Chapter 5

Conceptual Design and Verification

5.1 Introduction

From the requirement analysis phase, there are crystallized, minimal, complete and consistent requirements in the form Requirement Conceptual Graph in figure 4.8 of chapter 4, which acts as input to the design phase. Same way, using ontology incorporating the syntactic theory, the whole system is designed by subdividing into different modules. From there, a Design Conceptual Graph is extracted. Requirement Conceptual Graph and Design Conceptual Graph are two inputs to the verification phase coming from Software Requirement Specification and Design of the System. Then it is compared to each other depending upon each requirement in natural language.

5.2 Teleteaching Module Design

5.2.1 Different Module Design Using Syntactic Theory

Here with these conceptual elements present in the Teleteaching System Hierarchy, eight different session modules are designed using Syntactic theory.
5.2.1.1 Login Module

The conceptual design of Login Module is shown in figure 5.1.

![Login Module Diagram]

Figure 5.1: Login Module

The formal specification of Login Module is as follows:

\[ N = \{ \text{Login page, Authentic agent terminal, Login database, Authentication, Virtual University} \} \]

\[ C = \{ m_4, m_5, ma_1, ca_1 \} \]

\[ \text{SCR} = \{ \langle \text{Login page}, m_4 \rangle, \langle \text{Authentication}, m_5 \rangle, \langle \text{Authentication}, ma_1 \rangle, \langle \text{Virtual University}, ca_1 \rangle \} \]

\[ \text{DST} = \{ \langle \text{Authentication}, m_4 \rangle, \langle \text{Authentic agent terminal}, m_5 \rangle, \langle \text{Login database}, ma_1 \rangle, \langle \text{Login page}, ca_1 \rangle \} \]

\[ \text{Task} = \{ \text{Authentication} \} \]

\[ \text{Page Interface} = \{ \text{Login page} \} \]

\[ \text{System database} = \{ \text{Login database} \} \]

\[ \text{Agent terminal} = \{ \text{Authentic agent terminal, Virtual University} \} \]

5.2.1.2 Registration Module

The conceptual design of Registration Module is shown in figure 5.2.
The formal specification of Registration Module is as follows:

\[ N = \{ \text{Registration page}, \text{Login page}, \text{Provide user information}, \text{Login database}, \text{Database updation}, \text{Virtual University} \} \]

\[ C = \{m_1, m_2, m_3, ma_2, ca_2\} \]

\[ \text{SCR} = \{(\text{Registration page}, m_1), (\text{Provide user information}, m_2), (\text{Database updation}, m_3), (\text{Database updation}, ma_2), (\text{Virtual University}, ca_2)\} \]

\[ \text{DST} = \{(\text{Provide user information}, m_1), (\text{Database updation}, m_2), (\text{Login page}, m_3), (\text{Login database}, ma_2), (\text{Registration page}, ca_2)\} \]

\[ \text{Task} = \{\text{Provide user information, Database updation}\} \]

\[ \text{Page Interface} = \{\text{Registration page, Login page}\} \]

\[ \text{System database} = \{\text{Login database}\} \]

\[ \text{Agent terminal} = \{\text{Virtual University}\} \]

### 5.2.1.3 Lecture Module

The conceptual design of Lecture Module is shown in figure 5.3.

The formal specification of Lecture Module is as follows:

Teacher terminal consists of elements like,

\[ \text{TT} = \{(\text{Students video, Student WS, Teacher WS, Students info, message box, menu})\} \]

