5.1. Introduction

This chapter describes the method adopted for the development of Class diagrams. The focus is on automating the transition from analysis to design when using UML for object-oriented software development. The work deals with Use case representation using structured English, and Use case textual analysis (noun and verb analysis) using OO design method based on mental heuristics. The inclusion of internal actors as Agents to help the design of database table is also discussed. The methodology to identify classes from Use cases is explained facilitating the phase transition from analysis to design of the SDLC. Also, the building of AI system and the user interface design with multicolours are discussed in detail.

5.2. Representing Use case with structured English

There are many techniques to represent the dynamic behaviour of Use case scenarios, namely:

- State transition diagram,
- Interaction diagram,
- Flowchart,
- Activity diagram, and
- Structured English.

Use case scenarios are used as a means to have better understanding of the system [42]. A Use case should be structured starting with a basic scenario or path called the “happy day” scenario.

A Use case may have another Use case. The <include> relationship is a generalisation. (If ‘A’ is a generalisation of ‘B’, then an instance of ‘B’ will include the behaviour specified by ‘A’. ‘B’ may include ‘A’ at more than one
place. There may be other subclasses for 'A'.) A larger, less common alternative path is handled by the <extend> relationship.

However, a system can only be used as described in the Use cases. The collection of Use cases logically decomposes the functionality of the system [25]. Each Use case imposes certain requirements on the system. This decomposition of the functionality of the system is itself a challenging task. The behaviour of Use case can be described in plain unstructured English or represented by expressions in formal language. Database tables are also used frequently to represent Use cases.

Here in the research work, the selected format for Use case representation is structured English. Pseudo code is also used to express the control flow in the Use case textual descriptions.

The Use case format using structured English should contain the following information [43]:

• The start state described as a precondition,
• How and when the Use case starts,
• The required order (if any) of actions (information transfers),
• How and when the Use case ends,
• The possible end states as post conditions,
• Prohibited paths,
• In-line alternative paths,
• Extracted alternative paths <extend> and inherited behaviour <include>,
• System interaction with actor(s),
• The information that must be retained (remembered) by the system to produce appropriate responses which are logically correct as viewed by the actor(s).

The above information can be used to construct a detailed information model of the system, how and when the concepts of problem domain are handled, and information is exchanged in a Use case.
Care should be taken while analyzing the Use cases, on the aspects given below:

• Non-functional requirements should be considered [44].
• Related information can be modelled using:
  Relational tables, and
  Object Constraint Language (OCL) of UML.

As an illustration, the structured English notation for the Use case "Consult doctor" occurring in the Hospital Patients Care Study is given as below:

1. If patient requests
   receptionist registers;
   else
   responds to enquiry.

2. If doctor available
   receptionist allots time for consultation;
else
   fixes an alternate date.

3. Consultation on condition:
   Case 1: normal
   Do a prescription.
   Case 2: serious
   Generate a scan report.
   Case 3: immediate
   Prescribe tests and drugs.
   Case 4: very serious
   Propose an operation.

5.3. Identification of classes using heuristics.

The author followed the OO design heuristics, based on the mental heuristics of a set of Post graduate students. Case studies like Video Rental System, Library Management System, Railway Ticket Reservation System,
Olympic Games Information, Hospital Patients care study and Electricity Billing System were taken up and solved by the students, who had prior knowledge about Use cases and OOAD concepts without using any CASE tool.

The mental thinking processes of the students in solving the above problems were recorded by conducting interviews. The mental thinking process differs from individual to individual [8]. OO design heuristics based on common human intuition for the identification of classes were formulated based on the above recordings in the form of rule-base.

The Software Requirements Specification (SRS) and the Software Requirements Analysis (SRA) are both communication intensive processes between the software developer and the end user [27, 45]. It is the role of the developer to bring out the needs of the end user, (since the end user is usually not technically competent to convey his/her requirements in a suitable format for software development). Therefore, the mental thinking process of the analyst or developer, who is involved in transforming the requirements specification of a given problem into the related Use cases, is highly important in the identification of classes based on heuristics. It plays a vital role in the entire SDLC. The research work is focused on OO Design concepts along with mental heuristics [i].

Till date, in software industries during the analysis phase, the analyst usually lists out all possible nouns as a rough list manually. The construction of class diagram is essential for the design phase. The establishment of relations, associations among the classes is of high importance in the construction of class diagrams. A class may turn out to be an attribute under certain software requirements. Also the roles of interactions among the classes are difficult to predict. Hence in order to overcome all the problems encountered in the manual construction of class diagrams an automatic approach was taken by the author based on mental heuristics.

