Chapter 4

Requirement Engineering

4.1 Introduction

In this work Teleteaching is taken as a case study. Teleteaching can be regarded as the process of distance learning, which provides a virtual class room environment. Here administrator serves as the Distance Learning Organization (DLO). A Teleteaching lecture should be as close as possible to a conventional lecture to guarantee the same success in teaching learning process. It has a target of development and it adopts special methods and techniques for the enhancement of quality and effectiveness of the teaching system. Many synchronous and asynchronous events are associated with this system. Lectures, Test, Viva are some of the instances of synchronous session whereas Study material upload, on demand lecture download, assignment upload represent the asynchronous events. Preliminary design must be done according to the requirements specified by the stakeholders, based on which detailed design is to be carried out further. In this research work i* diagram is used for early requirement analysis followed by means end analysis which are the standard and well established tool for agent based requirement analysis. Then Ontology is used for categorization of different elements and the Conceptual Graph(CG) mainly for the ease of natural language processing. CG has played an important role for natural language processing. Though, for the requirement engineering, the means end analysis is enough to place the stakeholders requirements, still there is a need for a tool
which can process natural language, compatible with requirements. For the verification, comparison of natural language with the means end analysis could have been a chaotic job as finally some concepts for this comparison are needed. So CG is introduced as the intermediary language which can be compared with the natural language easily.

4.2 Requirement Engineering of Teleteaching

4.2.1 Requirements Gathering from Stakeholders

There are ten crystallized requirements of Teleteaching system from different stakeholders are considered. The requirements are as follows.

n1. Facility of showing Teaching aids to all the students.

n2. To make a reliable communication, student should receive teachers audio/video and teacher must get students audio/video.

n3. Teacher must be able to access students workspace (WS).

n4. Teacher must be able to see all the students i.e. students audio/video must be visible to teacher.

n5. Teacher must be able to upload study material to facilitate the event of study material upload.

n6. Student must be able to see teachers workspace (WS).

n7. Student must be able to grab teachers attention by sending message to teachers message box.

n8. Student must be able to download study material to facilitate the event of On demand lecture download.

n9. Assignment upload facility must be there for submission of assignments done by students.
4.2.2 Requirement Analysis of the System

4.2.2.1 i* diagram

i* model offers actors (agents, roles, or positions), goals, resources and actor dependencies as primitive concepts for models used in different phases of software development. In i* (which stands for distributed intentionality), stakeholders are represented as (social) actors who depend on each other for goals to be achieved, tasks to be performed, and resources to be furnished. Actors can be humans, hardware and software, or combinations thereof. Actors are taken to be inherently autonomous - their behaviors are not fully controllable. The i* framework includes the strategic dependency model for describing the network of inter-dependencies among actors, as well as the strategic rationale model for describing and supporting the reasoning that each actor goes through concerning its relationships with other actors. It is an abstraction which is used to refer to an active entity that is capable of independent action. Reasons for using i* framework for requirements analysis is that, i* provides the possibility to achieve information in an early phase of the software engineering process. In former days UML was used to make information visible, but as UML often focuses on organizational objects, which are not so important in the early phase, when the emphasis should be on helping stakeholders to gain better understanding of the various possibilities for using information systems in their organizations. i* models offer a number of levels of analysis, in terms of ability, work-ability, viability and believability. Figure 4.1 depicts the proposed i* diagram. There are three types of elements, such as Actor(Student, Teacher, Virtual University and External Agent), Hardgoal(Providing Distant Learning and Providing Teleteaching Environment), Softgoal(Location Independence of Services) and Resource(Remuneration, Course fee and Public Information). The elements are connected to each other depending on their dependency among each other.

The specification of the proposed i* diagram is as follows:
Actor: Student.
Actor creation condition: Registration.Student → Registered Student.

Actor: Teacher.
Actor creation condition: Registration.Teacher → Registered Teacher.

Actor: Virtual University.
Actor creation condition: Virtual University.Requirements = Provide distance learning.

Actor: External Agent.
Actor creation condition: External Agent ∈ Virtual University.

