inactive node is a node that is currently not participating in the group work. An inactive node may become active at any time in the design process. An active node may become inactive when the person(s) owning the active node leave the design process temporarily.

**Supernode** : A supernode is an always active node that oversees the group work, controls the design process, and provides certain management functions. This node does not participate in the actual issues of the group work.

**Master node** : Master is one of the active nodes, i.e., workstations through which the group decisions are made permanent. The master node is the only place from where the decisions are made. The decisions made at the master node are visible at all other nodes, i.e., the observer nodes.

**Observer node** : A non master node that is active. The decisions made at an observer node are visible only to the observer and his subgroup. The users or the analysts may be observers at various times. The observer node will have the capability to intervene in the decision making process by raising flags in the form of messages.

**Subgroup** : A group of observer nodes that make and pass decisions among themselves. The collection of nodes within a subgroup acts as one observer node for the master and behaves as a single group among themselves. A subgroup architecture is shown in figure 1.2

**Buddys** : Buddys are the participating members in a node. The buddys make a subgroup of a node. Each of the buddys may have their own workstation or they may share a single workstation among themselves. In large complex software design process, it may be necessary that a group of people may be required to analyze a particular software configuration item, as the knowledge of the requirement is usually distributed among various individuals.

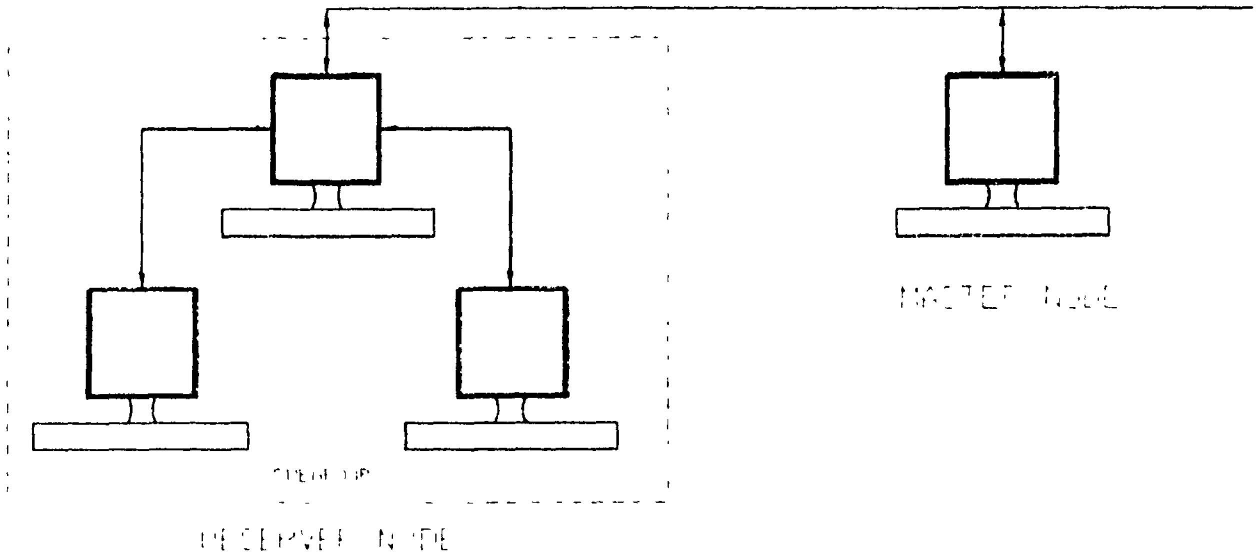


Figure 1.2 - A Subgroup architecture

**Master change:** Is an event that causes the current master to become an observer and one of the observers to become a master. Figure 1.3 shows a master change event

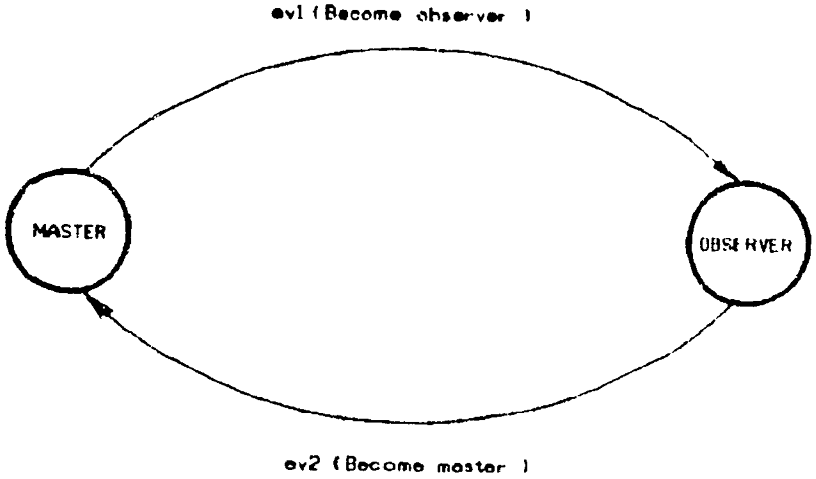


**Figure 1.3 - Master change event**

**Session :** A session is defined as the time between two consecutive master changes The length of the session may vary from time to time.

**Permanent changes:** Permanent changes are changes made on the "groupwork" file during the group work. They are implemented at the master node only.

**Temporary changes:** Temporary changes are changes made to the "groupwork" file by the observer nodes. The temporary changes usually result from the permanent changes effected by the master. Temporary changes are visible only to the local observer nodes or to the observer nodes in a subgroup.

**Stable State :** It is the process state when there is no master, i.e., all the active nodes are observer nodes and no permanent design changes are made. In this state the participants are discussing among their buddys in the subgroup, or analyzing the design changes effected by the previous session.

**Group decisions :** These are events resulting from the collaborative process, which may or may not result in the permanent changes. These decisions may be made permanent subsequently.

### . *CSCW sessions*

Work in a collaborative environment can commence whenever there are at least two active nodes in the group. A session in the collaborative process is defined as the time period during which none of the active nodes change their roles, i.e., a masternode will remain as a masternode and all the observer nodes remain as the observer nodes. During this time period one of the nodes assumes the role of the masternode and controls the collaborative work. The masternode owns the session for the duration of the session i.e., he is given access to make changes, add or delete the contents of the database or the document. The other participants who assume the roles of observer nodes take note of the changes implemented to the document during the session for further evaluation

### . *Session commencement*

A session commences whenever a minimum number of required participants are available for the work. This number is usually determined by the total number of participants involved with the specification generation process and how well the requirements are partitioned. The minimum number of participants required for any session will be two. In some applications a specification object may be partitioned so well, that each of the partitions can again be split in to sub-specifications and sub sessions.

A session which commences only on the availability of participants may not always result in the desired goal as regards to schedules on completion of the task. This kind of session may not be suitable for an industrial design or a real time control system specification environment. A time enforced session, which requires that all of the participants be available for a session, at pre-assigned times, may ensure the progress of work at the desired rate. In an industrial environment availability of participants always becomes a limiting factor, and that a mutually agreeable time table is difficult to set on the fly. A design schedule, hence, should clearly identify the session periods, and each of the participants shall ensure their availability at the marked period.

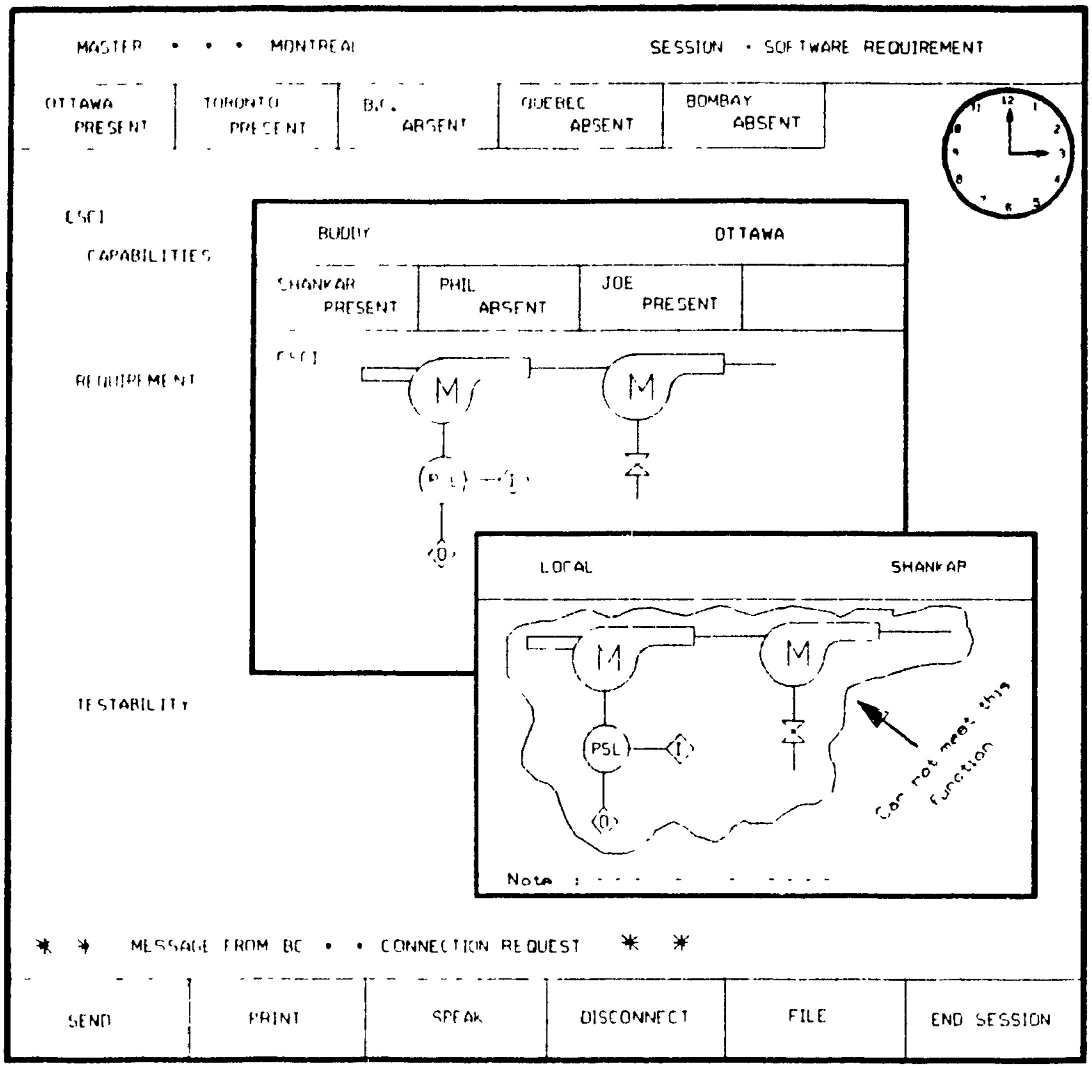
### . *Information visibility during a session*

The primary requirement of the CSCW system is to enable participants to interact with each other, pointing out changes needed in the document through an editor cursor, on a display which can be seen by all the participants on their own screen. To accomplish spontaneous interaction the users must have a similar view of the work being edited at all the times. I propose a "**three window**" display similar to the one proposed in the two window, DE-based MicroEmacs [10] type of display editor. Each of the nodes is equipped with three display windows. One of the windows, which we will call the main window, displays the text that is displayed on the masternode's window, complete with all the changes the master has made. This window displays every change and cursor movement made by the master, i.e., the observer's cursor will be in lock step with the master's at this point. The main window also carries a status window that will indicate which session is active currently (i.e., with respect to design partitions), which participants are present, (and which were invited and are unavailable or have not yet responded), and who the master is. In addition the window displays important changes, such as when a participant is leaving or joining the collaborative process or passing of control.

The second window, which we call the "group window", is visible only to the participants within a "subgroup" or "buddys". The "buddys'" in a subgroup share this display, for discussing relevant issues and modify the contents, when they are in caucus In other words, the group window concept will try to create a collaborative process, within the main collaborative work environment. I can see an application for this type of display, when the CSCW group consists of many distinct subgroups For example, the client, the developers, and equipment manufacturers, each may have their own subgroups during the specification work.

The other window, which we call the local window, is visible only to the local participant, and display on this is not shared with any other members of the group or subgroup. The participant may mark any changes that he wishes to on his local window As during the collaborative process the participant may have to switch between public work space and private work space, the availability of a local window is quite important A three window display screen is shown in figure 1.4.

At the beginning of the session, the local buffer, which drives the local window and the group window, contains the copy of the unchanged specification file or the "groupwork" file. The local buffer does not get updated automatically. Since a particular observer may assume the role of the master after a number of active sessions, his local buffer would remain as a scratch pad for most of the duration. The one disadvantage that may result, due to this group window visibility, is the longer resolution time; since each change may now result in many changes within a subgroup, a change shall only be acknowledged when all the participants in the subgroup understand and



* * MESSAGE FROM BC • • CONNECTION REQUEST * *

Figure 1.4 - A Tri window display system

implement the change.

For example, consider a control system software specification which is clearly partitioned with regards to the control aspects. Furthermore, let us assume that the partitions are specified in files Fa, Fb, and Fc, each of which is a different set of control functions that are required from the system. Let us also assume that the three partitions are owned by three individual groups A, B, and C. With A, B and C as the three partition owners, assume A1, A2, A3, B1, B2, B3, C1, C2, C3, etc. are the subgroups of A, B & C respectively. Now with each session the changes implemented by the master, say A, may result in each of Bs and Cs making changes within themselves. Now since B consists of B1, B2, B3 each of the Bs may end up consulting among themselves about the effect on their partition Fb due to the changes that A made on Fa. Each of Bs now try to look into the their local windows before accepting the changes that A made on Fa. This may result in more conflicts, and more interruptions to the design process. This problem may be alleviated considerably, if in each of As, Bs and Cs, one of the participant's is designated as a leader. In this case the leader may be allowed access to look through his teammates local windows and apply the necessary changes to his local window for necessary action during the time he assumes the role of the master.

### . *Session termination*

A session may be terminated at any time by the master. This is accomplished by the master by invoking some type of control command. This will result in a status line update, on the shared window, which indicates that the current master has relinquished control, and the current session has terminated. At this time the underlying software executes the necessary access control mechanisms to ensure that the file is protected from the previous master and assigns him the observer status.

### . *Master transfer mechanisms*

Commencement of each new session involves the appointment of a new master. Since each of the participants are considered equally responsible for the design, it is important that each of them is given certain amount of time to have the design changes/modifications implemented. In the design of large computer software projects, the knowledge of the system will normally be distributed among many participants. Each of the participants will have adequate knowledge on a particular segment of the design file. It will be simpler to have the role of master allocated if a design hierarchy can be maintained. In this case the design leader may appoint the master for each session. Even this may not be done arbitrarily. The following section discusses various methods by which sessions can be terminated and new ones begun.

#### a) **Time controlled sessions**

Time controlled sessions are defined as sessions that last for only a fixed amount of time.

**Motivation :** Time controlled sessions ensure that each of the design participants get a certain amount of time to take control of the group work. This will also ensure that changes on the work will be incorporated at regular intervals since each of the observers will get their turn to implement the changes that have either resulted from new changes, or are made as part of a new modification. Furthermore, allocation of a fixed amount of time for each session would also result in a more productive session as compared to an "open session", where the session time is kept as long as required. People tend to be more productive if they are given a fixed deadline to discuss issues in a meeting. One main factor that is to be considered in a time controlled session is the allocation of time for each session. The session interval must be determined such that each session would result in achieving the desired amount of progress in the group work. Defining this time is not a trivial task. Since each change results in a passive interactive time due to transmission delays, and awareness delays, a time will have to be chosen to satisfy a maximum transmission time and maximum awareness time. The awareness time is defined as the time required for the effected change to be understood by all of the participants. So a minimum session time is defined as follows:

$$\text{Ts (Minimum Session time)} > \text{N} * (\text{Maximum awareness time} + \text{Maximum transmission time})$$

Where N is the number of changes that will be attempted for implementation during the session. As the sessions progress, N decreases from a maximum to a minimum and the awareness time will be following the variation of N. In this case then, Ts will be a varying quantity and hence the underlying operating system will have to correct Ts for each new session. In the beginning, Ts will have to be fixed by some predetermined number of changes per session with a rough calculation of awareness time per change. A factor Tf is included for the calculation of subsequent Tss for the subsequent sessions.

**Side effects :** A time controlled session may invariably become a pre-emptive process, if strictly enforced. A pre-emptive type session may result in a total failure of the session as far as groupwork is considered. Additional tools may have to be provided in the operating system, so that a master may have provisions to extend the session time if required.

The time for each session in the beginning may be fixed by a consensus among the participating members. Participants may be permitted to change the session time as the sessions progress, when they will have a better understanding of the duration for each session.

### b) Owner controlled sessions:

In this type, a session would last as long as the master has changes to be implemented or as long as the current master holds it. A master is the only person who can relinquish control on the sessions.

**Motivation:** Since the change awareness time cannot be determined for each change, the changes that the master implements, may take as long as the process needs. Some of the changes suggested during the group work may take long to be understood by the participants, and hence an unlimited time for a session may result during the group work.

**Side effects:** In the case of long transactions during a session, which result in long session times, each of the participants may not be assured of control of the session over long periods of time. Also long sessions may be interrupted by a system crash or the master may leave his work unfinished at the end of the day with the intention of continuing at some later time. This requires that necessary status information maybe required to be saved into the system for a later day resumption.

## 1.9 Summary

In this section I have highlighted the main design issues involved in a Computer Supported Cooperative Work environment. A diverse set of requirements for managing data that is shared across many users, and some of the specific features for access control and concurrency control were also addressed. The various database models that are utilized in CSCW applications were also discussed.

The user interface requirements for the CSCW applications were discussed in some detail. One of the user interface systems that I propose in this work, **"tri window"** is based upon the public and private window displays used on the DI based Micro

Emacs[10]. Communication and network requirements for managing the group work was discussed along with a proposal for a CSCW environment.

# CHAPTER II

# SOFTWARE REQUIREMENT SPECIFICATION

## 2.1 Introduction

A software requirement specification (SRS) document produced by a group consisting of clients and analysts is used as an example work for the purposes of demonstration of this project. The client is the person who requires the software product, and the analyst is the person who develops the software. Requirements "specification" is a precise statement of needs intended to convey the understanding of a desired software system. It describes the external characteristics, or user visible behaviour, of the result as well as constraints such as performance, reliability, safety, and cost. The activities involved in generating an SRS for a large system generally involves participation of many people having various kinds of knowledge. These people normally work with each other and collaborate to define the requirements of the computer system software.