The approach with mental heuristics used in this research work differs
from traditional AI heuristics algorithms like A*. A* algorithm is a measure for finding out the goal state for a given function. Hence, the author has not explored the possibility of A* approach [i, ii]. But AI techniques such as rule-based systems are applicable for extracting information from the Use cases. The mental heuristics are recorded using the if-then format.

To start the process of identification of the key domain abstractions using the noun identification technique, there are two stages:

- Identification of classes by all the nouns and noun phrases out of a requirements specification of the system (considered in a singular form).
- Discarding the candidates which are inappropriate for any reason.

A class describes a set of objects with an equivalent role or roles in a system. Objects and their division into classes often derive from one of the following sources.

As an example, for the Hospital Patients care study, the following categories of objects are listed out [xiv]:

1. **Tangible or real-world things**: Hospital
2. **Roles** : Patient, Doctor, Receptionist, Nurse, Lab Technician
3. **Events** : Consultation, Admission, Discharge
4. **Interactions** : Examine

The identified objects fall into the first two categories as listed above. The remaining two categories may help to find name and association between them.

To start with, write or type out potential classes as a rough list. One would probably add and cross off many items before the list becomes reasonably valid. Once an initial list is prepared, brief description of each item
is written to check through the list carefully. It may be useful in finding out,
whether the noun identified is:

- Beyond the scope of the system?
- Referred by the system as a whole?
- Duplication of another class?
- Too vague?
- Too specific?
- Too much tied up with physical inputs and outputs?
- Really an attribute?
- Really an operation?
- Really an association?

5.4. Knowledge Elicitation – Identification of classes

The Hospital Patients Care Study is considered as an illustration for
Knowledge elicitation. The scenarios of the Use cases are spelt out in the
textual description form. They contain knowledge needed to model the system
and continue to the design phase. A first attempt to elicit knowledge is to
analyse the words in the Use case description, and figure out what UML
symbol might best fit-in each word like Actor, Use case, Class, Object etc.
Table 5.1 shows the knowledge elicitation from the Use case – "Admit Patient".

This kind of table is commonly used in knowledge engineering, human
computer interface practice and philosophy. It is called as the Abbott technique
[43]. It provides only guidance, and it still needs the creative perception of a
developer to make progress. The Use cases, or really scenarios, are written
with informal descriptions of the problem domain.

In this research, the focus on nouns as a means for knowledge elicitation
is dealt with. The noun ‘patient’ is an actor playing a role; patient and doctor
are roles that people play. A role may not map on to a class. Hospital is a
tangible object.
The verb ‘admit’ has been converted into a noun ‘admission’ in order to represent the process of registering a patient. Admitting a patient and discharging are events [x]. Examination is an interaction between a patient and a doctor.

Further analysis is therefore necessary. Once a rough list of potential candidates for Classes is formed, some of them may be eliminated by raising questions based on mental heuristics such as:

1. Is this object a duplicate of some previously identified object?
2. Is this object within the scope of the proposed system?
3. Is this object too broad and includes many things?
4. Is it a vague object that cannot be crystallized?

Table. 5.1. Knowledge Elicitation from the Use case “Admit patient”

<table>
<thead>
<tr>
<th>Part of Speech</th>
<th>Model Component</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proper noun</td>
<td>Object</td>
<td>Siva (a patient)</td>
</tr>
<tr>
<td>Improper noun</td>
<td>Class</td>
<td>hospital</td>
</tr>
<tr>
<td>Doing verb</td>
<td>Operation</td>
<td>A hospital wishes to store info.</td>
</tr>
<tr>
<td>Being verb</td>
<td>Generalization</td>
<td>There are many labs</td>
</tr>
<tr>
<td>Having verb</td>
<td>Composition</td>
<td>nil</td>
</tr>
<tr>
<td>Modal verb</td>
<td>Condition</td>
<td>The hospital needs</td>
</tr>
<tr>
<td>Stative verb</td>
<td>Invariance</td>
<td>nil</td>
</tr>
<tr>
<td>Adjective</td>
<td>Attribute</td>
<td>nil</td>
</tr>
<tr>
<td>Adjective phrase</td>
<td>Association</td>
<td>Laboratory examination</td>
</tr>
<tr>
<td>Adjective phrase</td>
<td>Operation</td>
<td>Schedule of administration</td>
</tr>
<tr>
<td>Transitive verb</td>
<td>Operation</td>
<td>The hospital prepares a list</td>
</tr>
<tr>
<td>Intransitive verb</td>
<td>Event</td>
<td>Admitting a patient</td>
</tr>
</tbody>
</table>
5.5. Object-Oriented Design (OO Design) Heuristics

This research work includes description of the Software tool developed by the author to automate the transition from OOA to OOD. This work, also takes mathematical modeling of software specifications used in Structured Analysis for further knowledge elicitation from the OOA model.