Dependum: Location independence of services.
Type: Softgoal.
Mode: Should be provided.
Depender: Student.
Dependee: Virtual University.Fulfillment condition: Infrastructure.Virtual University= sufficient.

Dependum: Provide distance learning.
Type: Hardgoal.
Mode: Should be insured.
Depender: Student.

Figure 4.1: i* Diagram
Dependee: Teacher.
Fulfillment condition: i) Infrastructure.Virtual University= sufficient.
   ii) Teacher.ability=OK.

Dependum: Provide Teleteaching environment.
Type: Hardgoal.
Mode: Should be provided.
Depender: Teacher.
Dependee: Virtual University.
Fulfillment condition: Infrastructure.Virtual University=OK.

Dependum: Remuneration.
Type: Resource.
Mode: Should be satisfactory.
Depender: Teacher.
Dependee: Virtual University.
Creation condition: Teacher.Registration = True.

Dependum: Course fee.
Type: Resource.
Mode: Should be paid for registration.
Depender: Virtual University.
Dependee: Student.
Creation condition: Student.FeesPaid → Registration.

Dependum: Public Information.
Type: Resource.
Mode: Should be Available as public.
Depender: External Agent.
Dependee: Virtual University.
Creation condition: external Agent. access(information)= True.
4.2.2.2 Means End Analysis

Means-end analysis used to describe how goals, that are defined in i* model, are in fact achieved. Means-ends analysis aims at identifying goals, plans, resources and softgoals that provide means for achieving the goal (the end). Early requirements analysis ends in this means end analysis. It mainly consists of two types of links. Each task or actor connected to a goal by a means-ends link which is an alternative means for achieving the goal. It is also called Dependency link. Another link, called Decomposition link, which is defined as what other sub-elements needs to be achieved or available in order for a goal or task to be performed.

As per the application, these links are not enough to describe the system. So there is a need to extend it with two more links as per the requirements. They are universal access link(denoted by $A$) and specialized (access a particular instance) access link(denoted by $A(ui)$). Access link semantically specifies the admittance of a means end element to another. What types of access, is not specified by link A, it is used universally. More specialized access link, where the instance of accession can be specified, is symbolized using $A(ui)$ link. An element may have only read access to another element, or may have right to access both audio and video, etc. So depending on the propose system, this instance of access can be of seven types which are specified explicitly in the formalization of extended Means End analysis. The formalization is as follows.

**Formalization of Means End analysis**

In Means End Analysis the Means End elements can be defined by,

\[
\text{MA: \{ Hardgoal, Softgoal, Task, Resource, Actor \}}
\]

\[
\text{\{ Dependency, Decomposition, Access, Access\}}
\]

4.2.2.2.1 Means End Analysis: Providing Distance Learning

Figure [4.2] depicts the means end analysis for providing distance learning where this goal is subdivided further and related with other type of elements using different nodes and links that must conform to its formalization stated above.
4.2.2.2 Means End Analysis: Location Independence of Services

In the same way as mentioned above, figure 4.3 shows the means end analysis of *location independence*
of services where this softgoal is decomposed further using different nodes and links depending upon their relationships.

Figure 4.3: Means End Analysis for Location Independence of Services
4.2.2.2.3 Means End Analysis: Providing Teleteaching Environment  The means end analysis for Providing Teleteaching Environment can be obtained by merging the means end analysis: Providing Distance Learning and Location Independence of services. Therefore, it has not been shown explicitly here.

4.3 Ontology for Teleteaching System

Necessity to switch to ontology from the Means End Analysis is to categorize the means end elements and make a vocabulary out of it. For the final SRS, a hierarchy of concept with semantic catalog is needed for evaluation the semantic gap between the concept. The vocabulary and the hierarchy helps to define the semantic distance between two concepts within a particular domain. Evaluation of this distance is needed for the requirements verification of the system. So Ontology is used here to define the elements semantically (as concepts) and make a hierarchy of it.