Student terminal consists of elements like,

\[ \text{ST} = \{(\text{Teachers video, Student WS, Teacher WS, message box, menu})\} \]
In a Lecture Session, number of Teacher node=1
and number of student node=n
where \( P_1(x,y) \): x is connected to y at time \( t_i \),
x= Teacher node, y= Student node \( \exists x \forall y P(x,y)= true \ |x|= 1 \) and \( |y| = n \);  
N=\{Login page, Teacher terminal, Student terminal, Teacher.Teacher WS, Teacher. Teacher message box, Teacher. mouthpiece,Teacher. students info, Teacher. headphone, Teacher. webcam, teacher. monitor, Student. webcam, Student. monitor, Student.mouthpiece, Student.headphone, Student. Student message box, Student. Student WS, Authentication\}  
C=\{c (c_1u_1, c_2u_2, c_3u_3c_mu_m (where m=number of services exchanged between teacher and student terminal)), m_4, m_6, m_5−i\langle m_5−1, m_5−2, m_5−n (n=number of student terminal logged into the session)), m_8, m_9, m_7, v_1, v_2, v_3, v_4, v_5, v_6, a_1, a_2, a_3, a_4\}\}
SCR = \{ Teacher terminal/Student terminal, c, Login page, m_4, \langle Authentication, (m_{5-i}(i \in 1ton), m_6) \rangle, \langle Teacher terminal, \langle m_9,a_3,v_5 \rangle \rangle, \langle Teacher mouthpiece,a_4 \rangle, \langle Teacher.Teacher WS,v_6 \rangle, \langle Teacher terminal/ Teacher. Teacher message box, m_8 \rangle, \langle Student terminal/ Student. Student message box, m_7 \langle c_1 \rangle, \langle Student terminal, (v_2,a_2) \rangle, \langle Student. webcam, v_3 \rangle, \langle Student.mouthpiece,a_1 \rangle \rangle, \langle Teacher.webcam,v_4 \rangle, \langle Student. Student WS , v_1 \rangle \} \\
DST = \{ Student terminal/ Teacher terminal, c \rangle, \langle Authentication, m_4 \rangle, \langle Teacher terminal, m_6 \rangle, \langle Student Terminali, m_{5-i}(i \in 1ton) \rangle, \langle Teacher. Teacher message box/ Teacher terminal,m_8 \rangle, \langle Teacher. students info, m_9 \rangle, \langle Student. Student message box/ Student terminal, m_7 \rangle, \langle Teacher terminal, (v_6,a_4,v_4) \rangle, \langle Teacher. webcam, v_5 \rangle, \langle Student. headphone, a_2 \rangle, \langle Student terminal,(v_3,a_1,v_1) \rangle, \langle Student. monitor, v_2 \rangle, \langle Teacher. headphone, a_3 \rangle \} \\
Task = \{ Authentication \} \\
Agent terminal = \{ Teacher terminal, Student terminal \} \\
Page Interface = \{ Login page \} \\
Teaching Resources = \{ Teacher.Teacher WS, Teacher. Teacher message box, Teacher. mouthpiece, Teacher. students info, Teacher. headphone, Teacher. webcam, teacher. monitor, Student. webcam, Student. monitor, Student. mouthpiece, Student. headphone, Student.Student message box, Student. Student WS \} \\
Connection c consists of set of connectivity defined as, \\
c = \{(c_1u_1,c_2u_2,c_3u_3c_mu_m)\}, here in the propose application m=8. \\
and “c” can be defined as, 

5.2.1.4 Test Module 

As the component used in test and viva module are same as that of the lecture module. Their connectivity and syntactic specification of these modules are also same. So its specification is not shown explicitly here in order to avoid redundancy. The conceptual design of Test Module is shown in figure 5.5.
5.2.1.5 Viva Session Module

As mentioned earlier, the syntactic specification of this module is also same as the lecture module. So it is omitted here and can be checked from lecture module. The conceptual design of Viva Session Module is shown in figure 5.6.

---

**Figure 5.4**: Connectivity Link

**Figure 5.5**: Test Module
5.2.1.6 On Demand Study Material Download Module

The conceptual design of On Demand Study Material Download Module is shown in figure 5.7.

The formal specification of On Demand Study Material Download Module is as follows:

\[ N = \{ \text{Login page, Student terminal}, (\text{where } i \in 1 \text{ to } n; n = \text{number of logged in student for the session}), \text{Authentication, Ondemand lecture download, Checking to download lecture, Study material database, Completion} \} \]
C={m_{5-i}, m_{10-i}} \text{ (where } i \in 1 \text{ to } n; \text{ } n=\text{number of logged in student for the session)}, m_3, m_6, m_a

SCR= \{ \langle Authentication, m_{5-i} \rangle, \langle Student terminal_i, m_{10-i} \rangle, \langle Ondemand lecture download, m_{13} \rangle, \langle Checking to download lecture, m_{16} \rangle, \langle Checking to download lecture, m_{a3} \rangle, \langle Login page, m_4 \rangle \} \\
DST= \{ \langle Student terminal_i, m_{5-i} \rangle, \langle On demand lecture download, m_{10-i} \rangle, \langle Checking to download lecture, m_{13} \rangle, \langle Completion, m_{16} \rangle, \langle Study material database, m_{a3} \rangle, \langle Authentication, m_4 \rangle \} \\
Page Interface= \{ \text{On demand lecture download, Login page, Completion} \} \\
Agent terminal= \{ \text{Student terminal}_i \} \\
Task= \{ \text{Checking to download lecture, Authentication} \} \\
System database= \{ \text{Study material database} \}

5.2.1.7 Assignment Upload Module

The conceptual design of Assignment Upload Module is shown in figure 5.8.