Here a hybrid method, which is a combination of Use case driven approach and object-oriented method based on mental heuristics is used.

The author has developed a user interface with multicolours, to enhance productivity and quality which results in better OO design [i, ii].

The mental intuition is useful in identifying the classes. Since the mental thinking process varies for the individuals, the author recorded the mental heuristics for the OOAD, for the identification of classes. Based up on mental heuristics, the selection of the candidate classes and the rejection of unwanted classes by making use of user interface become easier than the manual identification of classes.

The life cycle approach used in this research is spiral model dealing with analysis, preliminary data design, reanalysis, redesign data and design classes. It is difficult to completely specify the software requirements with the use of external actors in Use cases. The data design produces the tables that comprise the database. This is similar to entity classes in Objectory. If the implementation is based on object-oriented databases, then this step is unnecessary.

The author feels that at least in the medium term, relational databases are here to stay and there will be a layer of object-orientation on top of it—called Object Relational Database. The database tables form internal actors and to distinguish them from external actors, they are designated as Agents.
Fig. 5.1. Use case diagram for Hospital Patients Care system with database as Agent
Fig. 5.1 gives the Use case diagram for the Hospital Patients system with the inclusion of the database as internal actor, designated as Agent.

Another reason for using the iterative approach is to get the basic flow first and alternative and exceptional flows (or, at least some of them) later. Usually the actor is involved in more than one sequence of operations. Each sequence is called a scenario. Thus the Use case is a set of scenarios. It may be difficult to include all the scenarios in the first iteration. An important ingredient of Use case descriptions is the post condition after the transaction has been completed. Naturally, there is pre condition too. It is possible to state pre and post conditions using set theory and two valued logic, as seen in Chapter 6.

Also in the OO Design heuristics, the following points should be kept in mind. A list of OOAD steps based on heuristics for the identification of candidate classes from the requirements specification in the form of Use cases is formulated [41].

1. An Actor is dependent on the public interface (external behaviour) of a class or a group of classes that collaborate, but not vice versa.
2. Implementation details should not cloud the design phase.
3. Classes collaborate with each other and provide a service to the Actor concerned. The sequence diagram gives the interaction of Objects through message passing. Excessive sending and receiving of messages should be avoided.
4. A class should ideally capture one and only one key abstraction.
5. Excessive dependency chain should be avoided (i.e., nesting).
6. There should be a close coupling between attributes and operations.
7. If the design model becomes simpler by removing some of the constraints, then build the simpler design model first and then later
superimpose the constraints either using UML or by using Object Constraint Language (OCL).

It is difficult to incorporate many of the above mentioned OO design heuristics in the software tool developed by the author. That is, the heuristics is based on the human thought process, mainly between the developer and the end user; therefore the author focused her attention towards the analysis and design phase of the SDLC. The remaining OO design heuristics are concerned with the implementation and later stages of the software. The author has therefore focused the attention on developing an intelligent editor using different colours for highlighting nouns, verbs etc. [ii, x]

5.6. Illustrative Example – Hospital Patients Care Study

The following description gives brief information about Hospital Patients care system. In Hospital system, information about the patients who are admitted to the hospital is to be maintained. It needs to be stored: viz first name, last name, date of birth, sex, date of admission, referred by whom, date of discharge, reason for admission, details of treatments, details of operation(s) done, if any, time spent in the operation theatre, operation theatre identification, department, laboratory reports, etc. In addition, department name, department head, location of the department, number of beds etc are to be coded. It is necessary to keep records of the visits of doctors to the patients in their care and medicines prescribed and details of their schedule of administration to patients. The laboratories in the hospital like X-Ray, ECG, Ultrasound and Blood Analysis, etc. are to be identified. Doctor referring a patient to the hospital also prescribes the laboratory examination to be conducted on the patient. After such examination, a report about the patient can be prepared based on the analysis of the results of the laboratory tests.
The usual manual procedure for the identification of classes is to take a coherent, concise statement of the requirements of the system and *underline* its *nouns* and *noun phrases*; to identify the words and phrases that *denote things* [46]. This gives a list of candidate classes.