4.3.1 Ontological Hierarchy for Teleteaching System

In the figure 4.4, the ontological hierarchy for Teleteaching system is shown to categories all the concepts, relation, keywords of the system, which is required for the requirements verification of the system. Here, use of ontology will give flexibility to add more requirements dynamically without changing the whole system. In the ontological hierarchy, different components are classified using different symbols and are subdivided further. Each component is associated with a label value. More generalized label has less numbered label and more specified label has higher valued label. These values will decide the semantic distance between different elements.
4.4 Algorithm for Converting Means End Graph to Conceptual Graph

Conversion of the means end elements to the elements of conceptual graph is needed here. As means end analysis is not implicitly capable of processing natural language, whereas Conceptual graph has the inherent capability to be compared with natural language. Each Concept-Relation-Concept combination gives a semantic interpretation to natural language. So CG is an essential tool for this research work.
4.4.1 Conceptual Graph

In CG, the rectangles are called concepts, and the circles are called conceptual relations. An arc pointing toward a circle marks the first argument of the relation, and an arc pointing away from a circle marks the last argument. If a relation has only one argument, the arrowhead is omitted. If a relation has more than two arguments, the arrowheads are replaced by integers 1,...,n.

4.4.2 Mapping of Means End Analysis Nodes to Conceptual Graph Nodes

The elements of Requirement Conceptual Graph are,

\[ \text{RCG} = \{ \text{Entity/Concept} \} \]

4.4.3 Algorithmic Steps

Input: MEG (Means End Graph)

Output: RCG (Conceptual Graph)

Algorithmic Steps:

Function Process()

Function Conversion()

Function node(n_i) // n_i = any node in Means End Graph.
if input.MEG.element==Hardgoal||Softgoal||Task||Resource||Actor then
    x=element;
    output.RCG.element = “x” concept;
end if

EndFunction

Function link(li) // l_i = any link in Means End Graph
if input.MEG.element==link then
\[ x = l_i; \]

if \( x == \) “decomposition” link \( \textbf{then} \)
    \( y = \) “Consists of” link;
end if

if \( x == \) “decomposition to task” link \( \textbf{then} \)
    \( y = \) “Can b achieved by” link;
end if

if \( x == \) “dependency” link \( \textbf{then} \)
    \( y = \) “Depends upon” link;
end if

if \( x == \) “universal access” link \( \textbf{then} \)
    \( y = \) “Acces” link;
end if

if \( x == \) “specialized access” link \( \textbf{then} \)
    \( y = \) “Can access ‘ui’ instance” link;
end if

output. RCG.element=y;
end if
EndFunction
EndFunction

Function creation() //n=any arbitrary node
if \( n.\text{level} ! = \) processed \( \textbf{then} \)
    Function node(n);
end if
if \( l.\text{source} == n \) then
    if \( l.\text{level} ! = \) processed \( \textbf{then} \)
        Function link(l);
        l.\text{level}=\text{processed};
        n=l.\text{sink};
        Function creation(n)
    end if
end if
end if
if l.sink == n then
    if l.level != processed then
        Function link(l);
        l.level = processed;
        n = l.source;
        Function creation(n)
    end if
else
    n = NULL;
    break;
end if
EndFunction
EndFunction

Complexity:
For conversion of each element from Means End Graph to CG, the time complexity is $O(1)$. Therefore, for conversion of $n$ elements the time complexity is $n \times O(1) = O(n)$.

4.4.4 Conceptual Graph: Providing Distance Learning

Figure 4.6 gives the graph representation of the CG *Providing Distance Learning*. The table where mapping details of conceptual graph from means end analysis is specified is consulted first. Then following the algorithms steps the MEG is converted to the corresponding CG.

4.4.5 Conceptual Graph: Location Independence of Services

In the same way as figure 4.6 the CG for *Location Independence of Services* is created in figure 4.7.
### 4.4.6 Requirement Conceptual Graph

Depending on the conceptual graph merging rule, these two conceptual figure 4.6 and figure 4.7 graphs are merged in figure 4.8 which is the graph representation of the requirements of the whole system.