A software requirement specification is normally preceded by a process called "software requirement analysis"[27]. The requirement analysis is done in order to understand the problem which the software system is required to solve. In the design of large systems that provide a large number of features, and that are needed to perform many different tasks, understanding the requirement is a major task. The requirement analysis is a difficult and error prone activity. In the beginning of the analysis the various needs of the system are in the minds of different people in the clients organization The system analyst must identify the requirements by talking to these people and understanding their needs. Hence specifying requirements necessarily involves specifying what some people have in their minds. SRS is a means of translating the ideas in the minds of the clients into a formal document. As the information in the minds of various people is by its very nature not formally understood or organized, the input to the software requirement analysis stage is inherently informal and imprecise, and is likely to be incomplete.

A minimum group environment for defining the system requirement will consist of two individuals, ie., the analyst or the developer and the client. The client is the one who knows what is needed of the system, and in most cases has a minimal knowledge of the system design issues, and the developer usually does not have indepth knowledge of the client's problems This will always result in a communication gap, which has to be adequately bridged during the requirement analysis.

The two major activities in the requirement analysis phase are understanding the requirements and stating them clearly in a document[27]. The task in the analysis phase typically involves in the developer asking questions to the client, and/or end users of the system, and reading various reference documents. It also involves discussions, dialogues review of documents and "walk-throughs" of the specification.

During the problem analysis phase, a massive amount of information is collected in the form of answers to various questions. The answers may be presented in the form of texts, graphs, hand drawn sketches, oral statements, example forms, etc. The main task in the analysis phase in a cooperative development environment phase, will be how to organize this information and how to structure it. The environment must be able to support various types of inputs which can be easily transmitted and shared among the participants. Various pieces of information may be required to be resolved and commented upon by more than one participant The participants may form several sub groups and discuss within their own "caucus" to arrive at certain conclusions During the problem analysis phase contradictions may arise due to different information sources this may have to be resolved during the "meeting" process This "meeting" process requires that the support environment provide tools to alarm the participants to inform one another of severe conflicts that occur and draw everyone's attention to such an event The cooperative environment must be able to support a private work space for the participants during the discussions, for them to take some notes or details to be analyzed subsequently. The software developer is required to understand the problem the client addresses and its context.

Once the problem is analyzed and the essentials are understood, the software requirements will be specified in the requirement specification document The inputs used to generate the requirements specification can come from many sources and cannot be totally formal. For the purpose of this project we will assume that the requirement specification will be summarized in a natural language (English) Tables and formal expressions may also be used. The requirement specification will specify all functional and performance requirements, the formats of all inputs and outputs and all design constraints that exist due to various reasons. The requirement specification thus generated will normally be validated by the client to make sure that the requirement actually reflects his true needs. The validation phase is normally done in a group environment with representations from the client, and system designers.

The goals of a good SRS are as follows :

(i). Establish the basis for agreement between the client and analyst on what the software product will do.

(ii). Reduce the development cost.

The production of SRS usually involves a detailed and rigorous discussion on the system design. Since SRS is normally produced by people involved with various aspects of the project, it will reveal omissions, inconsistencies and misunderstandings fairly early in the project, which can considerably reduce the cost.

(iii). Provide means for the generation of the system acceptance procedures and the software development test plan.

## . *Requirement Formulation*

"Information transfer difficulties" occur as users attempt to describe the system requirements and problems to analysts. The most serious of these difficulties relates to misinterpretation of user requirements. The problems associated with the transfer and transformation of requirements information may be classified into four broad categories. **Acquisition, representation, content, and analysis.**[28]

These difficulties account for approximately 80%[36] of the errors found on the delivered software. This can be overcome to some extent by a) developing an object management system that facilitates integration of diverse types of information to be utilized during the analysis process; and b) a means of developing reusable requirements information, and a set of knowledge based tools, for increasing the overall robustness of requirements information.

## . *Acquisition difficulties*

In situations where the problem is highly complex, the user generally is not able to accurately state the requirements of the software product. Additionally, the user and the analyst are frequently not able to communicate adequately concerning the system requirements. Primarily, there are two reasons that combine to keep the analyst from obtaining satisfactory requirements information.

First, there are situations that occur concerning complex and/or analytical problem domains for which the user is not able to accurately state requirements for the system.

Secondly, situations occur where users are unable to articulate the requirements in a language that the analyst can understand. As a consequence, information is lost, inaccurately represented or otherwise made less useful. The analyst often has a similar inability to relate to the language of the user and few analysts are trained in areas capable of providing assistance such as: management science, systems engineering, and cognitive psychology.

### . *Representation difficulties*

It has been noted that an implicit model formed by the analyst contributes to understanding of requirements information. A main problem faced is how to obtain robust representations of requirements information, for direct examination during the analysis process. This process, the transfer, transformation, and recording of requirements, is currently accomplished as a manual task. Information must be transformed from its actual representation: dynamic, real world, and multimedia, into computer oriented text and graphics. This transformation could possibly lead to misunderstandings, filtering, omission, and processing errors, all combining to reduce the utility of the information.

### . *Content difficulties*

Requirements information provided by users is often characterized by statements that are ambiguous, inconsistent, conflicting or possess other similar flaws. Software containing these flaws is at risk from the onset, and seldom, if ever, results in the desired outcomes for the software product. Performing manual examination of thousands of individual requirements for these flaws is nearly an impossible task and is often beyond human ability to perform. Current requirement elicitation methodologies do not appear to address these issues. The consequence of this is a lack of support for processes to examine requirements information when checking for flaws in the content

### . *Analysis difficulties*

System requirements are generally understood to become more reliable, accurate, and trustworthy as development proceeds. However, current methodologies discourage analysts from further enhancing written requirements, because they are not able to accommodate unplanned changes or assess the impact of requirements volatility Also inability to analyze requirements information in "forms" and "graphic formats" limits the available bandwidth for information analysis over a discussion over the telephone line A graphic "flow" or "process diagram", or sensor calibration data in a standard form, or a control diagram in a loop schematic diagram convey more information to the designer than the descriptions of these items in standard text form.

## 2.2 Analyzing the problem

Analysis is the systematic process of reasoning about a problem and its constituent parts to understand what is needed and what must be done Analysis also involves communicating with many people, and thus provides an ideal case study for a CSCW application

The requirements engineering environment must support analysis in several ways[20]. First the reasoning process must be guided by an analysis methodology appropriate to the problem. The environment should enforce the procedures of the methodology and facilitate its application through an appropriate supporting work place. The work place must provide the requirement engineer with the software tools needed to gather the information necessary to reason about and understand the problem domain. Such tools must include rigorous, but natural ways to describe models of the real world problem domain.

Secondly, the information must be organized to permit quick locating and easy access to pertinent facts. An environment that facilitates locating and accessing is necessary for stimulating the insights that produce requirements. A multimedia supported computer system could support such an environment. The communication between the requirement engineer and those in the community of interest, must take place at scheduled meetings as well as in spontaneous meetings. The environment must also support the presentation and discussion activities by providing a communication network, a work place to facilitate its easy and rapid access, and the audio visual tools.

Structuring of information during the analysis phase is essential in order to develop a good SRS. A **partitioning** principle is commonly used to understand large and complex systems[27]. Partitioning captures the "whole part of" relationship between entities. It is used when a large system can be easily described in terms of its parts. Each of the parts partitioned can be described further in detail subsequently.

**Abstraction** techniques may be utilized to describe a problem or system in general terms, and the details will be provided separately [27]. This technique provides the analyst with a tool to consider a problem at a suitable level of abstraction, where he can comprehend the general system and then analyze the details further.

Another useful technique in the problem analysis is the **projection** method[27]. Projection defines the system from multiple points of view from different perspectives. The problem is studied from different perspectives, and then the different projections obtained are combined to form the analysis for the complete system. The main advantage of projection is that trying to describe the system from a global view point is difficult and error prone, and it is more likely that the participants can forget main features of the system.

### . *Documenting the requirements*

While analyzing the user needs, the system developer has to document the different types of information because the requirements must capture and convey the overall scope of the problem, the semantics of its important objects and activities then relationships and their connections with the problem domain. The documents must be presented from different perspectives and with different audiences in mind. The environment must provide for presentations of several types of convenient forms for display. A simple, easy to use pictorial scheme would be an ideal notation and presentation medium for the requirements engineer. A formal notation that solidifies the meanings of the actions in terms of what happens to the data entities is also necessary. During the presentation phase, it must be possible to write a natural language statement of the problem that is based upon the formal representation. It is also required that all the documentations must be readily accessible for review and modification by the requirements engineer and by engineers involved in the post requirements activities.

### . *Data flow diagrams*

Data flow diagrams or flow charts are commonly used during the problem analysis phase[27]. A sample data flow diagram is shown in figure 2.1. Flow charts are very useful in understanding a system and can be very effectively used for partitioning during the analysis. A data flow diagram shows the movement of data through different transformations or processes in the system. There is no formal procedure that can be used to draw a data flow diagram for a given problem. One way to construct a data flow diagram is to start by identifying the major inputs and major outputs. A single data flow diagram may be too small to describe the data flow in a large system. It is necessary that some abstraction mechanisms must be used in analyzing large systems. Data flow diagrams can be hierarchically organized, which helps in progressively partitioning and analyzing large systems. Such a hierarchical data flow diagram consists of a starting data flow diagram, which is a very abstract representation of the system and which identifies the major processes in the system. Then each process is successively refined and a data flow diagram is drawn for each stage of refinement

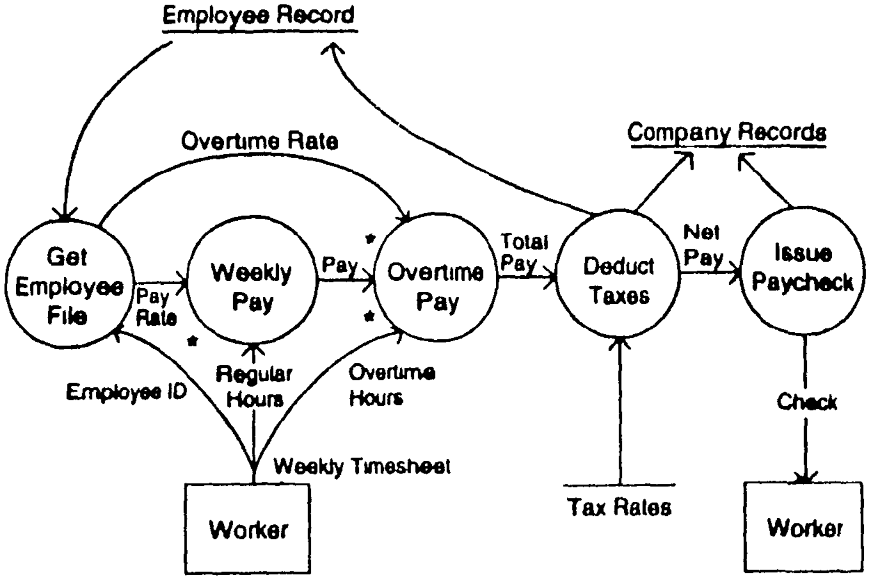


Figure 2.1 - A sample DFD for a pay-roll system[27]

Once the data flow diagrams are constructed, analyzed and refined they will be verified by "human processing" and rules of thumb. The data flow diagrams are walked through with the client, to detect any errors. The "walk through" process will verify the common errors like inconsistencies, missing information and missing processes. The data flow diagrams must be carefully scrutinized to make sure that all the processes in the physical environment are indicated in the diagram.

### . *Structured analysis*

The structured analysis method, which is a top down analysis method that relies heavily on data flow diagrams, is utilized in the analysis and production of the software requirement specification[27]. In this method all the functions that are necessary to solve a particular problem are considered. This method helps the analyst organize the information better and prevent overloading of information.

For an existing system which is required to be automated, the structured analysis process is accomplished by the following steps.

(i). The data flow diagram of the current existing system is drawn, indicating all the input and output data flows and all the processes operating on the data in the system. The next step invoives in the drawing of the logical equivalents for the DFDs of the physical system. All specific physical data flows will be represented by their logical equivalents The physical processes will be replaced by equivalent logical processes.

In the development of each of these transformations, the transformations will be verified with the client.

(ii). Once the current system is completely transformed into its logical equivalent, a complete DFD is drawn for the new system. This will illustrate how the data flow occurs in the new system. In this step, the analyst only expresses to the client as to what needs to be done, not how it will be accomplished. The next step involves in the analysis of the new DFD model and determining what needs to be modified and what needs to be retained as before. A man-machine boundary will be established for the new model which will specify as to which of the processes will be automated and which will remain as manual. Even if some of the processes remain as manual, they may operate differently from the original system.

(iii). Analysis of the different options for achieving the desired objectives and developing or presenting the specifications will be done next.

Structured analysis provides methods for providing and representing information about the existing system. It provides useful techniques for understanding and analyzing the existing system. Structured analysis does not offer much help for analyzing the target system and constructing the data dictionary for the new system to be built. It must be noted however that the study of the existing system will help develop the new system in a more systematic way.

### . *Analysis by prototyping*

Problem analysis by prototyping helps clients and users gain better understanding of their needs, particularly when the system is new[27]. In this technique a partial system is constructed, which is often used by the client, users and developers to gain better understanding of the problem. The clients can assess their requirement much better if they can see the working of the system beforehand. As is the case in many instances, practical experience is the best aid for understanding needs, since it is often difficult to visualize a working system. A prototype is a partial system, and it cannot be a model of the complete final system. In the prototyping analysis method two approaches can be used.,

A **throwaway** approach is an idea which will be used in developing a prototype and which will be discarded after the analysis is complete, and the final system will be built from scratch. In this approach the sequence of events that are executed will be to develop a preliminary SRS for the system which will be an SRS for the prototype, build the prototype, evaluate the client user experience with the prototype, develop the final SRS for the system and develop the final system.

In the **evolutionary** approach for prototyping, a prototype will be designed and built with the idea that it will eventually be converted as the final system. This technique results in the development of the iterative enhancement model. As clients and users interact with the prototype model, improvements and modifications are included in the system as the development and analysis progress.

Developing a prototype involves determining what aspects of the system need to be included in the prototype. Developing an SRS for the prototype is application dependent. The prototype is developed in much the same way as any software is developed, with a basic difference in the development approach. The final SRS is developed in the same way as any SRS is developed. The advantage here is that the clients and users will be able to answer questions and explain their needs much better through their experience with the prototype.

Prototyping is often not used in the development phase, because of the fear that the development costs may become much higher. However, in some situations the cost of software development without prototyping may be much higher than with prototyping. This is because the experience gained in prototyping will result in reduced costs of the development of the later stages of the software. Also, in many software system projects the requirements constantly keep changing, particularly when the developments take a very long time. Also, since the clients and users actually gain experience with the prototype, the SRS developed after the prototype will be closer to the actual requirement. This will result in fewer changes in the requirements at a later time

## 2.3 Requirement specification languages

Specific languages are used for specifying SRS and during problem analysis

These languages provide means for organizing and specifying information as well as producing reports. The languages do not provide much help in determining the requirements The languages help in writing the requirements in a manner that is useful in later activities of the development of the software system.

Although natural language like English, provides an easily understandable medium for the SRS, it usually results in ambiguities. On the contrary, the formal languages are not well accepted in industry. A main reason for this is the extensive use of mathematical symbols, their strict semantic interpretations and the terse textual nature that are employed in the formal specification languages. These do not provide a good medium of communication among software practitioners. A formal specification language however, provides conciseness, precision and a mathematical basis for automated program synthesis. There are semiformal specification methods available, which can be utilized to obtain formal specification techniques either interactively or off line[26]. The semiformal methods bridge the gap between formal methods and informal methods of specification.

## . *Structured Analysis and Design Technique*

Structured Analysis and Design Technique(SADT) utilizes a graphical language that is based upon the concept of blueprints used in the engineering design[22]. A model in the SADT is a hierarchy of diagrams supporting a top-down approach for analysis and specification. A typical diagram used in the SADT is shown in figure 2.2.

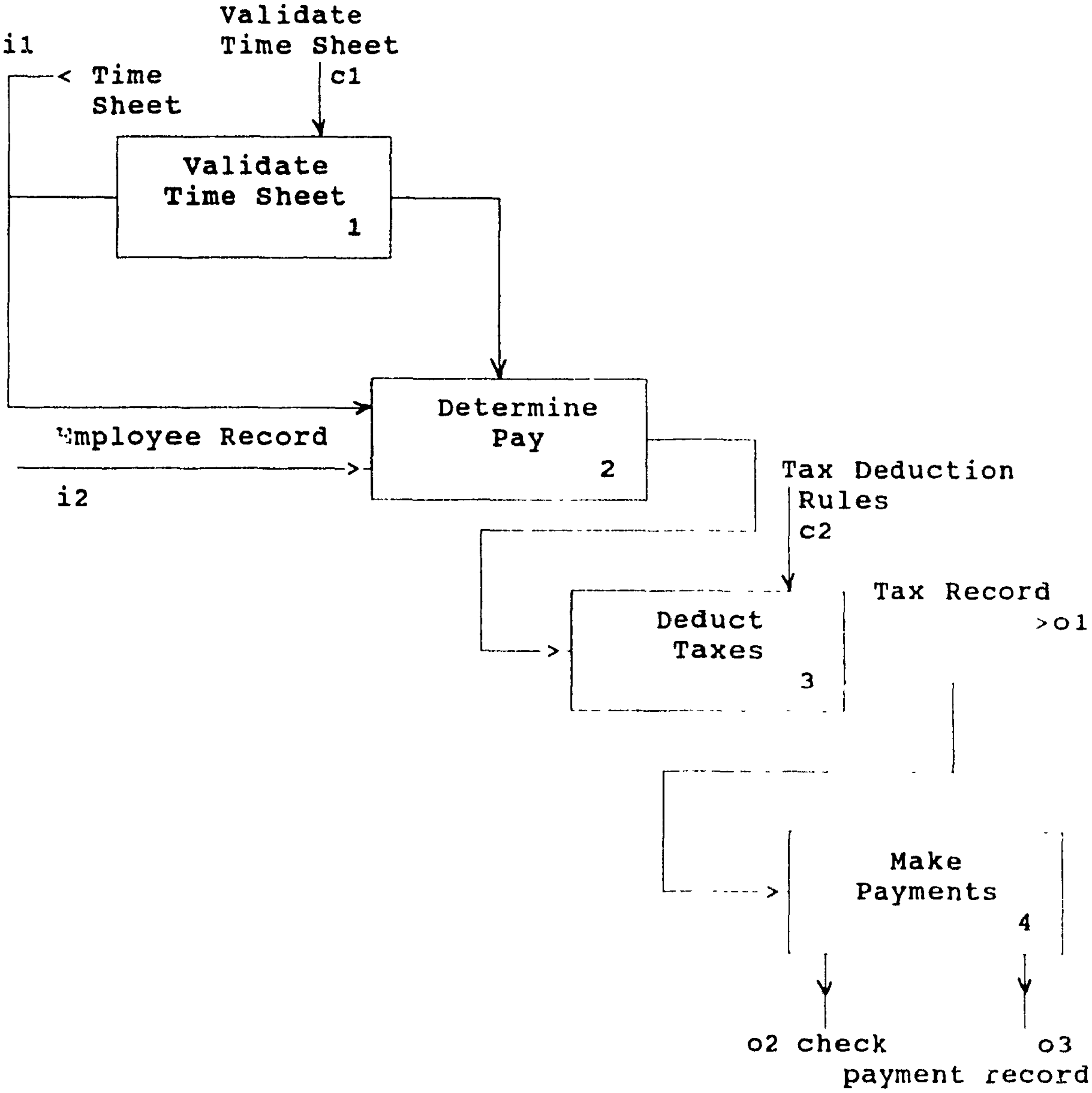


Figure 2.2 - A diagram for SADT

The diagram includes text written in natural language. The SADT stems from a single premise : The human mind can accommodate any amount of complexity as long as it is presented in "easy-to-grasp chunks", that together make the whole.