The author used the Use case driven method along with the OO design heuristics. In the Use case driven method, Use cases are used to model the scenarios in the system and specify how the actors interact with the scenarios. Modelling with Use cases is a recommended aid in identifying the objects of a system and is the technique used by the unified approach [47]. For this, the system has to be described in terms of its scenarios, and the analyst can examine the textual descriptions or steps of each scenario to determine what objects are needed for the scenario to occur.

### 5.7. Noun Database Formation

The Word Net Noun database developed by the research group of Princeton University has been taken as reference which consists of 26 noun files [17, 20]. The names of the lexicographer files and their corresponding file numbers are listed in Table 5.2 along with a brief description of each file.

<table>
<thead>
<tr>
<th>File Number</th>
<th>Name</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>adj.all</td>
<td>all adjective clusters</td>
</tr>
<tr>
<td>01</td>
<td>adj.pert</td>
<td>relational adjectives (pertainyms)</td>
</tr>
<tr>
<td>02</td>
<td>adv.all</td>
<td>all adverbs</td>
</tr>
<tr>
<td>03</td>
<td>noun.Tops</td>
<td>unique beginners for nouns</td>
</tr>
<tr>
<td>04</td>
<td>noun.act</td>
<td>nouns denoting acts or actions</td>
</tr>
<tr>
<td>05</td>
<td>noun.animal</td>
<td>nouns denoting animals</td>
</tr>
<tr>
<td>Category</td>
<td>Example</td>
<td></td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>noun.artifact</td>
<td>nouns denoting man-made objects</td>
<td></td>
</tr>
<tr>
<td>noun.attribute</td>
<td>nouns denoting attributes of people and objects</td>
<td></td>
</tr>
<tr>
<td>noun.body</td>
<td>nouns denoting body parts</td>
<td></td>
</tr>
<tr>
<td>noun.cognition</td>
<td>nouns denoting cognitive processes and contents</td>
<td></td>
</tr>
<tr>
<td>noun.communication</td>
<td>nouns denoting communicative processes and contents</td>
<td></td>
</tr>
<tr>
<td>noun.event</td>
<td>nouns denoting natural events</td>
<td></td>
</tr>
<tr>
<td>noun.feeling</td>
<td>nouns denoting feelings and emotions</td>
<td></td>
</tr>
<tr>
<td>noun.food</td>
<td>nouns denoting foods and drinks</td>
<td></td>
</tr>
<tr>
<td>noun.group</td>
<td>nouns denoting groupings of people or objects</td>
<td></td>
</tr>
<tr>
<td>noun.location</td>
<td>nouns denoting spatial position</td>
<td></td>
</tr>
<tr>
<td>noun.motive</td>
<td>nouns denoting goals</td>
<td></td>
</tr>
<tr>
<td>noun.object</td>
<td>nouns denoting natural objects (not man-made)</td>
<td></td>
</tr>
<tr>
<td>noun.person</td>
<td>nouns denoting people</td>
<td></td>
</tr>
<tr>
<td>noun.phenomenon</td>
<td>nouns denoting natural phenomena</td>
<td></td>
</tr>
<tr>
<td>noun.plant</td>
<td>nouns denoting plants</td>
<td></td>
</tr>
<tr>
<td>noun.possession</td>
<td>nouns denoting possession and transfer of possession</td>
<td></td>
</tr>
<tr>
<td>noun.process</td>
<td>nouns denoting natural processes</td>
<td></td>
</tr>
<tr>
<td>noun.quantity</td>
<td>nouns denoting quantities and units of measure</td>
<td></td>
</tr>
<tr>
<td>noun.relation</td>
<td>nouns denoting relations between people or things or ideas</td>
<td></td>
</tr>
<tr>
<td>noun.shape</td>
<td>nouns denoting two and three dimensional shapes</td>
<td></td>
</tr>
<tr>
<td>noun.state</td>
<td>nouns denoting stable states of affairs</td>
<td></td>
</tr>
<tr>
<td>noun.substance</td>
<td>nouns denoting substances</td>
<td></td>
</tr>
<tr>
<td>noun.time</td>
<td>nouns denoting time and temporal relations</td>
<td></td>
</tr>
<tr>
<td>verb.body</td>
<td>verbs of grooming, dressing and bodily care</td>
<td></td>
</tr>
<tr>
<td>verb.change</td>
<td>verbs of size, temperature change, intensifying, etc.</td>
<td></td>
</tr>
<tr>
<td>verb.cognition</td>
<td>verbs of thinking, judging, analyzing, doubting</td>
<td></td>
</tr>
<tr>
<td>verb.communication</td>
<td>verbs of telling, asking, ordering, singing</td>
<td></td>
</tr>
<tr>
<td>verb.competition</td>
<td>verbs of fighting, athletic activities</td>
<td></td>
</tr>
<tr>
<td>verb.consumption</td>
<td>verbs of eating and drinking</td>
<td></td>
</tr>
<tr>
<td>Verb Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>--------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>verb.contact</td>
<td>verbs of touching, hitting, tying, digging</td>
<td></td>
</tr>
<tr>
<td>verb.creation</td>
<td>verbs of sewing, baking, painting, performing</td>
<td></td>
</tr>
<tr>
<td>verb.emotion</td>
<td>verbs of feeling</td>
<td></td>
</tr>
<tr>
<td>verb.motion</td>
<td>verbs of walking, flying, swimming</td>
<td></td>
</tr>
<tr>
<td>verb.perception</td>
<td>verbs of seeing, hearing, feeling</td>
<td></td>
</tr>
<tr>
<td>verb.possession</td>
<td>verbs of buying, selling, owning</td>
<td></td>
</tr>
<tr>
<td>verb.social</td>
<td>verbs of political and social activities and events</td>
<td></td>
</tr>
<tr>
<td>verb.stative</td>
<td>verbs of being, having, spatial relations</td>
<td></td>
</tr>
<tr>
<td>verb.weather</td>
<td>verbs of raining, snowing, thawing, thundering</td>
<td></td>
</tr>
<tr>
<td>adj.ppl</td>
<td>participial adjectives</td>
<td></td>
</tr>
</tbody>
</table>