Figure 5.8: Assignment Upload Module

The formal specification of Assignment Upload Module is as follows:

N=\{Login page, Student terminal_i \text{ (where } i \in 1 \text{ to } n; \text{ } n=\text{number of logged in student for the session)}, Authentication, Assignment upload, Uploading assignment, Study material database, Completion\} \\
C={m_{5-i}, m_{11-i}} \text{ (where } i \in 1 \text{ to } n; \text{ } n=\text{number of logged in student for the session)}, m_{17}, m_{14}, m_a

SCR= \{ \langle Authentication, m_{5-i} \rangle, \langle Student terminal_i, m_{11-i} \rangle, \langle Assignment upload, m_{14} \rangle, \langle
5.2.1.8 Study Material Upload Module

The conceptual design of Study Material Upload Module is shown in figure 5.9.

![Study Material Upload Module Diagram](image)

Figure 5.9: Study Material Upload Module

The formal specification of Study Material Upload Module is as follows:

\[ N=\{\text{Login page, Teacher terminal, Study material upload, Authentication, Uploading material, Study material database, Completion}\} \]

\[ C=\{m_4, m_6, m_{18}, m_{19}, m_{20}, m_5\} \]

\[ \text{SCR}=\{\langle \text{Login page, } m_4 \rangle, \langle \text{Teacher terminal, } m_{18} \rangle, \langle \text{Study material upload, } (m_{19}, m_5) \rangle, \langle \text{Authentication, } m_6 \rangle, \langle \text{Uploading material, } m_{20} \rangle\} \]

\[ \text{DST}=\{\langle \text{Teacher terminal, } m_6 \rangle, \langle \text{Study material upload, } m_{18} \rangle, \langle \text{Uploading material, } m_{19} \rangle, \langle \text{Completion, } m_{20} \rangle, \langle \text{Study material database, } m_5 \rangle, \langle \text{Authentication, } m_4 \rangle\} \]
5.2.2 Integration of all the Modules

Following the graph merging rules, the different modules of Teleteaching system, found in the previous section are merged together to get the complete Teleteaching system shown in figure 5.10.

5.3 Algorithm for constructing Conceptual Graph from Syntactic Theory Module

As syntactic theory is not implicitly capable of processing natural language and on the other side conceptual graph has inherent capability for the same, so, there is a need to convert the syntactic theory elements to the elements of conceptual graph. Each Concept-Relation-Concept combination gives a semantic interpretation to natural language. So, CG is an essential tool for this work.

5.3.1 Mapping of Syntactic Theory Elements to Conceptual Graph Elements

The elements of Design Conceptual Graph are:
Figure 5.10: Teleteaching System Module

5.3.2 Algorithmic Steps

Input: Syntactic Theory Module
Output: DCG (Design Conceptual Graph)

Function Process1()

Function node(n_i) // n_i = any node in Syntactic Theory Graph
if input.STG.element== (System Database || Event || Page Interface || Goal || Agent Terminal || Task || Teaching Resource || Session) then
    x=element;
output.DCG.element = “x” concept;
end if

EndFunction

Function link(l_i) //l_i=any link in Syntactic Theory Graph

Figure 5.11: Mapping of Syntactic Theory elements to Conceptual Graph elements
if input.STG.element==“access” link then
    output. DCG.element= “can access” link;
end if
if input.STG.element==“connectivity” link then
    output. DCG.element= “passes multimedia data to” link;
end if
if input.STG.element==“audio stream” link || “video stream” link then
    if input.STG.element== “audio stream” then
        o=audio;
    end if
    if input.STG.element==“video stream” then
        o=video;
    end if
    if Conceptual relation.source== “Agent terminal” concept then
        output. DCG.element= “forwards incoming ‘o’ to” link;
    end if
    if Conceptual relation.source== “Tele teaching resource” concept then
        output.DCG.element= “forward captured ‘o’to” link;
    end if
end if
if input.STG.element==“message passing” link then
    if Conceptual relation. sink ==“Task/Event” concept then
        output. DCG.element= “forwards message for” link;
    else
        output. DCG.element= “forwards message to” link;
    end if
end if
if input.STG.element==“memory access” link then
    output. DCG.element= “access” link;
end if
EndFunction
Complexity:
For conversion of each element from Syntactic Theory Module to DCG, the time complexity is $O(1)$.
Therefore, for conversion of $n$ elements the time complexity is $n \times O(1) = O(n)$.

5.3.3 Design Conceptual Graph

Based on the table specified in figure 5.11 and the algorithm proposed in section 5.3.2, the syntactic theory graph is converted to a conceptual graph and is depicted in figure 5.12.