### 4.5 Requirements Verification

Suppose there is a set of requirements $A = \{a_i \mid i = 1, 2, 3, ..., n\}$, specified in the natural language by the stakeholders, and there is a set of requirements $S = \{s_i \mid i = 1, 2, 3, ..., n\}$.
1, 2, 3, ..., n}, specified in the SRS (System Requirements Specification) document by the system analyst.

**Traceability**

The proposed method of requirement analysis satisfies the Requirement Traceability if, \( \forall i \exists j \ a_i \in A \ s_j \in S \).
4.5.1 Verification Method and Corresponding Metric

Here the aim is to develop a mechanism, which will verify the traceability of the systems requirements. Metric value will measure the percentage of traceability. The overall framework for this mechanism can be depicted in figure 4.7. Requirements in natural language and the requirement conceptual graph are taken as input to the verification tool. Depending upon the algorithm, the output will conclude whether the requirements are traceable or not.

**Verification Tool** For the requirements verification process, the requirements conceptual graph is represented in a matrix form. Let us assume that the matrix is $RC_{ij}$ by representing all the concepts in row and column wise, where $i = j = n$ (i.e. it will be a square matrix, where $n$=number of concepts present in the Requirements conceptual graph). The value $RC[i][j]$ = relation between the concepts in the $i^{th}$ row and $j^{th}$ column. Now the matrix has certain rules,

- $RC[i][j] = \text{NULL}; \text{ for } i = j$
- $RC[i][j] = R'$ where $R$ is a relation (≠ NULL) between $i^{th}$ and $j^{th}$ concept present in the Requirement conceptual Graph.
Traceability Metric "TM" will compute the percentage of requirements of the system are traceable.
4.5.2 Verification of Traceability and Traceability Metric “TM”

To check the traceability of the systems requirements by checking if there exists any relation at all between two concepts participating in the requirement conceptual graph corresponding to particular requirements in natural language. If there exist one and only one relation between the concepts of a particular requirement which is semantically same, then that particular requirement is traceable. Else, null value leads the systems requirements specification to be untraceable.

**Algorithm**

**Input:** i) Requirements in natural language $n_i$.
ii) Requirement conceptual graph in matrix format.

**Output:** Traceability metric $TM$.

**Algorithmic Steps:**

**Step1:** Initialize $t = 0$;

**Step2:** for $i = 1$ to (number of requirements in natural language) do

**Step3:** Process requirements in natural language $n_l$. 

Figure 4.9: Verification framework
Step4: Read the keywords from that natural language and named as $K_1, K_2$.
Step5: Find the agent concept ($AC$) whose requirements it is.
Step6: Let us assume $K_1 = i^{th}$ concept of $RC_{ij}$ and $K_2 = j^{th}$ concept of $RC_{ij}$
Step7: Find the element $RC[i][j]$ from the matrix $RC_{ij}$
Step8: Check the semantic meaning between the natural language $n_i$ and the set of elements ($K_1, K_2, RC[i][j]$)
Step9:
if same then
    $n_i$ is traceable;
else
    not traceable;
end if
Step10: End loop.
Step11: for $i = 1$ to (number of requirements in natural language) do
Step12: if $n_i$ is traceable then $t = t + 1$;
Step13: End loop.
Step14: $TM = (t/ \text{number of requirements in natural language}) \times 100\%$
Step15: STOP.

Complexity: For verification of $n$ requirements, the time complexity is $O(n)$.

4.5.3 Verification of the Proposed Requirement Specification

For $TM$, $t=10$ as every natural language requirements is verified manually whether they have been mapped or not.

$TM = (10/10) \times 100\% = 100\%$.

So, the proposed requirements specification is fully TRACEABLE.
4.6 Conclusion

Here an unique approach for conceptual graph based requirements specification is presented. After capturing the requirements from the stakeholders in natural language, a standard methodology i* diagram is used for initial requirements analysis and then means end analysis is applied for late requirements analysis of the distributed Teleteaching system. Some algorithms have been developed for converting the means end graph to conceptual graph. Finally the software requirements specification are specified through conceptual graph, which checks the traceability by defining a metric.