A two-digit decimal code corresponds to the lexicographer file name. The list of filenames and their corresponding numbers are shown in Table 5.2. Table 5.2, shows a way of obtaining the lexcode for a given noun. The software tool that the author has built can compare the noun collection from Use cases against the 26 lexicographer noun files in hard disk. Prior to this comparison, all plural nouns in Use Cases are converted into their corresponding singular nouns, eliminating all duplicate nouns in order to have easy reference with the Word Net Noun data base, which consists of only singular nouns.

The lexcodes give valuable information. For example, lexcode of 14 in Table 5.2 shows grouping of people or objects; lexcode of 23 represents quantities and units of measure; lexcode of 28 represents time and temporal relations [20]. The textual analysis software built by the author consists of various modules like Binary Search module, Morphy module, Redundant module and Expert module. The various modules are described in the following sections.
5.7.1. Binary Search Module

This module is similar to the syntactic analysis of NLP. In NLP, a parser is used to convert the flat list of words or sentences into a hierarchical structure such as noun and non-noun phrases.

Bin_search() is the primary binary search algorithm to search for key as the first matching item on a line in the file pointed to fp. The delimiter between the key and the rest of the fields on the line, if any, must be a space. A lexcode of '0' is returned if a match is not found.

As soon as the Use case textual description in structured English notation is inputted to the software tool developed by author, it is parsed using the Bin_search() function, using the noun database, consisting of the 26 noun files, as cited in Table 5.2.. So, this module helps to find whether the given word is noun or not. The noun and noun phrase textual descriptions (requirements specification of the system) of a problem domain are considered as candidate classes or attributes.

Also the keywords of the structured English constructs like If... Then... do...While; etc are kept in a separate database [ii] formed by the author. In this process of noun identification, care has been taken that the keywords do not get identified as verb, adjective, adverb etc. This is the most important user interface of the software. Further noun extraction is also made possible using the textual descriptions of Activity diagrams.

Fig.5.2. gives the structured English notation for the input Use case "Admit Patient".
Enter the patient and doctor name.
Enter the date and time.
The receptionist verify the appointment.
If the doctor is available at the specified time then
Consultation is fixed.
Doctor diagnose the patient.
If patient condition is not severe then
patient treated as out-patient.
Medicine is prescribed.
Else
Patient treated as In-patient.
Check for bed status.
various lab test prescribed.
If bed not available then
Admission pending.
Else
Enter the information about patient.
Admission granted.
Else
Sorry, Appointment not granted.