5.4 Design Verification of the System

5.4.1 Verification Rule

**Input:** Requirements in natural language, Requirement Conceptual Graph, Design Conceptual Graph

**Output:** Requirement conceptual sub-graph and Design conceptual sub-graph semantically same with natural language requirement.

**Algorithmic steps:**

Step1:
for $i = 1$ to 10 do
Take requirement in natural language $n_i$.
end for

Step2: read the keywords from requirement in natural language $n_i$ and named as $k_1, k_2, k_n$. Find the agent concept AC whose requirement it is.

Step3: take $K_i$ and $K_j$ (each pair of keywords) $(i,j \in 1$ to $n)$ and find concept and in RCG.
Step4: If there exists m number of relation like $R_1, R_2, \ldots, R_m$ between $K_i$ and $K_j$, take sub graph corresponding to $n_i$ as

$$
\begin{align*}
K_i & \rightarrow R_1 \rightarrow C_1 \rightarrow R_2 \rightarrow C_2 \rightarrow \ldots \rightarrow R_m \rightarrow K_j
\end{align*}
$$

Figure 5.13: Verification Algorithm subgraph1

Where m=number of intermediate relation between $k_i$ and $k_j$ and $m_1$. 

83
$C_1, C_2, C_{m-1} = \text{any concept in RCG but } C_1, C_2, ..., C_{m-1} \neq K_i \text{ and } K_j.$

\textbf{Step5:} take each subgraph from \textbf{step4}.

\textbf{Step6:} take relation $R_i[i \in 1 \text{ to } m \text{ where } m=\text{number of intermediate relation between } K_i \text{ and } K_j]$ \textbf{Step7:} find the source of $R_i$.

find the destination of $R_i$.

\textbf{Step 8:} if relation. destination = any Teaching resource then design subgraph is like,

\begin{figure}[h]
\centering
\includegraphics[width=0.7\textwidth]{subgraph2.png}
\caption{Verification Algorithm subgraph2}
\end{figure}

Here $m \neq 0$;

P= number of source and $p \geq 1$.

if relation.destination = any Session then  
\begin{verbatim}
    A = “session”; 
    design subgraph looks like
\end{verbatim}

end if

\begin{figure}[h]
\centering
\includegraphics[width=0.7\textwidth]{subgraph3.png}
\caption{Verification Algorithm subgraph3}
\end{figure}

Here $m \geq 0$;

if relation. destination = any Event then  
\begin{verbatim}
    A=“Event”; 
    design subgraph looks like
\end{verbatim}
Here $i \in 1$ to $m$;
for $i \in 1$ to $m$, $\exists i$ do

$C_i = A$;
end for

Step 9: Continue;

5.4.2 Verification Metric

Input: Requirements in natural language, Requirement conceptual graph, Design conceptual graph, Teleteaching system hierarchy.

Output: Satisfiability metric value.

Algorithmic steps:

Step1:

for $i = 1$ to 10 do

Take the requirement in natural language $n_i$
end for

Step2: find out the agent concept AC whose requirement it is.

Step3: take the requirement subgraph $RCG_i$ corresponding to $n_i$.

Step4: find concepts within $RCG_i$ and named as $CR_1$, $CR_2$,

Step5: find $Distance(AC,CR_1)$, $Distance(AC,CR_2)$,

Step6: $DR = \max Distance(AC,CR_1)$, $Distance(AC,CR_2)$,

Step7: take the design subgraph $DCG_i$ corresponding to $RCG_i$.

Step8: find the concepts within $DCG_i$ and named as $CD_1$, $CD_2$.

Step9: find the $Distance(AC,CD_1)$, $Distance(AC,CD_2)$,
Step 10: \( DD = \max \{ \text{Distance}(AC, CD_1), \text{Distance}(AC, CD_2) \} \)

Step 11: Satisfiability of design for a particular requirement
\[ n_i = \left( \frac{DD}{DR} \right) \times 100\% \]

Step 12: Continue.

Step 13: Function Distance \((C_1,C_2)\)
\[
d = \text{Distance}(C_1, \text{CPP}) + \text{Distance}(C_2, \text{CPP});
\]
// CPP = closest common parent of c1 and c2 from ontological hierarchy of Teleteaching.
return d;
EndFunction

Function Distance\((C_1, \text{CPP})\)
\[
R = C1.\text{level} - C2.\text{level};
\]
Return R;
EndFunction

**Complexity:** For finding the verification Metric value of \( n \) requirements, the time complexity is \( O(n) \).

### 5.4.3 Case Study

For n1, a. “Facility of showing teaching aids to all the students”.
AC=Student.
From the ontological hierarchy metric value for concept can be calculated as (from figure 5.17).

\[
DR = \max(0,5)=5, \ DD = \max(0,2,6)=6, \ Satisfiability = (5/6)\times100
\]

For n2, a. “To make a reliable communication, student should receive teacher audio/video and teacher must get student audio/video”.
AC=Student.
From the ontological hierarchy metric value for concept can be calculated as (from figure 5.18).