SADT is a powerful methodology for working out a clear-cut understanding of an initially obscure, complex subject, documenting that understanding, and then communicating it to others SADT is a general methodology. In principle, SADT can be used for any kind of problem. In practice it is thought of primarily as a requirement definition methodology that interfaces well to other design and implementation methodologies

SADT consist of two principal parts : (1) the box-and-arrow diagramming language of structured analysis and (2) the design technique, which is the discipline of thought and action that must be learned and practised if the language is to be used effectively. The basic components of SADT diagram are context boxes and arrows entering or leaving the boxes. Boxes represent part of the whole, and arrows represent the interface between the different boxes. The box used in the representation has its four sides representing input, output, control and mechanism to achieve the transformation. A sample structured analysis context box is shown in figure 2.3.

Input, control, and output are the interfaces between the parts as they compose the whole and are completely different from the mechanism. These interfaces are indicated by branching arrows that connect outputs to inputs or controls, so that results of one transformation can be further transformed by another box or can control the transformation of some other input by another box. Thus SADT can be applied to analyze or design any system composed of objects and associated actions upon those objects.

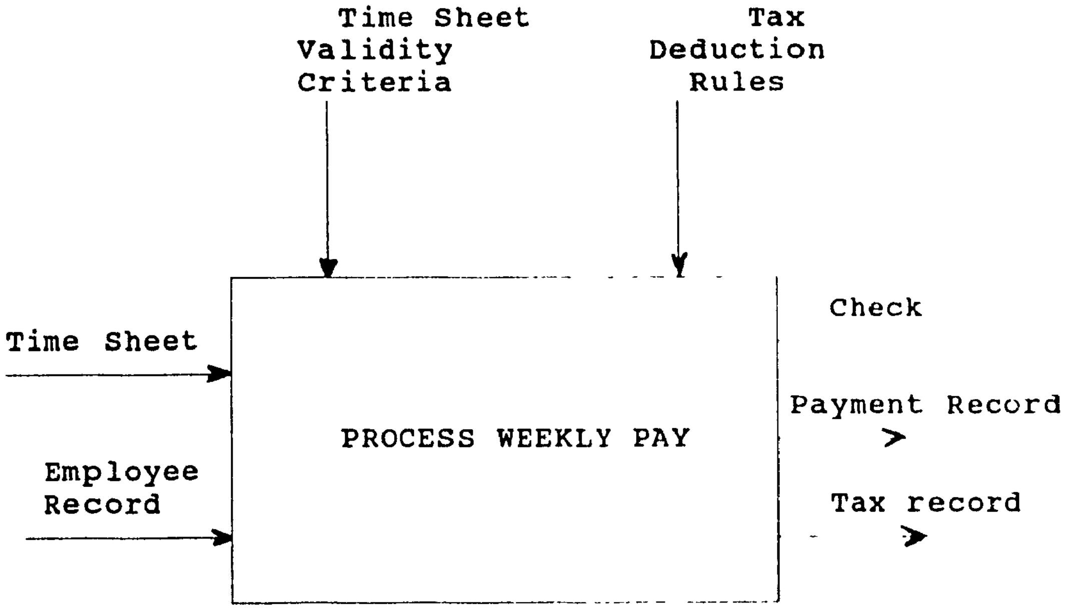


Figure 2.3 - The structured analysis box

### . *Requirements Statements Language (RSL)*

Requirements Statements Language is based on a set of elements, attributes, relationships and structures[23]. The elements are its nouns, the relationships its verbs, and the attributes its objectives. The structures describe the graph of processing requirements. RSL utilizes 21 different kinds of elements, such as the R-nets, ALPHAs, state data and messages to represent the features of the requirements model.

RSL was specifically designed for specifying complex real time control systems. The RSL uses a flow-oriented approach for specifying real time systems. For each flow the stimulus and the final response are specified. There is also control information specified in RSL, since in a real time control system, control information is an essential component for completely describing the system.

RSL is not limited to the specification of processing: it can express other concepts, such as traceability as well. RSL serves two overall purposes: It provides a checklist of characteristics for requirements specification, and it puts the requirements into a machine-readable form that can be translated into a database for automated consistency and completeness analyses.

### . *The Problem Statement Language (PSL)*

The Problem Statement Language(PSL) is designed for specifying the requirements of information systems[27]. It is a textual language. In PSL, a system consists of a set of objects, where each object has properties and defined relationships. The major parts of the system description are the system input output flow, system structure and data structure. The system input output flow describes the interaction of the system with the environment and specifies all the inputs received and all the outputs produced.

The system structure defines the hierarchies among the different objects and the data structure defines the relationships among the data that are used within the system

The PSL method is fundamentally the same as the structured analysis method A PSL description also specifies the existing procedures used in the process and the dependencies between an output and different inputs

The Problem Statement Analyzer is the processor that processes the requirements stated in the PSL and produces some useful reports and analysis

### . *Input/Output Requirements Language (IORL)*

Input/output requirement language is the language element utilized in the Technology for the Automated Generation of System (TAGS) paradigm[24] The main objectives in the development of IORL are.,

. to enforce a rigorous methodology for system development.

. to be applicable to not only computer systems but all systems.

. to be easy to use;

. to allow engineers to express system performance characteristics and algorithms using common mathematical notation,

. to use graphical symbols derived from general systems theory.

IORL is a graphical and tabular specification language. It consists of schematic block diagram(SBD) at its highest level. A representation of the IOR is shown in figure 2.4.

SBDs are rectangular boxes that identify all the principal system components and the data interfaces that connect them. The top level SBD can be broken down into lower level SBDs until the resulting SBDs can no longer be broken down. The SBD components are further described by an input/output relationship and timing diagram (IORTD), which show the control flow within the SBD. The predefined-process diagram or PPD is used to depict the detailed logic flow of a single predefined process referenced in an IORTD or another PPD. PPDs are used to improve the readability of the specification, to allow the identification of the dependent components and to permit the specification to be presented in a hierarchical manner.

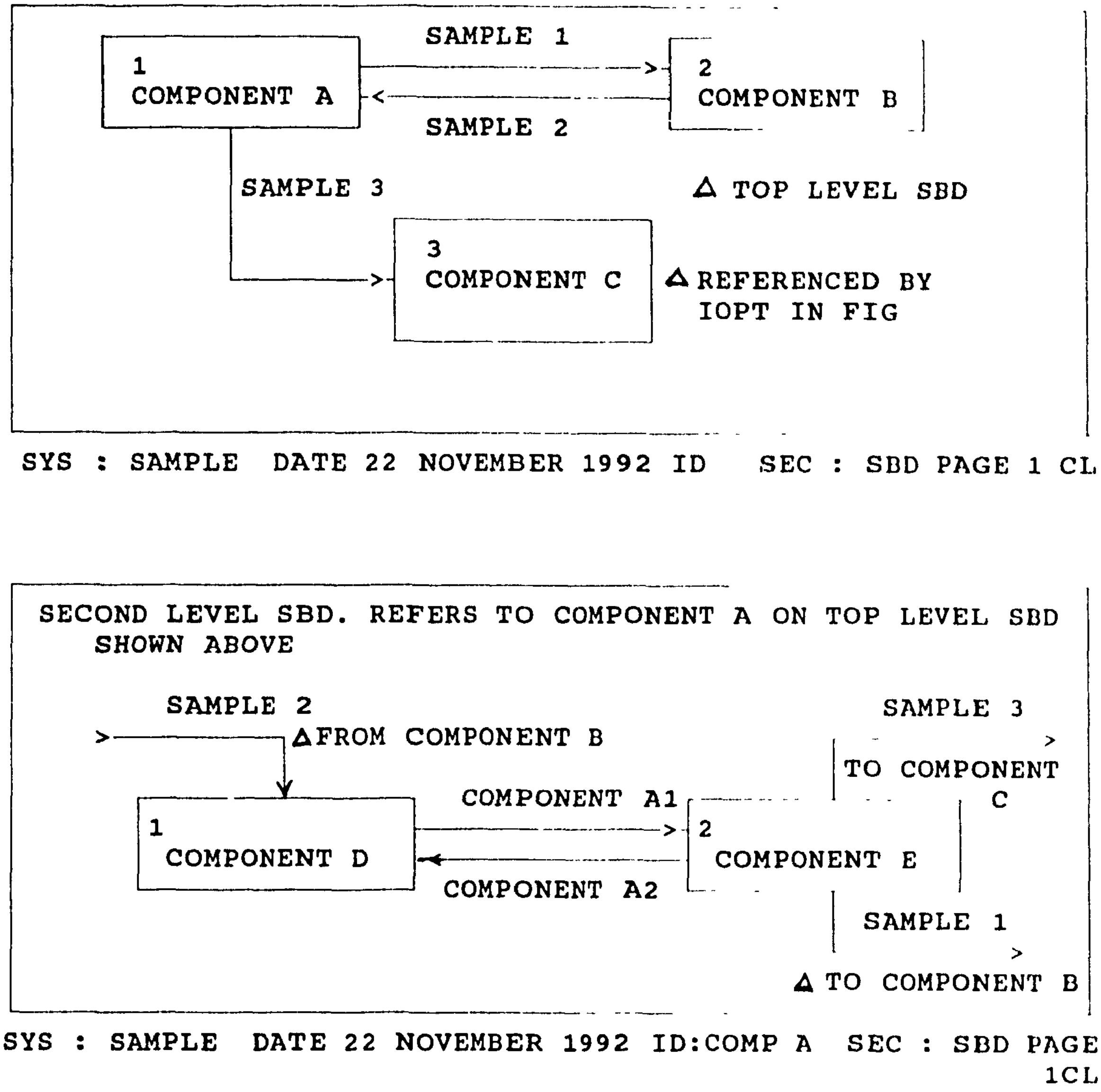


Figure 2.4 - IORL schematic block diagram representation

## 2.4 S.R.S.Document

The production of the software requirement specification (SRS) document follows the problem analysis phase. The software requirement specification document transforms the user intended requirements in to an easily comprehensible document. The software requirement specification document may also be done concurrently during the analysis phase. The SRS document specifies the engineering and qualification requirements for a Computer Software Configuration Item (CSCI). The main objective of the SRS document is that it provides a common means of agreement between the client, the users and the developers[25]. In order to avoid ambiguities, the software requirement specification may be written in some form of formal specification language. Informal descriptions are known to have the potential to contain ambiguities, partial descriptions, inconsistencies, incompleteness and poor ordering of requirements. In some cases, a formal language like VDM is considered for the production of an SRS.

The SRS must identify all the tasks that the CSCI must do, in order to achieve the required objectives. The requirements relating to functionality, performance, design constraints, attributes and external interfaces must be specified in the SRS. The SRS document must also ensure that all the specified requirements are verifiable. Requirements validation depends largely on the techniques used for specification. A Software Test Plan which will verify all the required objectives, must be producible from the software requirement specification document.

Writing an SRS is an iterative process. Even when the requirements of a system are well specified, they are later modified as the needs of the client change with time Hence, the SRS must be easy to modify. An SRS is easily modifiable if its structure and style is such that any necessary change can be made easily while preserving the completeness and consistency.

## . *Components of an SRS*

The main components of an SRS are the **functionality**, **performance**, **design constraints** and **external interfaces** [21,27]. These components are illustrated in figure 2.5.

### . *Functionality*

The functional requirements of an SRS specify the relationship between the input and output of the system. For each functional requirement, a detailed description of all the data inputs and their source, the units of measure and the range of valid inputs must be specified.

The functional aspects must specify the validity checks on the input and output data, parameters affected by the operation, and equations and other logical operations that must be used to transform the inputs into corresponding outputs. The validity checks also must specify the system behaviour in abnormal situations, such as invalid input or error during computational operations A good SRS will specify the system behaviour for all foreseen inputs and for all foreseen system states.

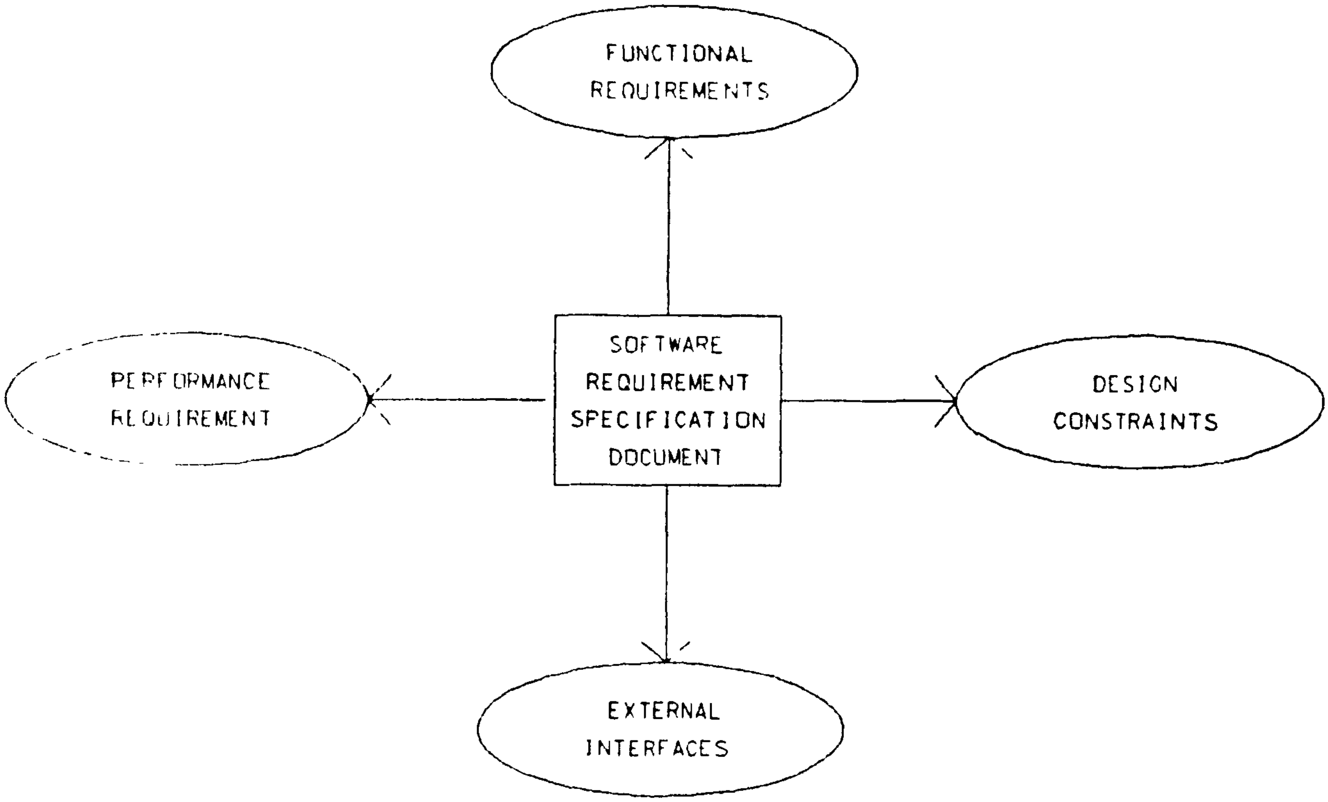


Figure 2.5 - Major components of an SRS

. *Performance requirements*

All the requirements relating to the performance characteristics of the system must be clearly specified in an SRS. The performance characteristics are static or dynamic. Static requirements do not impose constraints on the execution characteristics of the system. The dynamic characteristics will specify the execution behaviour of the system, such as the response time, throughput constraints and worst case operating conditions. The dynamic behaviour of the system must be specified in measurable terms, which can be easily verified under a software test plan.

. *Design Constraints*

The SRS must specify the design constraints that may restrict the designer in choosing the best method to implement the requirements. These may include compliance to a standard, hardware limitations, fault tolerances, reliability aspects, mean time to repair and security aspects. Hardware limitations may include the necessity of the software to operate on a previously designed or selected hardware, or a hardware that cannot support all the software functions. The security constraints are most significant on defence contracts where use of certain commands may be restricted, and access control mechanisms that the Computer Software Configuration Item (CSCI) must provide, etc.

. *External Interface requirements*

External interface requirements specify all the possible interactions of the CSCI with people, external hardware, and other CSCIs. The characteristics of each user interface must be clearly specified. A preliminary user manual with all user commands, screen formats, error messages must accompany the SRS. The SRS should specify the logical characteristics of each interface between the software product and the hardware components. The CPU usage and the memory capacity, both spare and used, and the spare hardware available should also be specified in the SRS. The interface of the CSCI with other software components like the operating system, or other CSCI must also be specified in the SRS.

## 2.5 Structure of an SRS

For our work, we use an SRS structure which is as specified in the well known DOD standard 2167A format[30]. This specifies the SRS to be documented in sections and subsections. The following paragraphs describe the various sections and subsections.

. Cover

. Title page

. Table of contents

. Scope

. Applicable documents

. Engineering requirements

. Qualification requirements

. Preparation for delivery

. Notes

. Appendix

## . *Title page*

The title page contains the CSCI name, the contract number, CDRL sequence number, preparer's name and signature blocks for authentication and approval with corresponding dates.

## . *Scope*

This section contains an identification block, an overview block for CSCI, and a subsection for Document overview. The identification paragraph contains the approved identification number, abbreviation of the system and the CSCI to which this SRS applies. The CSCI overview paragraph contains a brief explanation of the purpose of the system, and identifies and describes the role, within the system, of the CSCI to which the SRS applies. The document overview paragraph summarizes the purpose and the contents of the SRS.

## . *Applicable documents*

The applicable documents section lists all the reference documents that are used in conjunction with this SRS. The section is further divided into government documents. Specifications referred to, standards, drawings and other publications. The non government documents will include the client specifications, standards, drawings, and other publications.

### . *Engineering requirements*

This sectior is divided into the following paragraphs and subparagraphs to specify the engineering requirements necessary to ensure proper development of the CSCI. The requirements to be included herein are allocated or derived from requirements established by the applicable system specification and other references. The following subsection is included in this paragraph.