Fig. 5.2. Input Use case “Admit Patient” in Structured English notation

Fig.5.3 gives the structured English notation for the input Use case “Admit Patient” with the dialog box.
Enter the patient name and doctor name.
Enter the appointment time.
The receptionist verify the appointment.
   If the doctor is available at the specified time then
      Consultation is fixed.
   Doctor diagnose the patient.
   If patient condition is not severe then
      patient treated as out-patient.
      Medicine is prescribed.
   Else
      Patient treated as In-patient.
      Check for bed status.
      various lab test prescribed.
      If bed not available then
      Admission pending.
   Else
      Enter the information about patient.
      Admission granted.
   Else
      Sorry, Appointment not granted.

Fig. 5.3. Input Use case “Admit Patient” in Structured English notation (With
Dialog box)
Enter the patient and doctor name.
Enter the date and time.
The receptionist verify the appointment.
If the doctor is available at the specified time then
Consultation is fixed.
Doctor diagnose the patient.
If patient condition is not severe then
patient treated as out-patient.
Medicine is prescribed.
Else
    Patient treated as In-patient.
    Check for bed status.
    various lab test prescribed.
    If bed not available then
        Admission pending.
    Else
        Enter the information about patient.
        Admission granted.
Else
    Sorry, Appointment not granted.

Fig. 5.4. Identification of Noun, Verb, Adverb and Adjective

In Fig.5.4. by making use of the case tool developed by the author, the inputted structured English representation for the Use case "Admit Patient" is parsed to get the noun and non-noun phrases like verbs, adverbs, and adjectives in multicolour [i, ii]. The Binary search module is used for this purpose.

The nouns are highlighted in Red colour (doctor, patient, medicine, receptionist, consultation, name, date, time, and appointment, bed, lab test, admission, etc.).

Verbs in Blue colour (enter, granted, verify, treated, prescribed, fixed, diagnose, etc.).
Adverbs in Magenta colour (as, not, about) and Adjectives in green colour (sorry, available, pending, various, severe, etc.)

5.7.2. Morphy Module

This module is similar to the morphological analysis of NLP, hence the name Morphy module [40]. The Morphy function is used to convert a plural word into singular word. The plural words do not exist in the Word Net Noun lexicographic file. Morphy module works in the following manner. In requirements specification the words are either singular or plural. If the noun word is in plural, it is converted into singular by using this module and this converted singular word is used to verify the noun database. All classes are named by singular noun by convention.

5.7.2.1. Rules of Detachment

In morphological analysis, the interpretations for affixes for words (prefixes and suffixes) may depend upon on the syntactic category of the complete word. The following Table 5.3 shows the rules of detachment used by Morphy function. If a word ends with one of the suffixes, it is stripped from the word, and the corresponding ending like “s”, “x”, “ch” and “sh” are added.

<table>
<thead>
<tr>
<th>POS</th>
<th>Suffix</th>
<th>Ending</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOUN</td>
<td>&quot;s&quot;</td>
<td>&quot;s&quot;</td>
</tr>
<tr>
<td>NOUN</td>
<td>&quot;ses&quot;</td>
<td>&quot;s&quot;</td>
</tr>
<tr>
<td>NOUN</td>
<td>&quot;xes&quot;</td>
<td>&quot;x&quot;</td>
</tr>
<tr>
<td>NOUN</td>
<td>&quot;zes&quot;</td>
<td>Z</td>
</tr>
<tr>
<td>NOUN</td>
<td>&quot;ches&quot;</td>
<td>&quot;ch&quot;</td>
</tr>
<tr>
<td>NOUN</td>
<td>&quot;shes&quot;</td>
<td>&quot;sh&quot;</td>
</tr>
</tbody>
</table>
5.7.3. Redundant Module

This module is used to remove duplication of classes. The redundant function is used to remove the duplicate classes of same meaning by using the noun database. The noun database contains the word (noun) and lexcode of two digit decimal integer corresponding to the lexicographer file. The lexcode groups the nouns of different categories. The categories are man-made objects, body parts, natural objects, quantities, units of measure and so on. The objects may be the categories of people, organisations, structures, physical things, abstractions of people and physical things and so on.

The Fig. 5.5 shows the categorisation of the nouns identified [i, ii,x]. The Redundant module is used for this purpose.