AC=Teacher.
From the ontological hierarchy metric value for concept can can be calculated as (from figure 5.19).

Figure 5.17: Subgraphs for verification of requirement n1

Figure 5.18: Subgraphs for verification of requirement n2

Figure 5.19: Subgraphs for verification of requirement n2
For n3, a. “Teacher must be able to access student workspace (WS).” AC=Teacher.
From the ontological hierarchy metric value for concept can be calculated as (from figure 5.20).

\[
\begin{align*}
\text{DR} &= \max(0, 6) = 6, \\
\text{DD} &= \max(0, 2, 6) = 6, \\
\text{Satisfiability} &= (6/6) \times 100
\end{align*}
\]

Figure 5.20: Subgraphs for verification of requirement n3

For n4, a. “Teacher must be able to see all the student i.e. students audio/video must be visible to teacher.” AC=Teacher.
From the ontological hierarchy metric value for concept can be calculated as (from figure 5.21).

\[
\begin{align*}
\text{DR} &= \max(0, 6) = 6, \\
\text{DD} &= \max(0, 2, 6) = 6, \\
\text{Satisfiability} &= (6/6) \times 100
\end{align*}
\]

Figure 5.21: Subgraphs for verification of requirement n4

\[
\begin{align*}
\text{DR} &= \max(0, 6) = 6, \\
\text{DD} &= \max(0, 2, 6) = 6, \\
\text{Satisfiability} &= (6/6) \times 100
\end{align*}
\]
For n5, a. “Teacher must able to upload study material to facilitate the event of study material upload.”
AC=Teacher.
From the ontological hierarchy metric value for concept can be calculated as (from figure 5.22).

![Diagram](image)

Figure 5.22: Subgraphs for verification of requirement n5

\[ DR = \max(0,6) = 6, \quad DD = \max(0,4,5,6) = 6, \quad Satisfiability = \frac{6}{6} \times 100 \]

For n6, a. “Student must be able to see teachers workspace (WS).”
AC=Student.
From the ontological hierarchy metric value for concept can be calculated as (from figure 5.23).

\[ DR = \max(0,6) = 6, \quad DD = \max(0,2,6) = 6, \quad Satisfiability = \frac{6}{6} \times 100 \]

For n7, a. “Student must be able to grab teachers attention by sending message to teacher message box.”
AC=Student.
From the ontological hierarchy metric value for concept can be calculated as (from figure 5.24).
Figure 5.23: Subgraphs for verification of requirement n6

Figure 5.24: Subgraphs for verification of requirement n7

DR= max(0,6)=6, DD= max(0,2,6)=6, Satisfiability= (6/6)*100

For n8, a. “Student must be able to download study material to facilitate the event of On demand lecture down load.”
AC=Student.
From the ontological hierarchy metric value for concept can be calculated as (from figure 5.25).

DR= max(0,6)=6, DD= max(0,4,5,6)=6, Satisfiability= (6/6)*100

For n9, a. “Assignment upload facility must be there for submission of assignments done by students.”
AC=Student.
From the ontological hierarchy metric value for concept can be calculated as (from figure 5.26).

DR= max(0,6)=6, DD= max(0,4,5,6)=6, Satisfiability= (6/6)*100

For n10, a. “User authentication must be done by virtual university before allowing
any user to enter any session.”
AC=Virtual University.
From the ontological hierarchy metric value for concept can be calculated as (from figure 5.27).

\[
DR = \max(0, 6) = 6, \quad DD = \max(0, 2, 6) = 6, \quad \text{Satisfiability} = \frac{6}{6} \times 100
\]
5.5 Conclusion

Requirement conceptual graph is considered as the reference from the requirement analysis phase. Here conceptual design of the system is proposed based on the requirement analysis, followed by extraction of a design conceptual graph. The proposed conceptual design of the system demonstrates the static behavior of the system i.e. identifying different components and their in-between connectivity. Verification algorithm and the verification metrics are the most significant contribution here. The satisfiability metric values decide whether the system is designed properly or not by mapping all the requirements. If the metric value does not give 100 percent satisfiability for each requirement, then it is needed to think of better design of each module or changing the ontological hierarchy of the system for betterment.