. **CSCI external interface requirements** - This subsection identifies the external interfaces of the CSCI. A diagram of the type shown in figure 2.6 may be used to aid in this description.

Each external interface is identified by name and project unique identifier and a brief description of each identifier is provided. Any identifying documentation such as an interface control document or interface requirements specification is referenced for each interface.

. **CSCI capability requirements** - This section identifies all of the capability requirements that the CSCI must satisfy. The CSCI capability is identified by name and project unique identifier and states the purpose of the capability and its performance in measurable terms. It identifies and states the purpose of each input and output associated with the capability. It also identifies the allocated or derived requirements that the capability satisfies or partially satisfies. If the capability can be more clearly specified by decomposing it into constituent capabilities, the requirements for each constituent capability are provided as one or more subparagraphs.

## . *CSCI internal interfaces*

This section identifies the interfaces between the capabilities identified above. Each internal interface is identified by name and project unique identifier and a brief description of each interface is provided, including a summary of the information transmitted over the interface. Internal interface diagrams depicting data flow, control flow, and other relevant information may be used to aid in this description.

### . *CSCI data element requirements*

This section identifies data elements internal to the CSCI and data elements of the CSCI's external interfaces. Each of the internal data element has a project unique identifier, a brief description, the units of measurement, the limit values associated with the measurement, the accuracy requirement, and the resolution requirement. For data elements of the CSCI's internal interfaces, the source capability of the data element and the destination capability of the data element are defined.

The data elements of the CSCI's external interfaces are identified by a project unique identifier, its source or destination capability as applicable and the reference to the Interface Requirement Specification in which the interface is specified

## . *Adaptation requirements*

This section is subdivided into the following subparagraphs to specify the requirements for adapting the CSCI to site-unique conditions and to changes in the system environment.

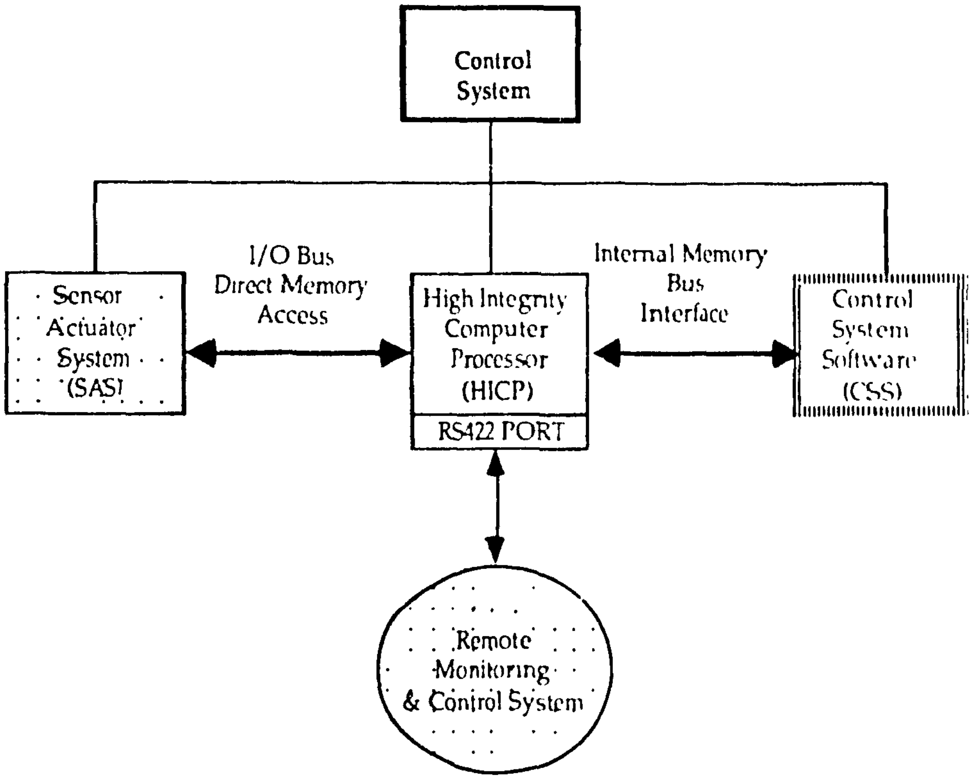

Control System
I/O Bus Direct Memory Access
Internal Memory Bus Interface
Sensor Actuator System (SAS)
High Integrity Computer Processor (HICP)
RS422 PORT
Control System Software (CSS)
Remote Monitoring & Control System

**Figure 2.6 - Example external interface diagram**

. **Installation dependent data** - This subparagraph describes the site unique data required by each installation. This subparagraph also identifies the CSCI capabilities in which these data are used.

. **Operational parameters** - This subparagraph describes the parameters required by the CSCI that may vary within a specified range according to operational needs. It also identifies the CSCI capabilities in which these data are used.

. **Sizing and timing requirements** - This subparagraph specifies the amount and, if applicable, location of internal and auxiliary memory and the amount of processing time allocated to the CSCI. It shall also specify the resources required of both memory and the central processing unit (CPU) for the CSCI.

. *Safety requirements*

This paragraph specifies safety requirements that are applicable to the design of the CSCI with respect to potential hazards to personnel property, and physical environment.

. *Security requirements*

This section specifies the security requirements that are applicable to the design of the CSCI, with respect to potential compromise of the sensitive data.

### . *Design constraints*

This section specifies other requirements that constrain the CSCI design, such as the use of a particular processing configuration, etc.

### . *Software quality factors*

This section specifies each software quality factor identified in the contract or derived from a higher level specification. For each quality factor required a method of compliance is specified along with the requirements for that factor.

### . *Human engineering requirements*

This section specifies the applicable human factors engineering requirements for the CSCI. These requirements include human information processing capabilities and limitations, foreseeable human errors under both normal and extreme conditions, and implications for the total system environment.

### . *Requirements traceability*

This section contains a mapping of the engineering requirements in this specification to the requirements applicable to this CSCI to the other specifications referenced.

. *Qualification methods*

This paragraph specifies the qualification methods and any special qualification requirements necessary to establish that the CSCI satisfies the requirements referenced earlier. The qualification methods can be established by similarity, analysis, demonstration or by inspection.

## 2.6 Summary

Requirements engineering for large complex systems is seldom performed by one or two individuals setting about their work with documents and interviews to generate requirements. Rather, it is always based on work performed by groups or teams who have particular assignments and utilize information from any media format that may be available to assist them. This may include documents, interviews, video tapes, viewing the system to be modified and various combinations of these. A multimedia work station environment in support of this approach to requirements engineering will be required as a means of providing support for computer supported cooperative work.

I have described many developments in software requirements specification methods. Some of the techniques that are utilized in the development of the SRS are described in this chapter. I have described in brief some of the formal specification languages utilized in the SRS development process. The main components of the SRS that I plan to utilize in my project work as a case study is also described in this section. Although the issues covered in the development of the SRS are well known, the main reason in highlighting this is to indicate that the development of an SRS is essentially a Cooperative task, which can be easily supported by a multimedia work station

# CHAPTER III

# CSCW CASE STUDY

## 3.1 Introduction

Production of a Software Requirement Specification document in a cooperative environment is described in this section. A real time control system intended for the control of a gas turbine engine is utilized as a case study for this purpose. The control requirements for the system are described in English text in conjunction with graphical notations. The Software Requirement Specification document is produced in accordance with the DOD standard 2167A format[30]. The groupware and user interface for the cooperative work is provided by Groupware Tele Communication Software (GTCS)[34], which facilitates cooperative editing and group data management. The implementation is demonstrated for a cooperative session consisting of two participants, who will consist of the client and the analyst. Each of the participants has the technical requirement specification document produced by the client for referencing and discussions during the collaboration.

## 3.2 A Case Study

Requirements engineering for large systems is performed by a group of people, who utilize information from various media to assist them. A software development process such as software requirements engineering, provides an ideal case study for a CSCW implementation and it satisfies the main criteria required for group work as described below :

. *Addressability of common topics -*

The Software requirements engineering is a group or team work, consisting of two distinct groups. i.e., the client group and the developers. Both groups will be addressing a common topic but with a different perspective. The client's knowledge or interest will mainly involve "what is to be done" and the developer will be concentrating on "how it is to be done", both addressing the same topic. The group work will involve a shared screen conferencing, with both the client and the analyst viewing the identical displays with a different perspective.

. *interactive dialogue -*

Interactive dialogues will be an essential feature in the software requirement definition process. Dialogues will always result in better integration of the different knowledge domains contained with the client and the developer.

. *Multiple participants -*

The group work may involve more than two members. Large system designs require many experts, to clearly state and specify requirements. The throughput of a session may however, vary depending upon the number of participants, as it may be difficult to arrive at a consensus on some of the conflicting issues.

. *Decision making -*

Decisions in a software requirement definition could involve voting, as there will be conflicting choices. Some of the decisions will be constraint driven, and some will be cost driven. In a situation like this, decisions may be made based upon a voting process or through consensus.

. ***Asynchronous and real-time interactions -***

Participants in the requirement definition process may vary from time to time, based upon the subject being discussed. The group may at times consist of a large number of participants or a few (a minimum of two) as the case may be.

## *Case Study Description*

The case study utilized for our project involves the design of some computer control system software for a real time control application. The controller design requirement is based on a true real time control application, the details of which cannot be published here, due to the confidential nature of the project. The real project involves a much larger requirement definition, and only some of the more generic control requirements are expressed as a subject for the software requirement specification for our case study. A gas turbine engine, which is utilized for power generation, is to be controlled utilizing a computer system. The gas turbine engine requires a complex propulsion control, torque limiting, and fuel control algorithms. The control system software will reside inside a High Integrity Computer Processor and utilize a high level computer language to meet the requirements.

***CLIENT :*** Gas Turbine Engine Manufacturer.

***END USER :*** Power plants, Ship and Aircraft Manufacturers.

The control system is req'ired to provide the following features to control the gas turbine generator :

. *control of fuel supply to the engine,*

. *control of starter motor and igniters for automatic start,*

. *control of turbine steady state and transient performance,*

. *protection of turbine under overspeed conditions,*

. *turbine emergency and normal stop control,*

. *a person machine interface to control the turbine,*

. *monitoring of turbine critical parameters by the operator,*

. *an interface to a remote computer, which will be used for engine health monitoring and supervisory control purposes.*

. ***Fuel Control***

Fuel control to the turbine is accomplished by two fuel control valves. The two fuel control valves are connected in series, and both valves must be energized to apply fuel to the gas turbine combustion section. The valves must be shut by the controller whenever the "turbine stop" command is invoked or whenever the turbine temperature, or speed has reached excessive limits. Independent control of each valve is provided to allow for periodic alternate de-energization of each valve, and to initiate shutdown and verify proper operation of the valve.

### . *Start Control*

The starter incorporates an output shaft shear section, which provides protection against overtorquing of the gas turbine gear box. The controller provides two start control modes, automatic and manual. The start mode in effect is selected by the ‑emote computer or the local person machine interface.

#### . *automatic start*

The controller begins an automatic start sequence when:

- The turbine can be safely started,
- Automatic start mode is in effect,
- A normal automatic start command i' received.

The Automatic start sequence is executed as follows:

- Starter regulator/shutoff valve is energized admitting air to the pneumatic starter.
- Applicable output signals are generated.
- Ignition units are energized after a time delay.
- Power is applied to the fuel valves and fuel is admitted to the gas turbine combustion unit.
- Fail to ignite time delay function is initiated.
- The gas turbine speed monitoring system senses that gas turbine has attained required speed.

- Starter air shut off valve is de-energized, shutting off air to the pneumatic starter
- Ignition units are de-energized.

. ***manual start***

The controller provides for manual start of the turbine when the manual start mode is in effect. The manual start commands shall control each discrete step (e.g., energize starter motor, turn on igniters, open fuel valves) in the engine start sequence, and responding to power commands from the remote monitoring and control system or from the local person machine interface.

. ***Steady state and transient performance***

The steady state and transient performance of the turbine includes the following

- Starting and acceleration to idle, with a closed loop acceleration schedule

- Provide a primary control mode which utilizes a closed loop control of torque A back up mode based upon engine speed is provided in the event that the controller is unable to control in primary mode.

- Provide protection to the turbine under high entry temperature, pressure, and turbine speed.

. *Overspeed protection*

The Gas turbine provides monitoring of the gas turbine speed by use of speed sensors. During the normal operation of the Turbine, the controller monitors the Turbine speed signal and closes the fuel supply valves on detection of overspeed signal. Also in the event of a broken shaft the controller prevents the turbine from reaching overspeed condition.

. *Emergency and Normal stop control*

The controller provides emergency and normal stop control for the turbine.

. *Person Machine Interface*

The person machine interface to the controller is provided by a menu driven CRT graphic display. The interface facilitates operation of the turbine from an operator via peripherals such as pushbuttons and joystick.

. *Turbine Parameter Monitoring*

The controller scans and displays the various sensors connected to the turbine. The iteration time is 250 msec.

. *Interface to a Remote Computer*

The Controller provides a serial interface to a remote computer. The serial interface is asynchronous at 38Kbps.

## 3.3 Case Study Implementation

A minimum configuration for this CSCW implementation consists of two computer workstations, which are interconnected by a null modem communication link. The computers are equipped with multimedia software, and provide utilities to manage the joint development process. The following sections describe the CSCW environment utilized for this project.

### . *Hardware*

Two personal computers are utilized for demonstrating the feasibility of this project implementation. Each computer is 80486 based, with a colour monitor, a hard disk, keyboard and mouse. Each of the computers has an additional serial port, which will provide the null modem link with the other. Since the participation for this implementation is limited to only two persons, a null modem link would be adequate for this project. The computers are also interfaced to a document scanner, and a laser jet printer.

### . *Software*

A multimedia, multipoint visual conferencing system software is utilized in the personal computers to provide the CSCW environment. The software provides a multiple window display screen, with tools to manipulate group work activity For this project, a multimedia software system, the Groupware TeleCommunications Software (GTCS)[34] is installed on both the computers.

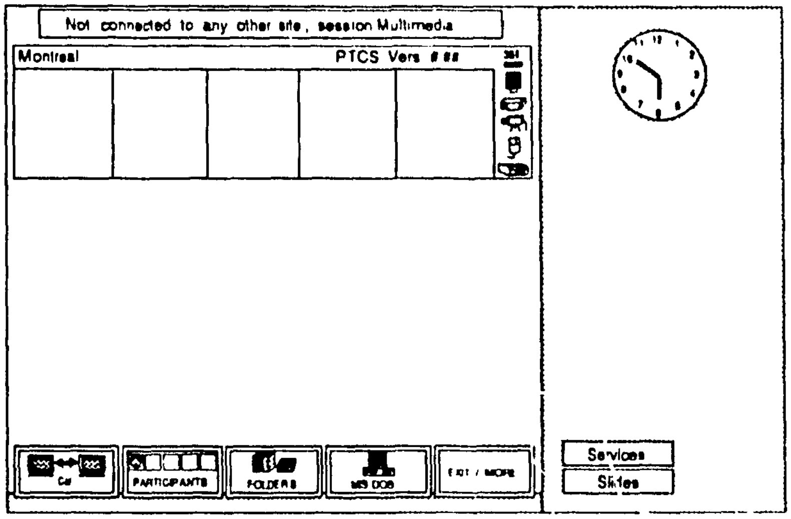
GTCS is a multimedia, multipoint visual conferencing software system. It uses the "shared screen" or "shared space" display concept. A data network between the two computers facilitates text and image sharing between them. A document or slide displayed at any site will be simultaneously displayed on all screens. GTCS supports document loading through a black and white document editor. Participants can simultaneously annotate and amend the visual display in the shared space using the mouse and icons provided on the display. GTCS provides the **Services** and the **Slides** modes of display for group activities.

. *Services mode*

The services mode of display provided by the GTCS is illustrated in figure 3.1. In this mode the GTCS allows the participants to initiate or listen to calls from different sites, and provides the various maintenance modes.

The services mode screen is divided into several windows. The border area is the control area, which contains the clock and two buttons to switch between the **Services** and the **Slides** modes. The area at the top of the screen is reserved for the meeting monitor display. It displays the communication status of the terminal. Just below this display, site windows are displayed. One window will be displayed for each site participating in the meeting, the first one being that of the local site. Each of the windows displays the participant's names and photos, if available, and also small icons describing the available hardware.

Messages for the user are displayed at the bottom of the screen. The five menu buttons provided at the bottom of the screen allow the user to select any of the five sub menus of the services mode.



Not connected to any other site, session Multimedia

**Figure 3.1 - GTCS Services mode display**

. **Call set up** - This menu is used to set up a call to the participating station. Since in our prototype, only two sites are involved, the call set up will result in connecting the local site to the other participating site. The call set up menu will also provide subsequent submenus which will facilitate the local station, to edit the participating site list, dynamically change the site name, listen for incoming calls and also to disconnect the local site from the conference.

. **Participants** - This menu is used to add new participant's names and pictures to the local site description bar, to take and send pictures of participants to other sites and to edit the local user list, from which participants may be selected. Participants already in the list may leave and re-join the conference asynchronously. When new participants join the conference their presence will be communicated to the other participating site, and the new participant's name will appear on the site's display. Similarly, participants can be deleted from the list during the conference using this menu. This menu also provides features to take the picture of the participant, whenever the system is equipped with a camera.

. **Folders** - GTCS provides file management features by providing **folders** on the display. Folders are essentially a form of information unit provided by GTCS to store documents. Three types of folders are provided by the GTCS. The **green**, **red**, and the **blue** folders provide for document storage and retrieval during the session. The contents of the folders may be saved on the hard disk, or the files on the hard disk can be stored on the folder for display and editing purposes during the session. As the folders are the main information unit during the GTCS session, features are provided for sending the contents of the folder to other participating sites during the session. Participants are thus able to transmit and receive the multimedia documents contained in their folders during a session. The document on any of the folders may be printed, when a printer is connected to the system.

. **DOS interface** - The GTCS provides execution of MS-DOS functions such as examining files, sending them to other sites during a session. More advanced functions from MS-DOS may be executed by interactively pausing GTCS during a session. Participants can also send any selected file to the other connected sites by clicking on the **send** button. GTCS will attempt to put these files in the same directory as they came from at the originating site. The sending site can cancel the transmission at any time during the transmission. All connected sites will receive the transmitted file. The participant may put any selected file in the blue folder by activating a sub menu function. For the purposes of consistency, GTCS allows participants to transfer the contents of all relevant directories to the other participants. A participant may also return to the very beginning state of the current GTCS session by activating the **reset folder** function.

. *Slides*

The **slides** mode is the mode used by the participants to exchange information during the conference. The participants can alternate between the **services** and the **slides** mode during the conference. A typical slides mode display will be as indicated in figure 3.2

Any site can save a displayed document originating from another site in a desired folder. This saving operation does not remove the document from the shared space.