In Fig. 5.5, the nouns, doctor, appointment and name comes under the category Noun.Act.

The noun patient comes under the category Noun.person.
The noun date comes under the category Noun.group.
The noun time comes under the category Noun.tops.
The noun consultation comes under the category Noun.communication.
Fig. 5.5. Categorisation of Noun phrases as per Word Net Noun database

5.7.4. Expert Module

The Expert Module is used to remove the inappropriate classes from the potential class list and it identifies the good candidate class. It identifies good candidate classes from the views of Coad and Yourdon (that represents things or events, devices, roles, sites and organizational units), Shlaer-Mellor (that represents tangible entities, roles, incidence, interactions and specification) and Abott (that identifies noun and noun phrases and discard inappropriate classes).

Bharathiar University
Library
T514
59
From the identified classes, viz. doctor, name, date, time, and free, by making use of the Expert module, the author has selected the following candidate classes, namely, doctor, patient and consultation. The User interface design by author is having options for selecting the candidate classes from the classes that got identified by making use of the software tool. This is shown in Fig. 5.6.
5.8. Automatic Class diagram generation

A User interface is designed to aid the software tool developed for the construction of the Class diagrams. The software tool is written in Visual Basic.

Fig. 5.7 gives the Class diagram for the Use case “Admit patient” with classes as Doctor, patient, consultation [ii, x].

The Class diagram is constructed by incorporating the software tool with Rational Rose.

Fig. 5.7. Class Diagram for the Use case “Admit patient”
5.9. Use case diagrams and Activity diagrams for Hospital Patients Care study (without Agent)

From the Hospital Patients Care study, few Use cases were taken as an illustration. For these Use cases, the name of the Use case, its description, flow of events, its relationship with other Use cases, its Pre and post conditions etc are done. The Use case diagram and Activity diagram are drawn for all these Use cases.

**Use case no:1**

**Name**
Request Consultation

**Brief Description**
A patient (either as an in-patient or out-patient) places a request for consultation to the receptionist.

**Flow of Events**
1. Patient gives details about his abnormalities and submits his previous reports (if any)
2. Seeks consultation
3. Receptionists checks the availability of doctor.
4. If specified doctor is available, the patient can be allowed to register for consultation

**Relationships**
The patient calls this Use case to request for consultation to the receptionist

![Use case diagram for Request Consultation](image.png)
**Fig. 5.9. Activity diagram for the Use case “Request Consultation”**

**Special Requirements**
- Monetary availability for registration

**Preconditions**
- None

**Post conditions**
- Patient is permitted for consultation

Fig.5.8 gives the Use case diagram for the Use case “Request Consultation” and Fig.5.9 gives the Activity diagram.

**Usecase no: 2**

**Name**
- Doctor Appointment

**Brief Description**
- Any patient who is registered in the hospital is referred to a doctor and can consult him/her any number of times. The patient takes an appointment from that doctor in the reception. The patient maybe an in-patient, out-patient or a case of casualty. The doctor could be a an in-house Doctor or a consultant.

**Flow of Events**
1. The registration number is verified in the reception
2. Check the doctor’s appointment list
3. The patient is given appointment on that particular day

Relationships

The receptionist calls this Use case to give appointment to the patient

Fig. 5.10. Use case diagram for the Use case “Doctor Appointment”

Fig. 5.11. Activity diagram for the Use case “Doctor Appointment”
Special Requirements  None
Preconditions  The patient should be registered in the hospital.
Post conditions  The patient’s name is added to the list of appointment for the doctor.

Fig. 5.10 gives the Use case diagram for the Use case “Doctor Appointment” and Fig. 5.11 gives the Activity diagram.

Use case no: 3

Name  Consult Doctor
Brief Description  During the appointment, patient describes her / his abnormalities to the doctor. The doctor performs various preliminary tests.
Flow of Events  1. The patient consults doctor on the appointment day
  2. The patient narrates the abnormalities to the doctor
  3. Doctor undertakes various preliminary tests
Relationship  The patient calls this Use case for consultation with doctor.

Fig. 5.12. Use case diagram for the Use case “Consult Doctor”
Special Requirements: None

Preconditions: The patient should already be given appointment for consultation.

Postconditions: Relevant treatment begins after consultation with doctor.

Fig. 5.12 gives the Use case diagram for the Use case “Consult Doctor” and Fig. 5.13 gives the Activity diagram.