The shared display area on the slides mode is provided by a big square in the middle of the screen. Anything appearing on this shared space will appear on all the other screens in the conference. To the right of the shared space is the clock, the pens icon, the lectern and the cleans icon. The "clean" button is used to erase the contents of the shared space. The slides mode provides three coloured pens, and white or liquid paper. Also provided on the slides menu are the highliter, eraser, close button and the keyboard. The pens will be used for annotations during the conference. A notepad, the three folders, and wastebasket are also provided on the display. The camera, the printer and the scanner icons are also provided on the display, whenever the system is configured with those devices.

The pens and the highliters are utilized to annotate the shared display area. There are three pen widths provided for annotating purposes. There are four modes of writing with the pens.

The eraser removes any annotations done by other annotators, thereby restoring the background document underneath The keyboard is used to annotate a document with text. This is the only annotator that can be selected while holding any other annotator. GTCS also provides varying size of characters to be used in conjunction with the keyboard.

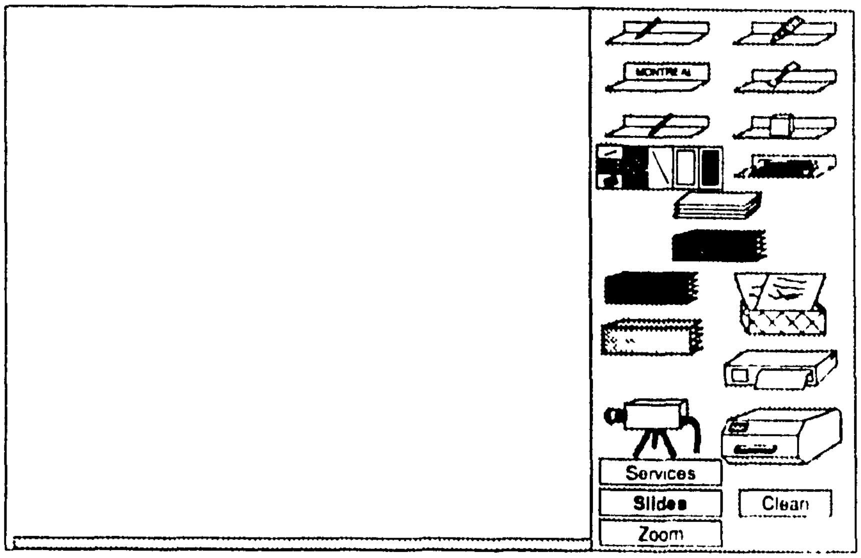


Figure 3.2- Slides mode display

Any document can be scanned whenever the system is provided with a document scanner. A user can introduce the scanned document onto the shared space during a conference. Three scanning modes, i.e., **portrait mode**, **landscape mode**, and **automatic mode** are provided by the GTCS.

### *Communication*

GTCS offers several different ways of communicating among various sites. For this project a back to back null modem link is utilized. The link is established through the COM2 port of the personal computer.

## . *GTCS Features for the support of CSCW*

### . *Bidirectional transmission :*

GTCS provides various forms of communication modes, in full duplex format. For our project, we will use back to back connected null modem link, on an RS 232 interface.

### . *Freedom from Error :*

GTCS provides various types of protocol for error free and recoverable communications.

### . *High connectivity :*

GTCS can be connected in a wide variety of network configurations and is able to transmit to any one or more of many destinations.

### . *High information rate :*

The GTCS communication can be operated at very high speeds when supported by an external communication network.

*.Speech capability :*

GTCS does not provide support for speech transmission. Speech capability is provided by external means.

*. Picture transmission :*

GTCS provides support for external scanner inputs and also for a video camera connection. It is able to provide picture transmission in the video mode and as well as graphic mode.

*. Short transit time delay :*

Transit delay is a function of the communication facilities utilized. GTCS is designed to support a wide variety of communication networks, operating at very high speeds.

*. Addressability of common topics*

GTCS by its provisions of one-to-many transmissions provides a broadcast type facility for group discussions. The shared screen display supports the common topic addressability.

*. Interactive dialogue*

Interactive dialogues based on speech are not supported by GTCS. External audio components are used for speech support GTCS however, uses the "mail box" approach for supporting interactive dialogues.

. ***Asynchronous and Real time interactions***

GTCS provides for participation of the sites in a session asynchronously. Participants can leave their sites and join the conferences at their own convenience. The presence of the participants for a session is indicated on each display screen.

. ***Shared Screen conferencing***

The GTCS offers shared screen conferencing, as it allows the same topics to be displayed on all the participants.

## 3.4 SRS For The Controller

The Software Requirement Specification (SRS) for the controller is described in the following sections. The SRS is structured in the DOD Standard 2167A format. The CSCW session is utilized between the client and the analyst to refine the document to ensure that the requirements set forth by the client has been interpreted properly by the analyst.

All the section and subsection headings which are underlined in the following pages, indicate the applicable sections and subsections for the SRS as per the DOD Standard 2167A requirements. Text enclosed inside a box indicates the conflicting issues arising in the preparation of the SRS, which may invoke dialogues and discussions. Special forms or schematics that will be presented during the discussions are included for reference.

## *Scope*

### *Identification*

This example SRS establishes the requirements for one of the software components of the controller called the "Computer Software Configuration Item (CSCI)", which is a part of the larger Turbine Control System Software (TCSS).

### *Control System Software Overview*

The purpose of the TCSS is to:

- Accept inputs from the Sensor System (SS) and commands from a Person Machine Interface.

- Process the sensor data in a systematic manner and generate output and signals to the Sensor System and provide information to a remote computer for monitoring purpose and to the operator controlling the system.

The TCSS shall realize the following high level functions:

. *control of fuel supply to the engine,*

. *control of starter motor and igniters for automatic start,*

. *control the steady state and transient performance of the turbine,* *protection of turbine under overspeed conditions,*

. *turbine emergency and normal stop control,*

. *monitoring of turbine critical parameters by the operator,*

. *interface to a remote computer.*

## Applicable Documents

TCS1234 ..... Turbine Controller Requirement Specification.

## Engineering Requirements

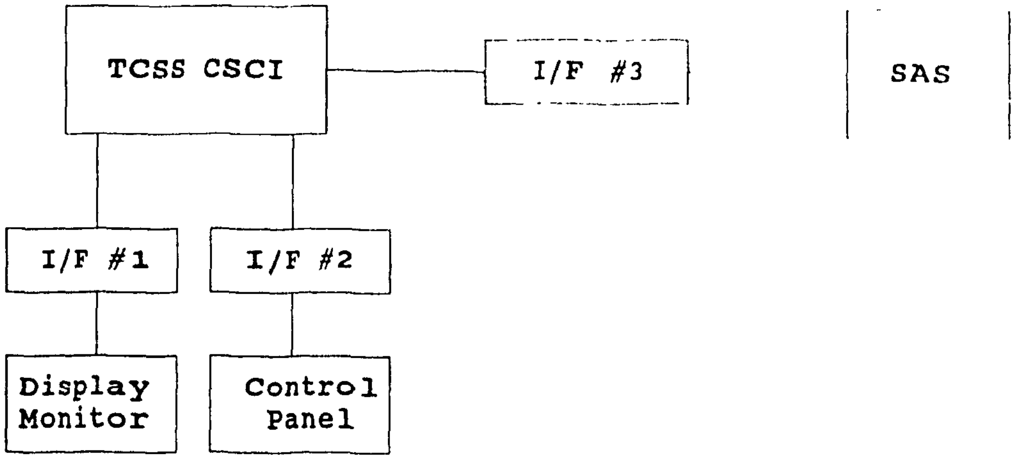
### TCSS External Interface Requirements

In addition to the TCSS, the Control System architecture contains two hardware configuration items (HWCI). They are:

- Sensor System

- Control system hardware components, which will consist of electronic modules, Display monitor and a control panel. The external interface of the TCSS is as shown in figure 3.3.

**This part will invoke discussions and dialogues from the participants, which will result in the modification of the proposed block diagram. The interface to the External computer is missing from the interface diagram.**



SAS : Sensor Actuator System.

I/F : Interface.

CSCI : Computer Software Configuration Item.

TCSS : Turbine Control System Software.

**Figure 3.3 - TCSS External Interface Diagram**

*TCSS Capability Requirements*

The Turbine Control System Software consists of the following functional capabilities:

- Executive Capability (EXEC)
- Person Machine Interface Capability (MMI)
- Sensor Monitoring and Control Capability (SMON)
- Turbine Sequencer Capability
- Remote Computer Communication Capability.

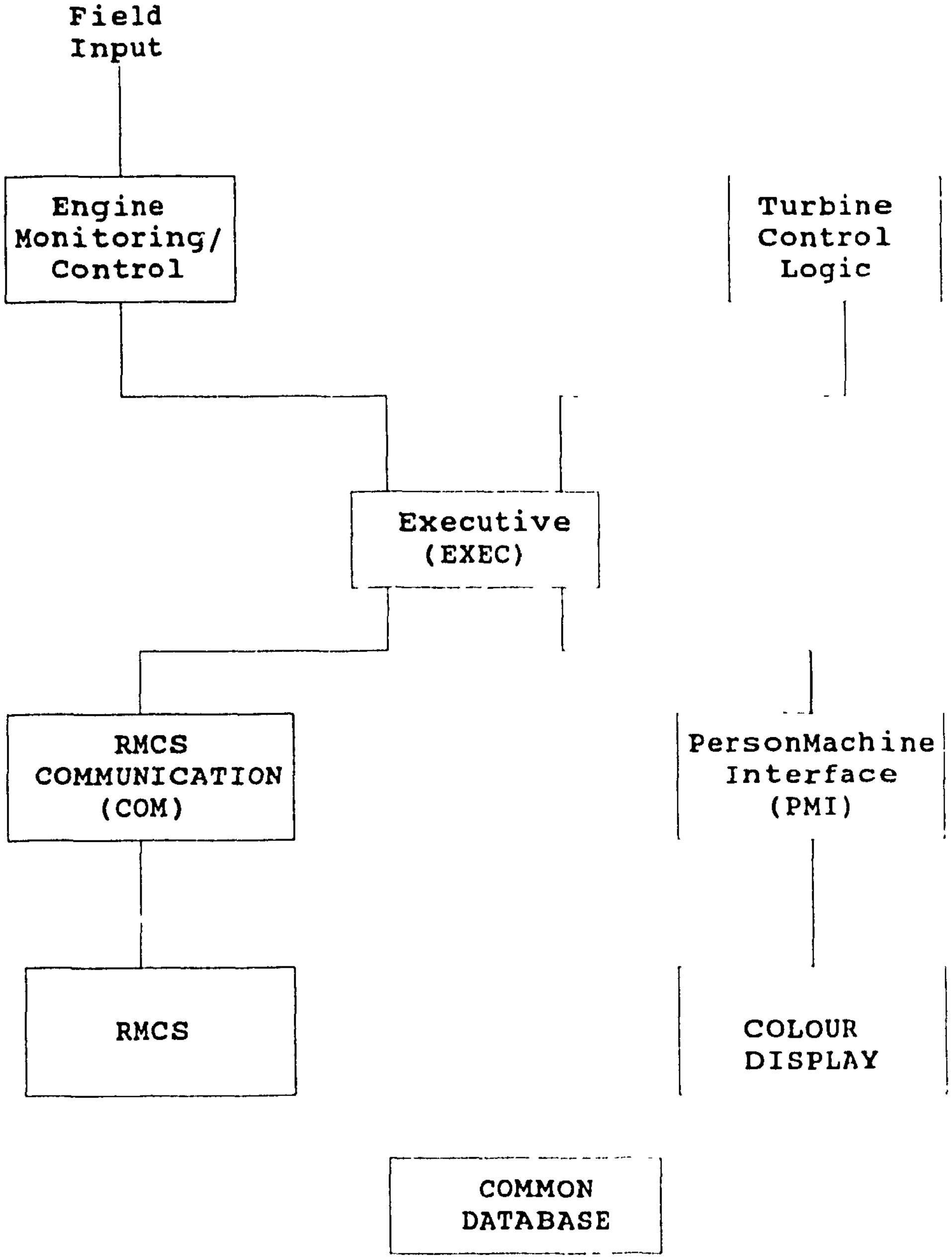
Figure 3.4 shows the conceptual arrangement of these capabilities.

**. Executive Capability**

The purpose of this capability is to provide an environment to execute control system tasks. It provides a mechanism for concurrent process synchronization and communication as well as allocation of resources. In addition to those, EXEC shall also be responsible for initialization of the system, event logging and database management. The EXEC consists of the following sub-capabilities:

*a). Runtime system manager*

The runtime system manager shall be responsible to call all initialization sections of each main task and then start all the main tasks when the system is reset. This capability is executed once after power up or hardware reset. The system manager will be involved in the following areas:



COM : Communication module

RMCS : Remote Monitoring Computer System

Figure 3.4 - TCSS Functional Capabilities

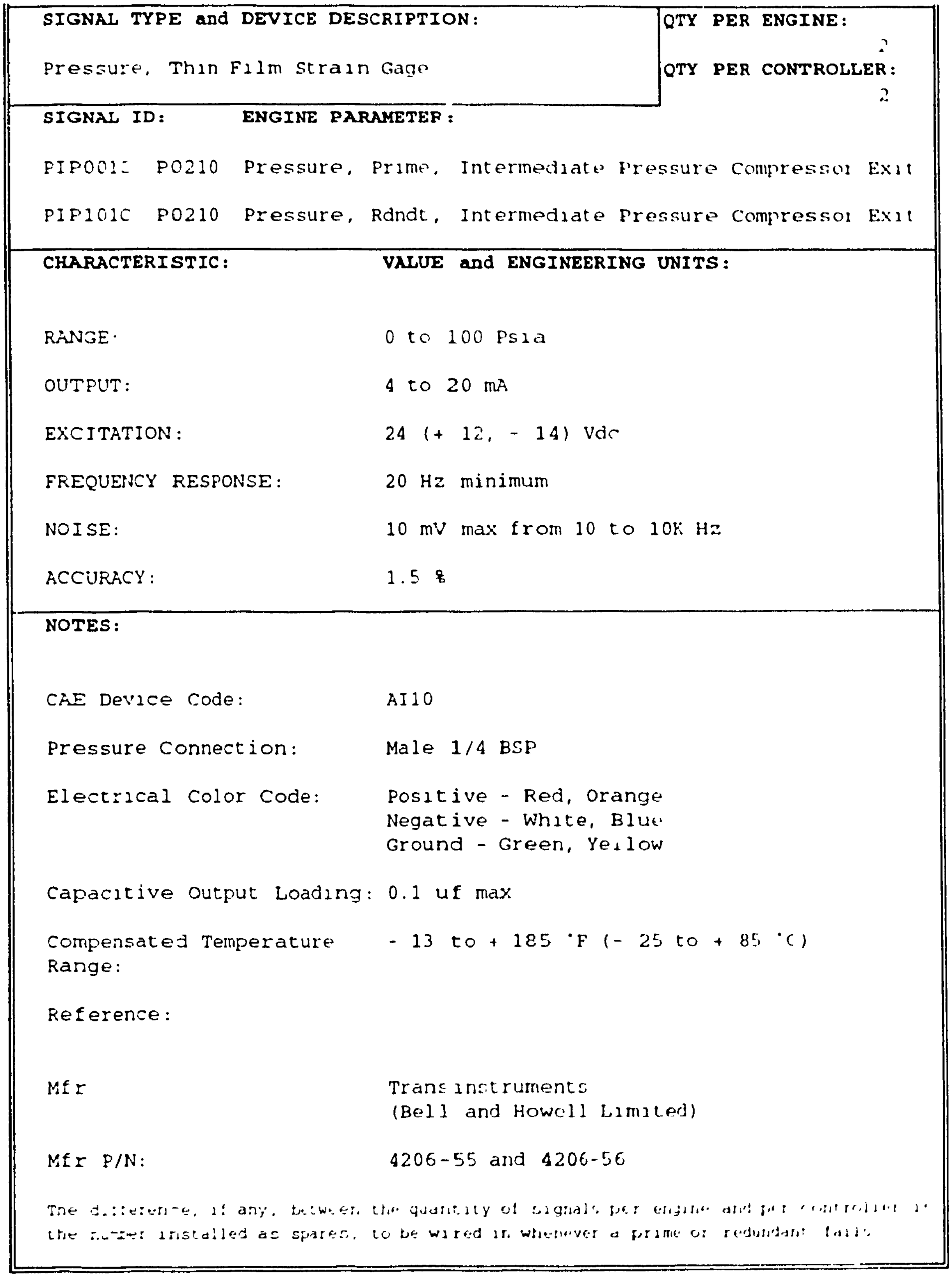
- For any signal value/status changes, the system manager will update the database and inform PMI functions that there was a change in the database.

- For any signal parameter changes (alarm limits, deadband) the system manager will update the database and inform the PMI that there was change in the parameter.

> **This section is not clear. The CSCI did not describe as to how the parameter changes can be effected. Also there is no descriptions of the MMI display as of yet. A sample sensor parameter data sheet as shown in figure 3.5 will be presented during this discussion to convey the concept. The analyst is also required to use the shared display to explain the parameter changes and its significance to the client.**

*b). Data Base Handler*

The database handler is a set of procedures to deal with the different databases. These procedures are the only ones to directly access the database information. Any component that needs to manipulate the database information will do it through a call to the database handler procedures.



**SIGNAL TYPE and DEVICE DESCRIPTION:**

Pressure, Thin Film Strain Gage

**QTY PER ENGINE:** 2

**QTY PER CONTROLLER:** 2

| **SIGNAL ID:** | **ENGINE PARAMETER:** |
|---|---|
| PIP001C P0210 | Pressure, Prime, Intermediate Pressure Compressor Exit |
| PIP101C P0210 | Pressure, Rdndt, Intermediate Pressure Compressor Exit |

| **CHARACTERISTIC:** | **VALUE and ENGINEERING UNITS:** |
|---|---|
| RANGE: | 0 to 100 Psia |
| OUTPUT: | 4 to 20 mA |
| EXCITATION: | 24 (+ 12, - 14) Vdc |
| FREQUENCY RESPONSE: | 20 Hz minimum |
| NOISE: | 10 mV max from 10 to 10K Hz |
| ACCURACY: | 1.5 % |

**NOTES:**

| | |
|---|---|
| CAE Device Code: | AI10 |
| Pressure Connection: | Male 1/4 BSP |
| Electrical Color Code: | Positive - Red, Orange<br>Negative - White, Blue<br>Ground - Green, Yellow |
| Capacitive Output Loading: | 0.1 uf max |
| Compensated Temperature Range: | - 13 to + 185 °F (- 25 to + 85 °C) |
| Reference: | |
| Mfr | Trans instruments<br>(Bell and Howell Limited) |
| Mfr P/N: | 4206-55 and 4206-56 |

The difference, if any, between the quantity of signals per engine and per controller is the number installed as spares, to be wired in whenever a prime or redundant fails

Figure 3.5 - Sensor parameter data sheet

## . Person Machine Interface

The Person Machine Interface (PMI) shall provide all monitoring and control functions required by the operator to interact with the Gas Turbine engine. It will include the following functions:

- Automatic control sequence requests,
- Manual control sequence requests,
- Input/output command support by quick action buttons.

When interfaced with a colour CRT interface, the display on the CRT shall be as shown in figure 3.6. The CRT screen is split into 3 areas (page window area, message window area, menu window area) in the normal mode of display.

The page window area is used to display the pages and can be subdivided into four window regions, such that minimized pages can be displayed in these windows.

Message window area occupies 3 lines of text which are:

line 1 : Event Messages

line 2 : Most recent alarm/device error message

line 3 : Operator information and keypad inputs.

Menu window area occupies three lines of text at the bottom of the screen. This area is subdivided into seven menu option rectangles and one graphic area located at the right of the menu window area. The top graphic is called the "MENU PUSHBUTTON" graphic and the bottom graphic is called the "SELECTED WINDOW" graphic. Menu options can scroll horizontally so that a menu that has more than 7 options, can have all its options made available to the operator.

**The MMI descriptions indicated may not be the one desired by the client. A dialogue in this area is anticipated which may result in the change of display concepts of the MMI. The CSCW features of presenting graphics and forms may be utilized to display the required forms/graphics on the screen during the discussions. The MMI design usually attracts clients attention, as this is one area where he may differ with the developer. Sample graphical symbols will be presented during the MMI design discussions. Figure 3.7 will be presented during the MMI design discussions to show the general format of a typical MMI layout. Figure 3.8 will be displayed using the scanner(if available), to exhibit a sample MMI page that the developer has designed, for client's approval.**

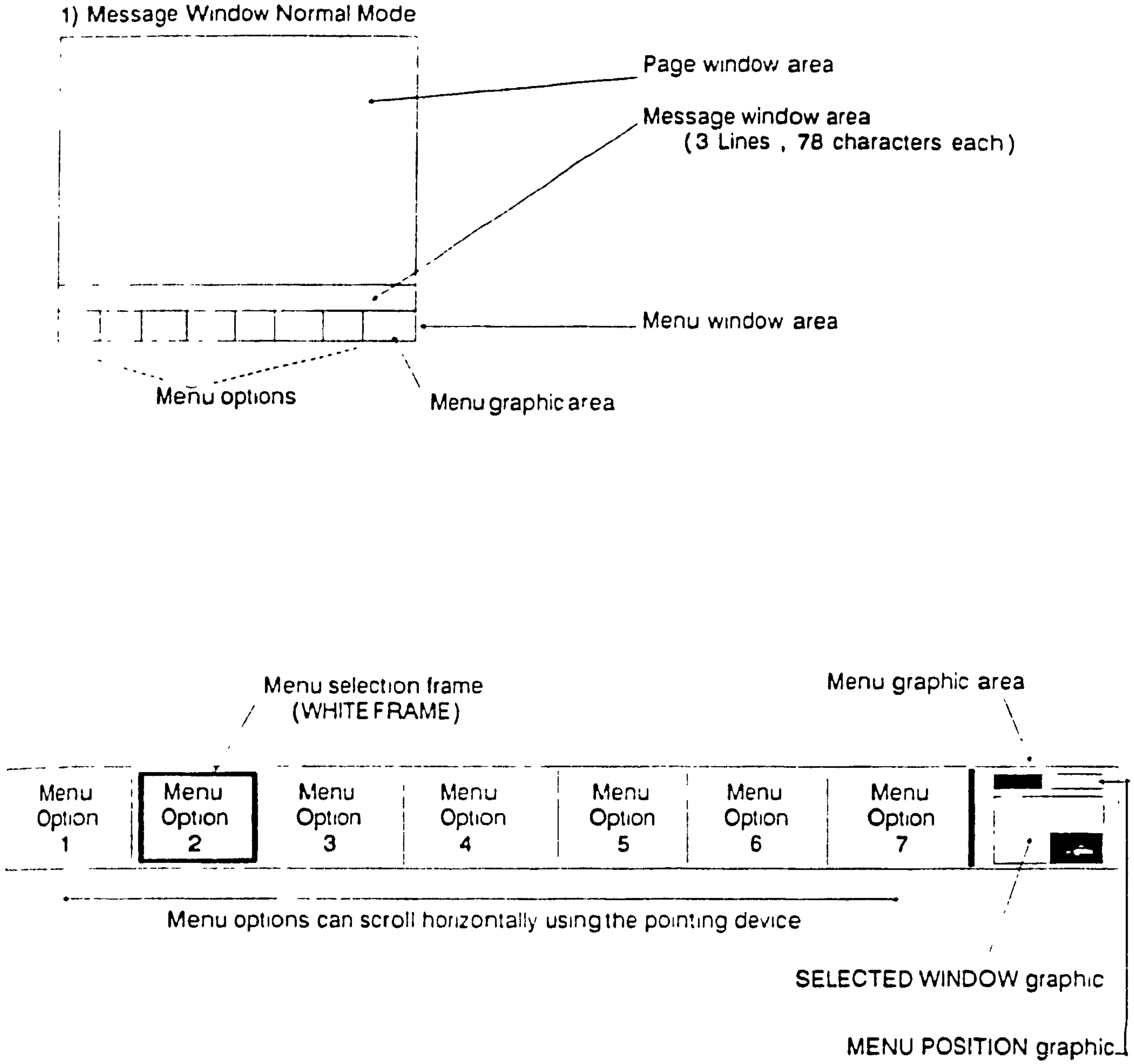


Figure 3.6 - PMI Display layout

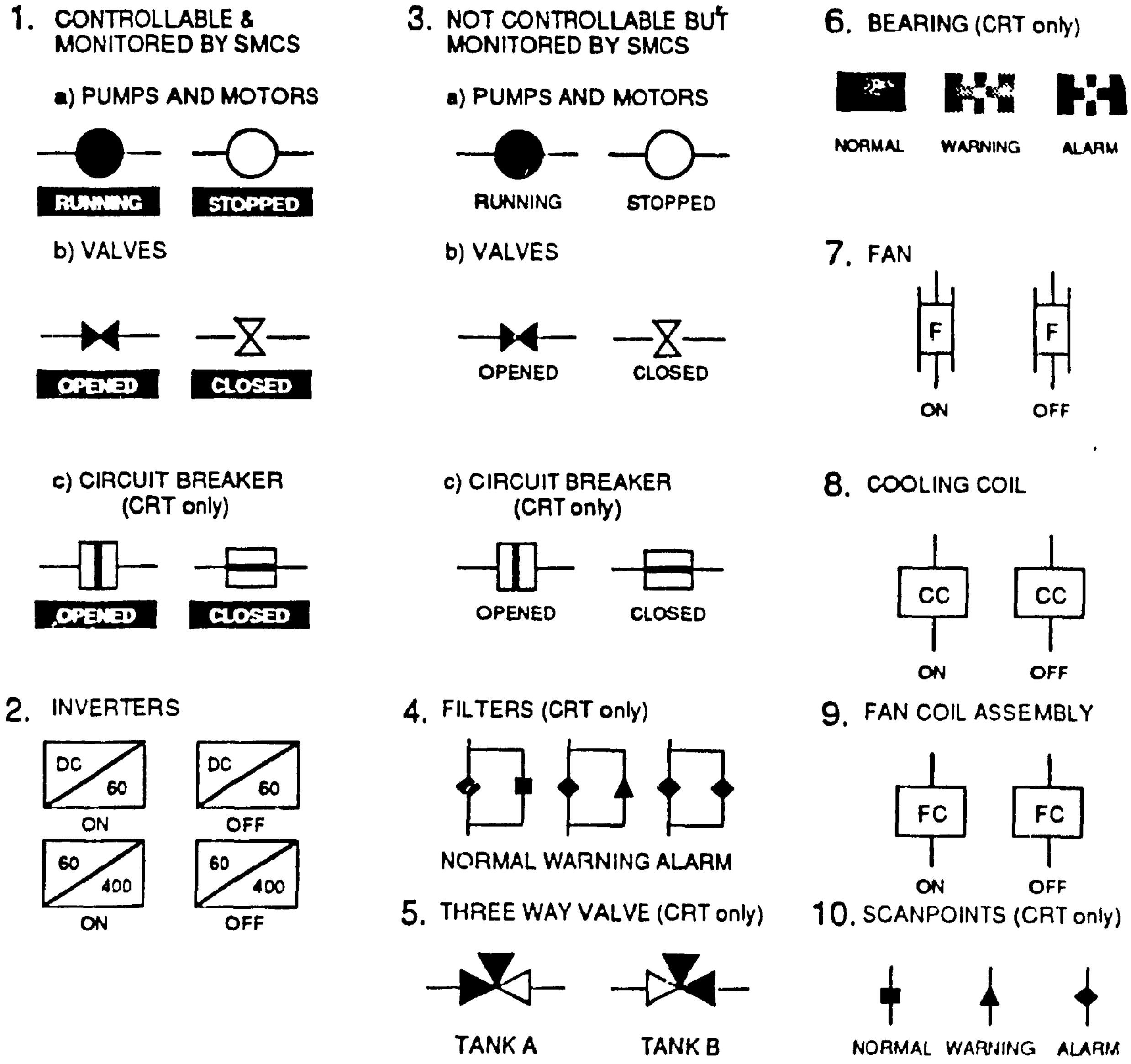


Figure 3.7 - Graphic symbols for the PMI

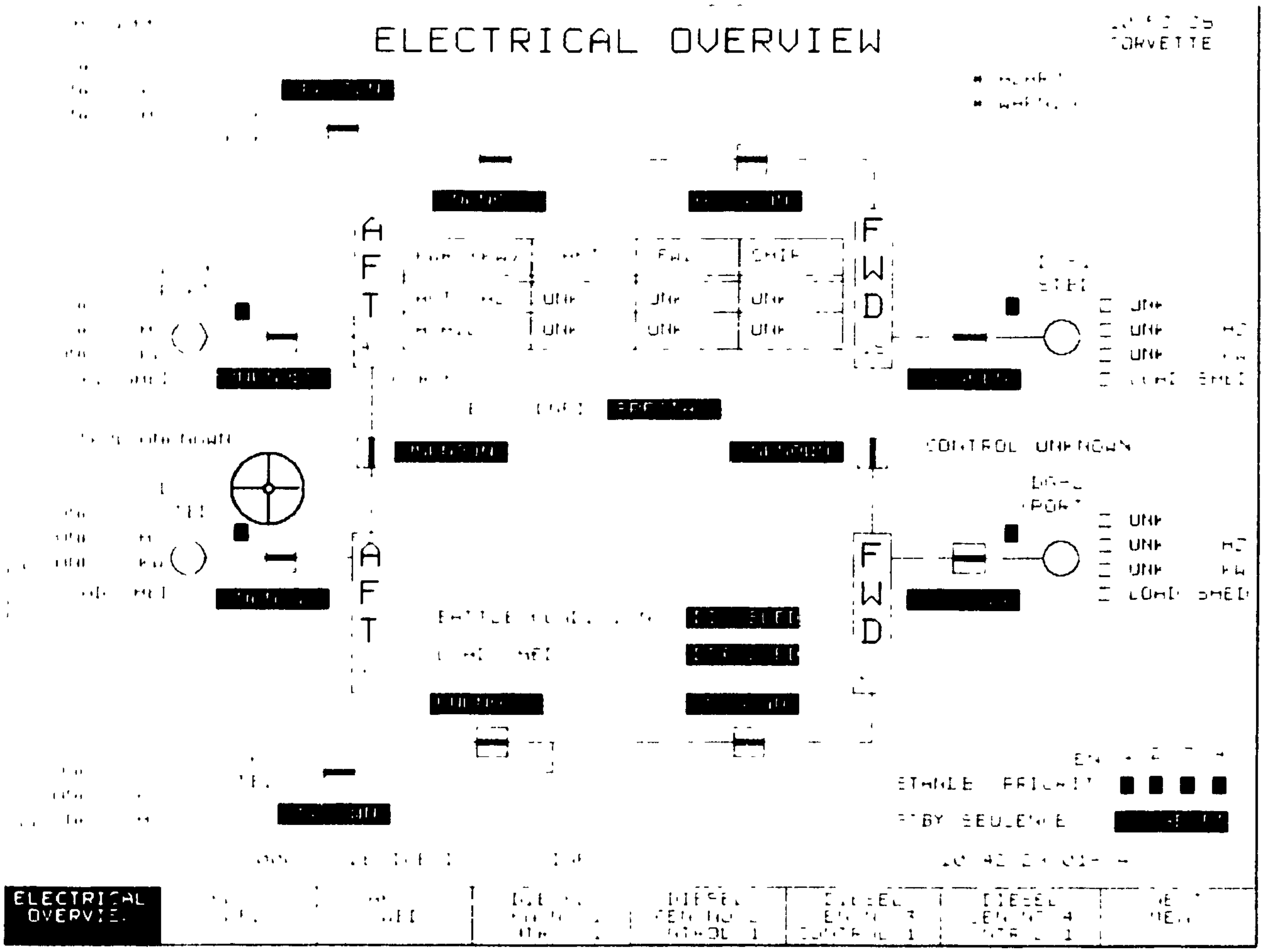


Figure 3.8 - A sample PMI page

**. Sensor Monitoring and Control Capability (SMON)**

Sensor monitoring shall acquire sensor data from the sensors connected to the engine. Sensor monitoring shall sample each sensor point every **250 msecs** for normal engine monitoring. For critical parameters the sensors are monitored every **100 msecs**

The following functions shall be performed on each sensor input:

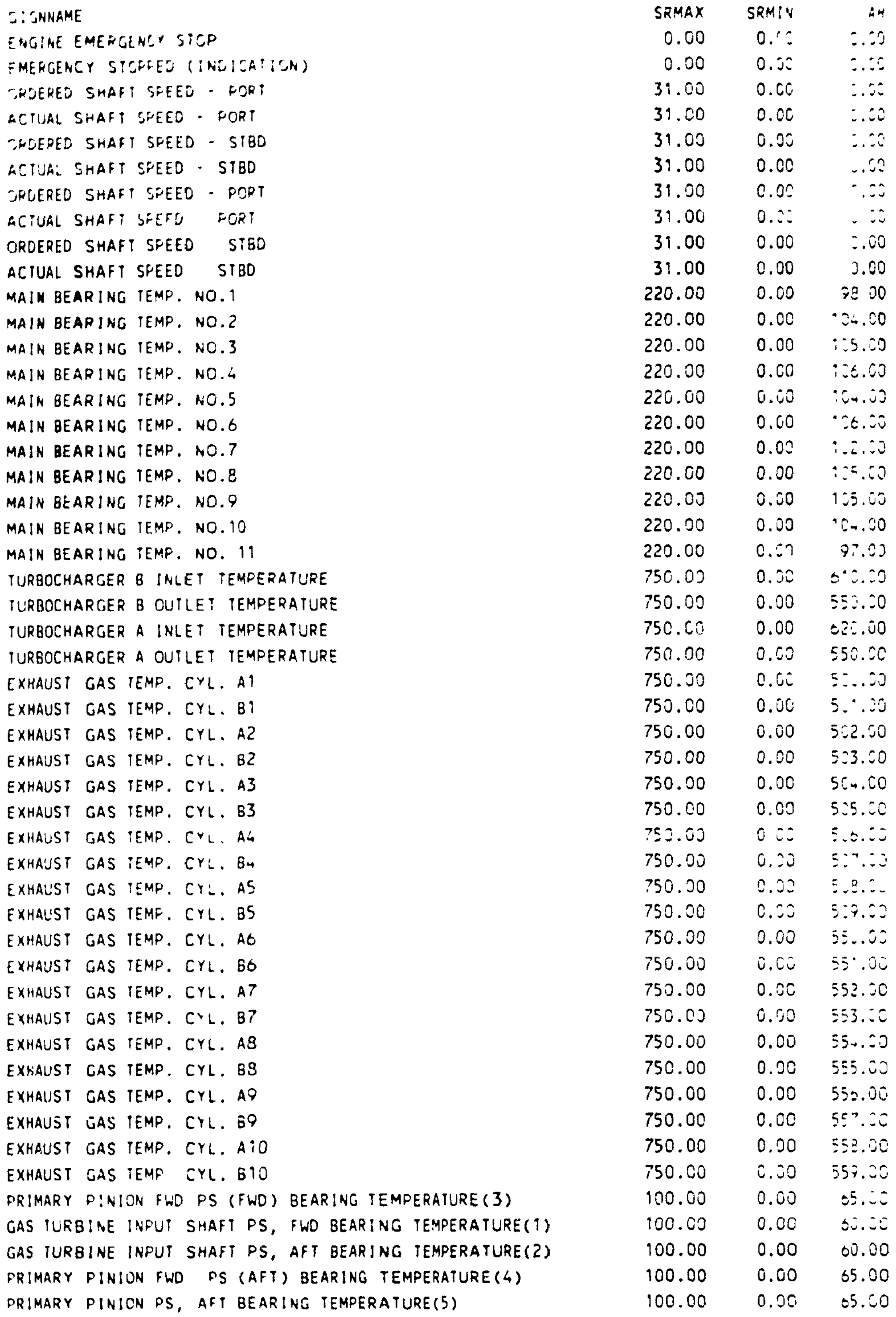
- Update the database if a change is detected in the sensor value.

- Test the sampled signal for alarm and warning limits. Alarm and warning limits shall be used to limit reporting of state changes around the alarm and warning thresholds.

- Inhibit the alarms and warnings automatically when the sensor associated with the signal is out of operation.

- Incorporate rate of change algorithms to validate an alarm.

**This section needs inputs from the client as to the details of scan frequency, sensor ranges and dead bands required for each sensor input. Furthermore, the sensor list will have to be checked individually for determining the criticality of the sensor parameters. The sensor list will be displayed during the session, to verify parameter values and ranges for each sensor. Figure 3.9 shows a sample sensor data sheet.**



| SIGNNAME | SRMAX | SRMIN | AH |
|---|---|---|---|
| ENGINE EMERGENCY STOP | 0.00 | 0.00 | 0.00 |
| EMERGENCY STOPPED (INDICATION) | 0.00 | 0.00 | 0.00 |
| ORDERED SHAFT SPEED - PORT | 31.00 | 0.00 | 0.00 |
| ACTUAL SHAFT SPEED - PORT | 31.00 | 0.00 | 0.00 |
| ORDERED SHAFT SPEED - STBD | 31.00 | 0.00 | 0.00 |
| ACTUAL SHAFT SPEED - STBD | 31.00 | 0.00 | 0.00 |
| ORDERED SHAFT SPEED - PORT | 31.00 | 0.00 | 0.00 |
| ACTUAL SHAFT SPEED PORT | 31.00 | 0.00 | 0.00 |
| ORDERED SHAFT SPEED STBD | 31.00 | 0.00 | 0.00 |
| ACTUAL SHAFT SPEED STBD | 31.00 | 0.00 | 0.00 |
| MAIN BEARING TEMP. NO.1 | 220.00 | 0.00 | 98.00 |
| MAIN BEARING TEMP. NO.2 | 220.00 | 0.00 | 104.00 |
| MAIN BEARING TEMP. NO.3 | 220.00 | 0.00 | 105.00 |
| MAIN BEARING TEMP. NO.4 | 220.00 | 0.00 | 106.00 |
| MAIN BEARING TEMP. NO.5 | 220.00 | 0.00 | 104.00 |
| MAIN BEARING TEMP. NO.6 | 220.00 | 0.00 | 106.00 |
| MAIN BEARING TEMP. NO.7 | 220.00 | 0.00 | 102.00 |
| MAIN BEARING TEMP. NO.8 | 220.00 | 0.00 | 105.00 |
| MAIN BEARING TEMP. NO.9 | 220.00 | 0.00 | 105.00 |
| MAIN BEARING TEMP. NO.10 | 220.00 | 0.00 | 104.00 |
| MAIN BEARING TEMP. NO. 11 | 220.00 | 0.00 | 97.00 |
| TURBOCHARGER B INLET TEMPERATURE | 750.00 | 0.00 | 610.00 |
| TURBOCHARGER B OUTLET TEMPERATURE | 750.00 | 0.00 | 550.00 |
| TURBOCHARGER A INLET TEMPERATURE | 750.00 | 0.00 | 620.00 |
| TURBOCHARGER A OUTLET TEMPERATURE | 750.00 | 0.00 | 550.00 |
| EXHAUST GAS TEMP. CYL. A1 | 750.00 | 0.00 | 500.00 |
| EXHAUST GAS TEMP. CYL. B1 | 750.00 | 0.00 | 501.00 |
| EXHAUST GAS TEMP. CYL. A2 | 750.00 | 0.00 | 502.00 |
| EXHAUST GAS TEMP. CYL. B2 | 750.00 | 0.00 | 503.00 |
| EXHAUST GAS TEMP. CYL. A3 | 750.00 | 0.00 | 504.00 |
| EXHAUST GAS TEMP. CYL. B3 | 750.00 | 0.00 | 505.00 |
| EXHAUST GAS TEMP. CYL. A4 | 750.00 | 0.00 | 506.00 |
| EXHAUST GAS TEMP. CYL. B4 | 750.00 | 0.00 | 507.00 |
| EXHAUST GAS TEMP. CYL. A5 | 750.00 | 0.00 | 508.00 |
| EXHAUST GAS TEMP. CYL. B5 | 750.00 | 0.00 | 509.00 |
| EXHAUST GAS TEMP. CYL. A6 | 750.00 | 0.00 | 550.00 |
| EXHAUST GAS TEMP. CYL. B6 | 750.00 | 0.00 | 551.00 |
| EXHAUST GAS TEMP. CYL. A7 | 750.00 | 0.00 | 552.00 |
| EXHAUST GAS TEMP. CYL. B7 | 750.00 | 0.00 | 553.00 |
| EXHAUST GAS TEMP. CYL. A8 | 750.00 | 0.00 | 554.00 |
| EXHAUST GAS TEMP. CYL. B8 | 750.00 | 0.00 | 555.00 |
| EXHAUST GAS TEMP. CYL. A9 | 750.00 | 0.00 | 556.00 |
| EXHAUST GAS TEMP. CYL. B9 | 750.00 | 0.00 | 557.00 |
| EXHAUST GAS TEMP. CYL. A10 | 750.00 | 0.00 | 558.00 |
| EXHAUST GAS TEMP CYL. B10 | 750.00 | 0.00 | 559.00 |
| PRIMARY PINION FWD PS (FWD) BEARING TEMPERATURE(3) | 100.00 | 0.00 | 65.00 |
| GAS TURBINE INPUT SHAFT PS, FWD BEARING TEMPERATURE(1) | 100.00 | 0.00 | 60.00 |
| GAS TURBINE INPUT SHAFT PS, AFT BEARING TEMPERATURE(2) | 100.00 | 0.00 | 60.00 |
| PRIMARY PINION FWD PS (AFT) BEARING TEMPERATURE(4) | 100.00 | 0.00 | 65.00 |
| PRIMARY PINION PS, AFT BEARING TEMPERATURE(5) | 100.00 | 0.00 | 65.00 |

Figure 3.9 - Sample sensor list

### . Turbine sequencer capability

The Turbine sequencer shall provide the following functions :

- Turbine start and stop sequences,

- Turbine protection functions,

- Turbine Emergency control functions,

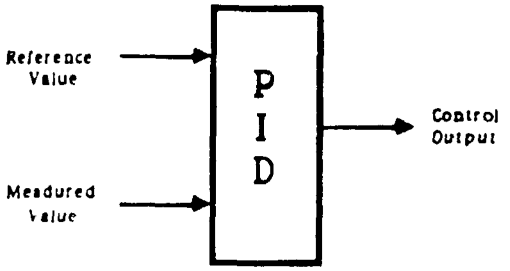
The Turbine sequencer shall incorporate a PID type controller for fuel and speed control.

Details of start/stop and emergency interlocks are to be discussed and provided during the collaborative process. The definitions of the PID controller loop and its operations are explained via figure 3.10, exhibited through the scanner.

### . Remote Computer Communication Capability

The controller shall provide a link for remote computer communications.

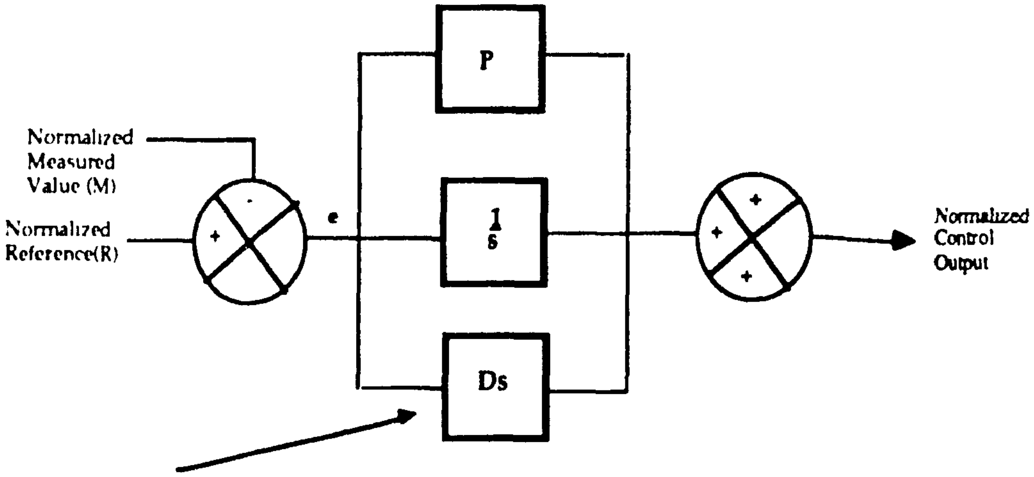
Details of the remote computer link and associated protocol are to be discussed during the CSCW process. Details of data transfer rate, command and response formats are finalized during the session.



$$Op = P.e + I\int e.dt + D.\frac{de}{dt}$$

where
P is the proportional gain
I is the integral gain
D is the derivative gain

Op is the controller normalised output
e is the normalized error



The pure differentiator is approximated by $\frac{s}{\tau s + 1}$ wher$\tau$ = 0.05 sec

Figure 3.10 - PID control loop

## . *CSCI Internal Interfaces*

The interfaces between the capabilities in the previous section is shown in figure 3.11. The following internal interfaces are provided:

- Interface between MMI and Executive capability.

- Interface between Sensor monitoring and Executive capability.

- Interface between turbine sequencer and the Executive capability.

- Interface between the remote computer and the executive capability

## . *TCSS CSCI data element requirements*

### . *Internal data elements*

Will be included in the software design document.

### . *External data elements*

Will be as indicated below.

**This section will be edited during the collaborative process.**

## . *Adaptation Requirements*

### . *Installation-dependent requirements :*

- None

### . *Operational parameters :*

- None

### . *Sizing and Timing requirements*

*Memory*

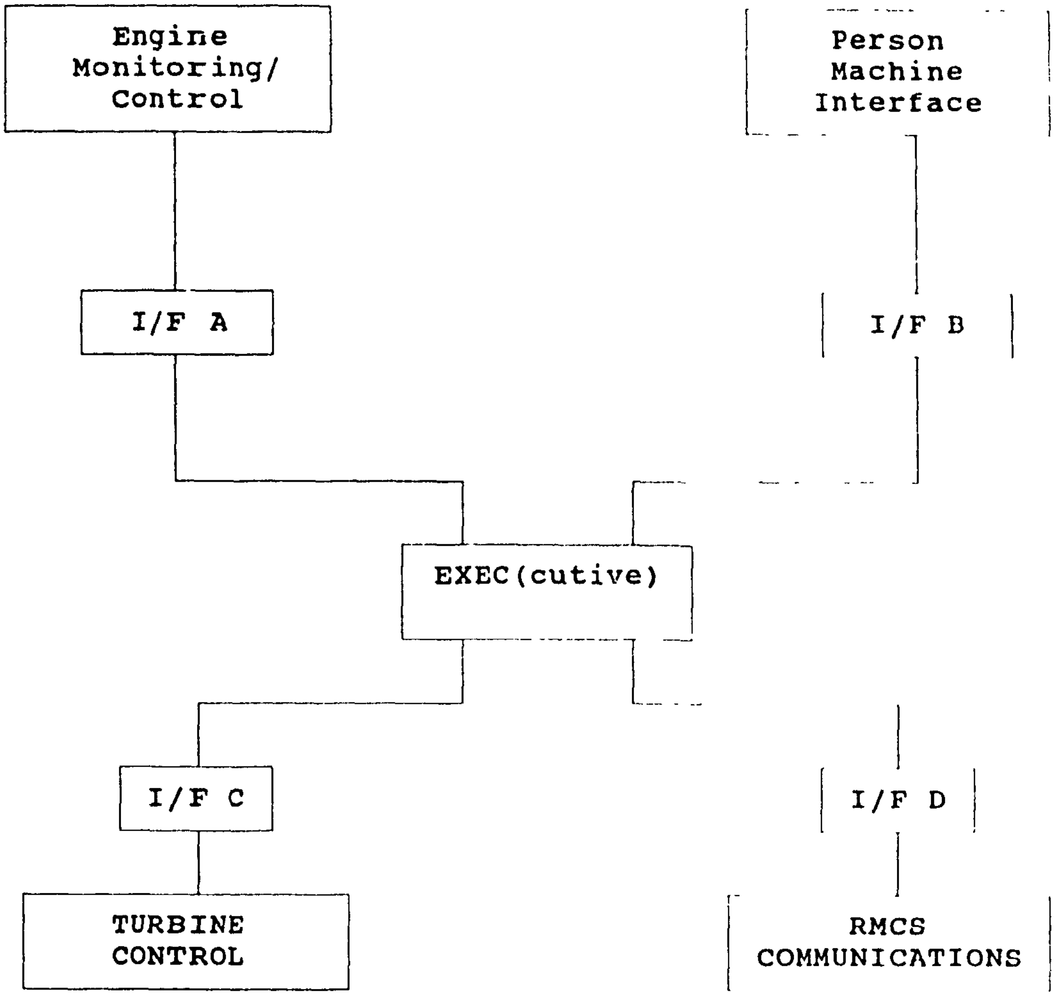
The control system will provide a memory spare capacity for a growth of 20% from its current design.

> **This issue will be required to be defined more precisely during the collaborative process. The client must address all of his future requirements, with respect to possible growth and any other added requirements.**

### . *Security Requirements*

None.

> **If any , will be defined during the collaborative process.**



I/F : Interface

RMCS : Remote Monitoring Computer System.

Figure 3.11 - TCSS Internal Interface

. *Design Constraints*

The TCSS shall be designed, developed and tested in accordance with DOD-Standard-2167A, sections 4.1,4.2,4.4,5.1,5.3 and 5.4. The software shall be written in C+ + object oriented language.

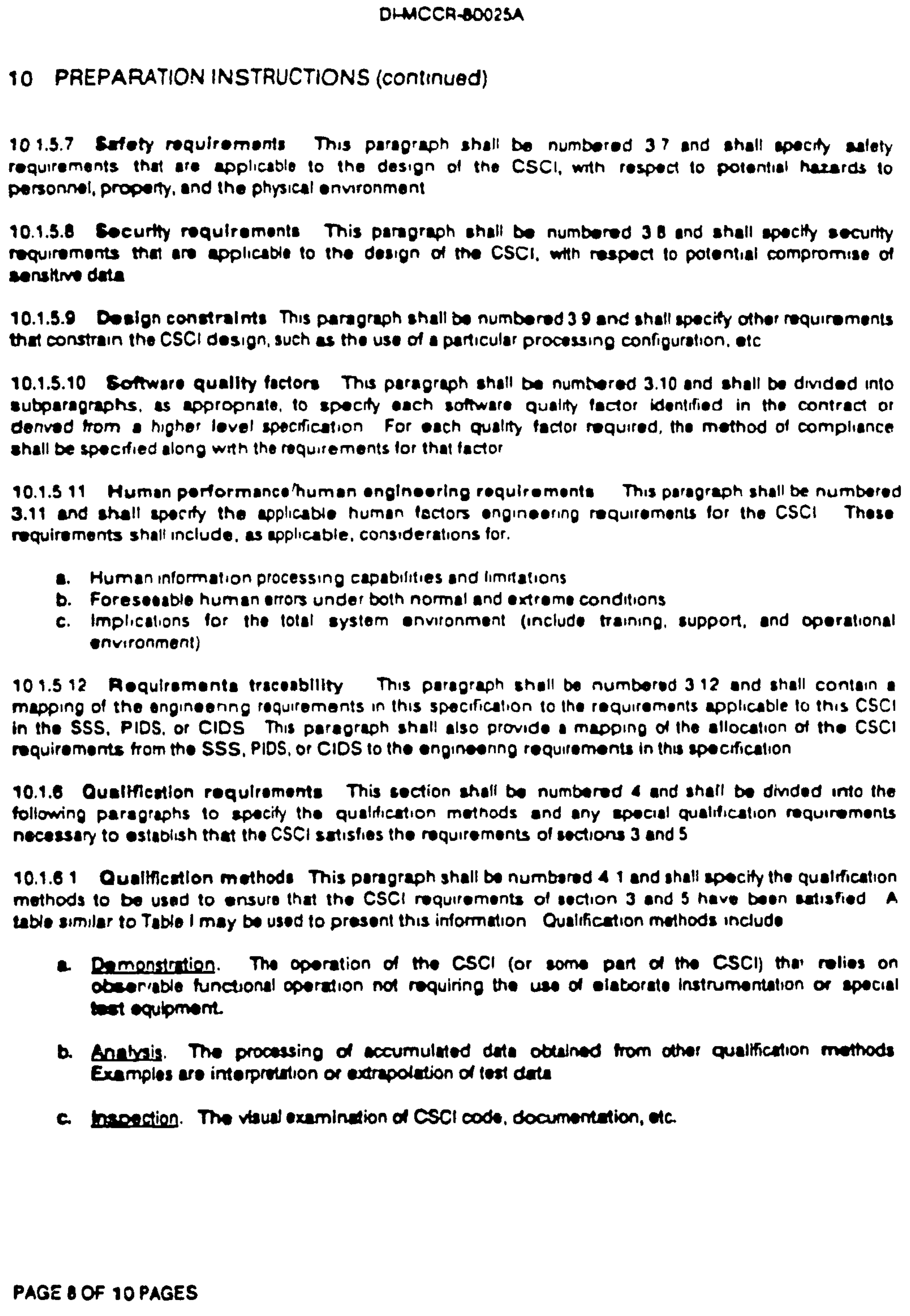
**The client must be informed about the applicable sections, if he has some reservations about any particular requirement. The DOD Standard 2167A sections are produced on the shared display during this discussion. The sample DOD standard document with the relevant sections is shown in figure 3.12**

. *Software Quality Factors*

Quality evaluation methods shall be :

- By analysis,
- By demonstration,
- By inspection.

**The appropriate methods used must be described by the joint process. Requirement verification for each category must be detailed here.**



DI-MCCR-80025A

10 PREPARATION INSTRUCTIONS (continued)

10 1.5.7 **Safety requirements** This paragraph shall be numbered 3 7 and shall specify safety requirements that are applicable to the design of the CSCI, with respect to potential hazards to personnel, property, and the physical environment

10.1.5.8 **Security requirements** This paragraph shall be numbered 3 8 and shall specify security requirements that are applicable to the design of the CSCI, with respect to potential compromise of sensitive data

10.1.5.9 **Design constraints** This paragraph shall be numbered 3 9 and shall specify other requirements that constrain the CSCI design, such as the use of a particular processing configuration, etc

10.1.5.10 **Software quality factors** This paragraph shall be numbered 3.10 and shall be divided into subparagraphs, as appropriate, to specify each software quality factor identified in the contract or derived from a higher level specification For each quality factor required, the method of compliance shall be specified along with the requirements for that factor

10.1.5 11 **Human performance/human engineering requirements** This paragraph shall be numbered 3.11 and shall specify the applicable human factors engineering requirements for the CSCI These requirements shall include, as applicable, considerations for.

a. Human information processing capabilities and limitations
b. Foreseeable human errors under both normal and extreme conditions
c. Implications for the total system environment (include training, support, and operational environment)

10 1.5 12 **Requirements traceability** This paragraph shall be numbered 3 12 and shall contain a mapping of the engineering requirements in this specification to the requirements applicable to this CSCI in the SSS, PIDS, or CIDS This paragraph shall also provide a mapping of the allocation of the CSCI requirements from the SSS, PIDS, or CIDS to the engineering requirements in this specification

10.1.6 **Qualification requirements** This section shall be numbered 4 and shall be divided into the following paragraphs to specify the qualification methods and any special qualification requirements necessary to establish that the CSCI satisfies the requirements of sections 3 and 5

10.1.6 1 **Qualification methods** This paragraph shall be numbered 4 1 and shall specify the qualification methods to be used to ensure that the CSCI requirements of section 3 and 5 have been satisfied A table similar to Table I may be used to present this information Qualification methods include

a. Demonstration. The operation of the CSCI (or some part of the CSCI) that relies on observable functional operation not requiring the use of elaborate instrumentation or special test equipment.

b. Analysis. The processing of accumulated data obtained from other qualification methods Examples are interpretation or extrapolation of test data

c. Inspection. The visual examination of CSCI code, documentation, etc.

PAGE 8 OF 10 PAGES

Figure 3.12 - DOD standard specification sheet.

*. Requirement Traceability*

A cross reference traceability matrix will be developed for each requirement stated in this document with the control system requirement specification provided by the client.

**This section will be completed during the collaborative process, with the client addressing each of his requirements, and the analyst providing the appropriate section in the SRS.**

## *Qualification Requirements*

Qualification requirements shall be demonstrated by the following methods :

- Analysis
- Demonstration
- Inspection

**Each of the methods will be explained with applicable references to the requirement specification.**

## *Preparations for delivery*

**The client to indicate the proper media for the delivery of software and other related instructions.**

## 3.5 Summary

I have addressed the production of a Software Requirement Specification document in this section, using a multi media communication supported computer system Sections in the SRS where the client and analyst are needed to discuss and edit the document are highlighted. This model will be used during the demonstration to explore the capabilities of the GTCS environment for a CSCW. Descriptions of the GTCS is also provided in this section as a quick summary, and its features which provide the needed requirement is also addressed. The software project used for the implementation is only a sample design and does not apply to any particular equipment.

# CHAPTER IV

# SYSTEM CONSTRAINTS AND FURTHER ENHANCEMENTS

## 4.1 Introduction

The computing environment used for this project implementation, consisted of two work stations connected via a null modem communication link. The groupware supporting the collaboration, ie., the GTCS (groupware Telecommunication Software), was a PC based multimedia, multipoint visual conferencing system. The "shared space" display concept of the GTCS provides support for most of the issues required for collaboration. GTCS however does not offer some of the desirable features needed for collaboration. The following sections identify those constraints.

## 4.2 GTCS Constraints

### *a) Public and private work space*

GTCS does not provide a separate public and private work spaces. The shared screen conferencing provided by the GTCS is strictly a WYSIWIS (what you see is what I see) type of display. This prevents participants from editing or note taking of the subject in a private environment during the collaboration. The only way the participants can go into a private caucus or discussion is for them to disconnect from the conference, and reconnect back, after the completion of their private work.

Strict WYSIWIS by itself, is not sufficient enough to provide an ideal CSCW environment. The **"Tri window"** display concept described in this report, based upon both public and private work spaces for the participants, is desirable during the collaborative process.

### *b) Stage Control mechanisms*

GTCS does not provide any internal mechanisms for stage control during collaboration. Stage control will have to be accomplished by some form of external means. One of the possible techniques will be to implement the stage control through discussions among the participants. For our project stage control is accomplished by conferring, using an external telephone link.

### *c) Database Model*

GTCS uses a "cooperative data base model". Each machine will have a copy of the database, and changes are installed by broadcasting the change without any synchronization. Maintaining the consisting of the data base during the collaboration is entirely left to the participants. Updating of the data base during the collaboration is accomplished by each of the participants, by way of transferring the contents of the screen to their own local data base.

### *d) Consistency control*

GTCS provides consistency control only to the shared display space during the collaboration. Consistency of the shared space is accomplished by the GTCS by transferring the shared display contents to all of the participant's display during the collaboration.

### *e) Concurrency Control*

GTCS provides concurrent display updates through its communication facilities. The communication utilized for the implementation consists of a null modem link, running at 2400 bits per second. However, higher speed LANs may be supported by GTCS to ensure concurrency control.

### *f) Asynchronous participation*

GTCS allows participants to join or leave the collaborative session asynchronously. When participants leave an active session a message will be displayed on each of the remaining participant's shared space that the particular participant has disconnected from the session. However, when a participant wishes to join an active session, he will have to interrupt the active session by sending a connection request message, and the "master station" must allow the new participant to join the active session by accepting the connection request. This may be slightly disadvantageous when a participant wishes to join the active session in the middle of an important discussion.

### *g) Voting*

GTCS does not provide mechanisms for voting during the collaboration. Consensus or voting on multiple choices will have to be obtained by some means external to GTCS, possibly by the external voice link.

### *h) Flexible user coupling*

One important issue in collaborative systems is the kind of coupling a user can have among the various windows displaying a shared workspace. A flexible coupling allows users to control several aspects of the coupling among shared windows including which values in these windows are coupled, when changes to these values are broadcast and received, how "correct" a value must be before it is broadcast or received, which users see the same view of a value, and whether a user can specify coupling parameters for other users. GTCS does not support flexible user coupling, and none of the parameters with reference to the shared display, ie., the display size, orientation, etc, can be modified by the participant.

### *i) Roles*

GTCS provides an open environment for group work. The master-observer role concept proposed for a collaborative session in this report is not supported by the GTCS. Also on the shared display there is no indication as to who is controlling the session in progress. Master or observers roles will have to be identified by a mutual discussion among the participants, and a consensus must be arrived at during the session.

## 4.3 Project Enhancements

The GTCS used on the project can be enhanced with additional features, in order to provide a better collaborative environment. The following sections describe the tools and utilities that can be used with the GTCS to achieve a better user interface.

### *a) Communication*

GTCS provides tools for synchronous high speed communications. An EICON card in conjunction with the access/X.25 network interface operated with the GTCS can handle line speeds of up to 56000bps.

A DATAPAC type line can be utilized to connect one GTCS station to many sites in a single conference. One DATAPAC 3000 link can contain several virtual circuits which the GTCS can use to bridge remote locations. GTCS in the listen mode will bridge several remote DATAPAC locations together. Figure 4.1 shows the GTCS configuration using DATAPAC.

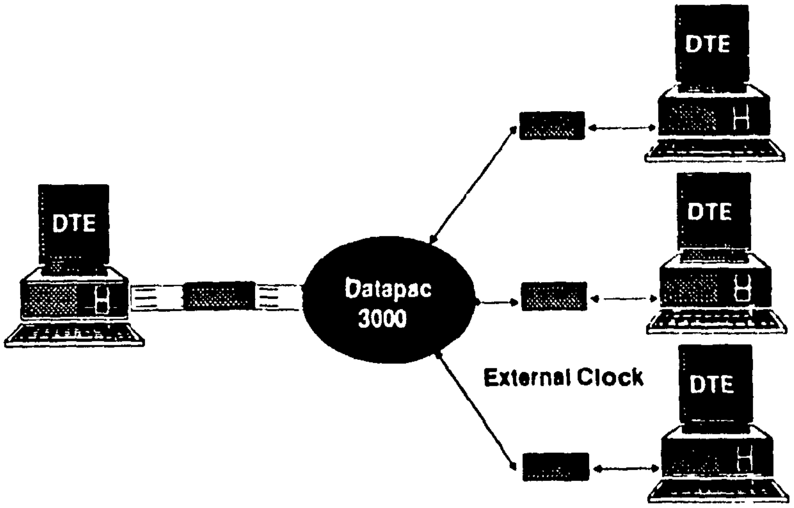


Figure 4.1 - GTCS using DATAPACK

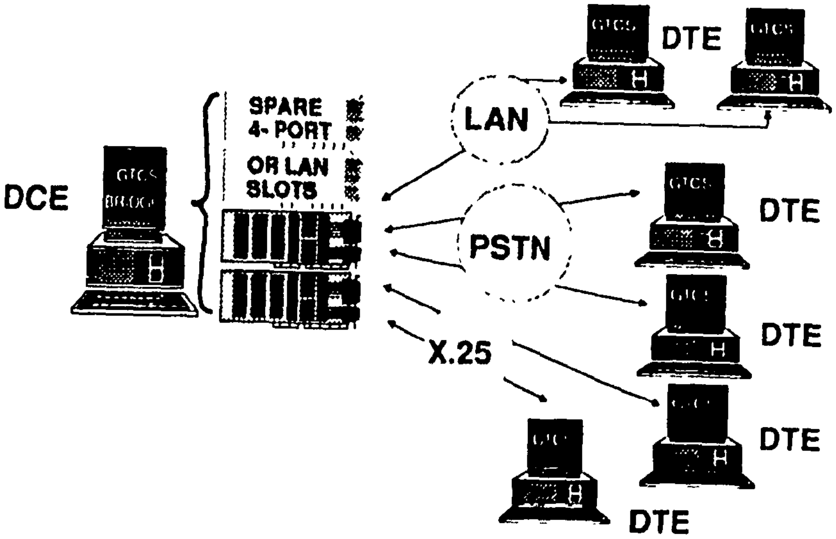


Figure 4.2 - GTCS data Bridge

GTCS can also provide hook ups to Local Area Networks, ISDN and bridge communication. The bridge is a separate PC that is used as a communication bridge that will allow different communications configurations to talk with each other. GTCS configurations with a bridge network are shown in figure 4.2.

### *b) Document Scanner*

GTCS can communicate with a scanner attached to it, which can be used during the collaboration for scanning documents and transmitting the documents to the participating sites. A scanner will be useful to send displays in formats that are easily comprehensible during the collaboration. A user can introduce scanned documents onto the shared space during a conference.

### *c) Video Camera connection*

A video camera can be connected to the GTCS, which can be used to take pictures of a participant and send it onto the shared display screen during the conference. When the participant's picture is taken it will appear on the various site's shared display as shown in figure 4.3

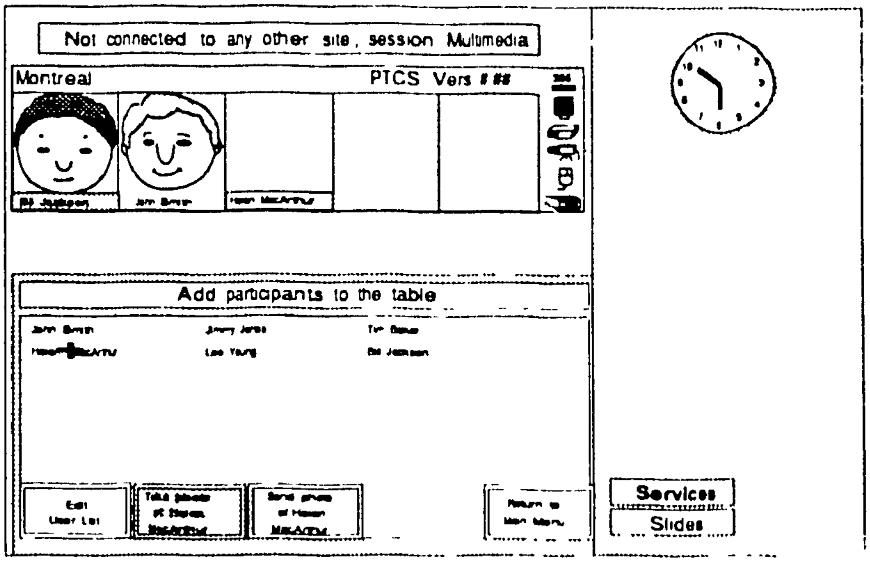


Figure 4.3 - Taking a photo of the participant

## 4.4 Summary

Some of the constraints that are associated with GTCS are explained in this section. Features such as, lack of flexible user interface, consistency control, and concurrency control mechanisms with the GTCS can result in major shortfalls for some collaborative applications. However, with tools such as the camera, document scanner and GTCS's ability to support a very high speed communication, GTCS will be sufficient to support a variety of collaborative applications.

# CHAPTER 5

# CONCLUSIONS

## 5.1 Conclusions and Further Work

The goal of this project work was to explore the application of Computer Supported Cooperative work, to software engineering, in particular, for specifying the software requirements for a real time control application. Various techniques and issues associated with the Computer Supported Cooperative Work were studied and presented in this report. Applications and benefits of CSCW were demonstrated with a case study, for specifying the software requirement specification. A teleconferencing based software on a two node network, was used to show its application. The network could be modified to include multiple nodes and used for larger applications. A two node network was implemented because of the unavailability of a larger network at the time of writing this report. A set of improvements to the teleconferencing software were suggested, which would result in creating a more beneficial CSCW environment.

Based on the requirements for suppor ng the CSCW, a proposal was made with a **"tri window display"** protocol, for effectively managing a variety of CSCW application ..

This report is only a preliminary investigation of the potential for use of CSCW techniques in the software requirement specification stage. Based on the GTCS implemented and the "tri window" protocol proposed in this report, a more detailed investigation can be carried out.

# ACRONYMS

| | |
|---|---|
| CDRL | Contractor Data Requirement List |
| CSCW | Computer Supported Cooperative Work |
| CSCI | Computer Software Configuration Item |
| DFD | Data Flow Diagram |
| GTCS | Groupware Telecommunication Software |
| HWCI | Hard Ware Configuration Item |
| IORL | Input Output Requirement Language |
| ISDN | Integrated Services Digital Network |
| LAN | Local Area Network |
| MCS | Multipoint Communication Service |
| PMI | Person Machine Interface |
| PPD | Predefined Process Diagram |
| PSL | Problem Statement Language |
| RSL | Requirement Statement Language |
| SADT | Structured Analysis Design Technique |
| SAS | Sensor Actuator System |
| SRS | Software Requirement Specification |
| TCSS | Turbine Control System Software |
| UI | User Interface |

# REFERENCES

1. Computer supported co-operative work : A book of readings

   Edited by Irene Greif.

   Morgan Kaufmann Publishers.

2. WYSIWIS Revised : Early experiences with Multiuser Interfaces

   Stefik M, Bobrow D G, Foster G, Lanning S, Tatar D

   ACM Transactions on Office Information Systems

   April 1987. pp 147-167

3. CSCW (1986) Computer Supported Cooperative Work(CSCW '86)

   Proceedings Austin TX, pp 147-174

4. NLS Teleconferencing Features: The Journal and Shared Screen Telephoning

   Engelbart D C

   Proceedings of 1975 IEEE computer society conference, Washington DC.

   pp 173-176.

5. The user interface of a personal calender program

   Greif I

   Human Factors and Interactive Computer Systems

Proceedings of the NYU Symposium on User Interfaces 1984

Norwood,NJ. pp 207-222

6. "QUILT : A Collaborative Tool for Co-operative Writing"

Fish R, Kraut R, Leland M, Cohen M.

Proceedings of the ACM SIGOIS Conference, 1988. pp 30-37

7. Shared work spaces for group collaboration : An experiment using Internet and Unix inter process communication.

Abdel-Wahab H M, Guan S, Nievergelt J

IEEE Communications magazine Nov 1988, pp 10-16

8. MAEstro - A distributed Multimedia Authoring Environment : research project.

George D Drapeau - Stanford University, drapeau @air.stansford.edu

Howard Greenfield - Sun Microsystems, howardg @sun.com.

9. A Communication - Based Framework for Group Interfaces in Computer Supported Collaboration

Francis J Lim, Izak Benbasat

Faculty of Commerce and Business Development

University of British Columbia.

Presented at the 24th Hawai conference on system sciences, Kauai, Hawai.

January 8-11, 1991.

10. Distributed Editor : A toolkit for collaborative
Editing of files in Distributed Environments :
Prakash A, Knister M J
Report No CSE-TR-45-90
University of Michigan - Comp Science and Eng

11. Collaboration awareness in support of collaboration transparency : Requirements
for the next generation of shared window systems -
Chris Lauwers, Keith A Lantz.
Computer Human Interface(CHI) proceedings 1990, pp303  pp311

12. User interface requirements for face to face groupware-
Mary Elwart-Keys, David Halonen, Marjorie Horton, Robert Kass,Paul Scott.
Computer Human Interface(CHI) proceedings 1990, pp 295 - pp 301.

13. Beyond the chalkboard: Computer supported collaboration and problem solving
in meetings -
Stefik M, Foster G, Bobrow D G, Kahn K, Lanning A, Suchman I
Computer supported co-operative work: A book of readings, Morgan Kaufmann
publishers, edited by I Greif, pp 335 - 366.

14. An Interactive Knowledge-Based System for Group Problem Solving

Mildred L G Shaw

IEEE Transaction on Systems, Man, and Cybernetics

Vol 18,No 4 July/August 1988.

15. LIZA : An extensible Groupware Toolkit

Gibbs S J

MCC, Software Technology Program Austin, Texas

Computer Human Interface(CHI), proceedings May 1988, pp 29-35.

16 As We May Think

Vannevar Bush

Computer Supported Cooperative Work - A Book Of Readings, Morgan Kaufmann publishers, pp 17-34.

17. Data Sharing In Group Work

Greif I, Sarin S

Computer Supported Cooperative Work - A Book Of Readings, Morgan Kaufmann publishers, pp 477-508.

18. A Version Server for Computer-aided-design data.

Katz R H, Anwarrudin M, Chang E

Proceedings of 23rd ACM/IEEE Design Automation Conference.

ACM New York 1986, pp 27-33.

19. An object server for an object oriented database system.
Skarra A H, Zdonik S B,Reiss S P
Proceedings of International Workshop on Object Oriented
Database Systems - Sept 1986, pp 196-204.

20. Requirements Engineering Environments :
Software Tools for Modelling User Needs -
William Rzepka, Yutaka Ohno .
IEEE Computer, Vol 18(4), April 1985, pp 9-12.

21. A Taxonomy of Current Issues in Requirements Engg -
Gruia - Catalin Roman
IEEE Computer, Vol 18(4), April 1985, pp 14-21.

22. Applications and Extensions of SADT -
Douglas T Ross
IEEE Computer, Vol 18(4), April 1985, pp 25-35.

23. SREM at the Age of Eight: The distributed Computing Design System

Mack Alford

IEEE Computer, Vol 18(4), April 1985, pp 36-46

24. Specification Based Software Engineering with TAGS -

Gene E. Sievert and Terrence A. Mizell

IEEE Computer, Vol 18(4), April 1985, pp 56-65.

25. Embedded Computer System Requirements Workshop -

Stephanie M White, Jonah Z Lavi

IEEE Computer, Vol 18(4), April 1985, pp 67-70.

26. Transformation of a Semi-formal Specification to VDM -

Juliette D'Almeida, Achutan R, Radhakrishnan T, Alagar V S

Department of Computer Science, Concordia University, Montreal, Canada.

27. An Integrated Approach to Software Engineering

Pankaj Jalote

Springer-Verlag, Newyork 1991.

28. Utilizing Interactive Multimedia to support knowledge based development -

James D Palmer and Peter Aiken

5th Annual knowledge based software assistant conference, New York, Sept

1990, pp 105-119.

29. Computer Supported Cooperative Work - A Book of Readings
Edited by Greif I
Morgan Kaufmann Publishers.

30. DOD Standard 2167A

31. Applications of Information networks
Licklider J C R, Albert Vezza
CSCW : A Book Of Readings, Edited by Greif I
Morgan Kaufmann publishers.

32. Interactive Human Communication
Alponse Chapanis
CSCW : A Book of Readings, Edited by Greif I
Morgan Kaufmann Publishers.

33. The MCS Protocol
Bulkovshteyn T (BNR), Paul Lebel(Bell Canada)
BNR internal Report

34 GTCS Professional System

Bell Canada 1988

35. Integrated Services Digital Network

Stallings W

IEEE Computer Society Press

36 Improving Software Productivity

Bohem B W

IEEE Computer, 20(9), Sept 1987, pp 43-57