Usecase no: 4

Name: Admit Patient

Brief Description: A patient gets himself registered either as an in-patient, Out-patient or a case of casualty. This information is entered by the receptionist.
Flow of Events

1. The patient gives information about himself like name, address, age, sex and previous reference doctor (if any).

2. The patient is given a registration number.
   If the patient is an inpatient, the bed status is checked and if there are no free beds then the patient is not admitted into the hospital.

3. The visit date and the doctor On Duty (OD) are noted.

4. If the patient comes to take only tests in the hospital, a consultant doctor is not referred to him.

5. Else he is referred to a doctor and a medical department in the hospital.

Relationships
The receptionist calls this Use case to admit the Patient.

Fig. 5.14. Use case diagram for the Use case “Admit Patient”
Fig. 5.15. Activity diagram for the Use case “Admit Patient”

Special Requirements None.
Preconditions None
Postconditions The patient is now admitted in the hospital.

Fig. 5.14 gives the Use case diagram for the Use case “Admit Patient” and Fig. 5.15 gives the Activity diagram.

Use case no: 5

Name Prescribe tests/drugs
Brief Description The doctor may look into the case history of the patients and prescribe tests/drugs.
Flow of Events

1. The doctor logs onto the system using the id and password allotted
2. He checks the case history of the patients and various preliminary tests underwent by patient recently and prescribes tests/drugs if necessary.

**Relationships**

The doctor uses this Use case to prescribe tests/drugs for the patient

![Use case diagram for the Use case “Prescribe tests/drugs”](image)

**Fig. 5.16.** Use case diagram for the Use case “Prescribe tests/drugs”

![Activity diagram for the Use case “Prescribe tests/drugs”](image)

**Fig. 5.17.** Activity diagram for the Use case “Prescribe tests/drugs”

**special Requirements**

The user has to be a doctor

**Preconditions**

The doctor can prescribe tests only to the patients who are being treated by them.

**Postconditions**

The patient goes to pharmacy (to get drugs) or lab (to take tests)
Fig. 5.16. gives the Use case diagram for the Use case “Prescribe tests/drugs” and Fig. 5.17. gives the Activity diagram.

5.9.1 Activity diagram with Agent

The following activity diagrams are drawn after the inclusion of database as Agent with the Use case system for Hospital Patients care study.

Use case no: 1

Name: Add doctor/staff

Brief Description: The system administrator may add a new doctor/staff member to the list of doctors working in the hospital.

Flow of Events

➢ The system administrator has to log into the system with proper identification.
➢ The system administrator enters the information about the doctor/staff member like name, address etc and the department.
➢ The id of the doctor/staff is generated.

Relationships: The administrator calls this Use case to add a new member to the list of doctors/staff in the hospital.

System administrator → Add Doctor / Staff → Database

Agent

Fig. 5.18. Use case diagram for the Use case “Add doctor/staff”
**Fig. 5.19. Activity diagram for the Use case “Add doctor/staff”**

**Special Requirements**  
The operation is permitted only to the system administrator

**Preconditions**  
None

**Post conditions**  
The doctor is now an employee of the hospital

Fig. 5.18 gives the Use case diagram for the Use case “Add doctor/staff” and Fig. 5.19 gives the Activity diagram.

**Usecase no: 2**

**Name**  
Delete doctor/staff

**Brief Description**  
The system administrator can remove the name of a doctor/staff member who has left the hospital.

**Flow of Events**

1. The administrator has to log into the system with proper identification.
2. The administrator enters id of the doctor/staff member to be deleted.

3. The list of doctors and staff members is modified.

The administrator uses this Use case to delete a member from the list of doctors/staff in the hospital.

Fig. 5.20. Use case diagram for the Use case “Delete doctor/staff”

Fig. 5.21. Activity diagram for the Use case “Delete doctor/staff”
**Use case no : 3**

**Name:** Edit doctor/staff info  
**Brief Description:** The information about a doctor/staff member such as salary, contact information etc can be edited.

**Flow of Events:**

1. The administrator has to log into the system with proper identification.
2. The administrator enters the id of the doctor/staff member whose information is to be edited.
3. The administrator enters the modified information.

**Relationships:** The administrator uses this Use case to edit the information of a doctor/staff member of the hospital.

---

**Fig. 5.22. Use case diagram for the Use case “Edit doctor/staff info”**
Fig. 5.23. Activity diagram for the Use case “Edit doctor/staff info”

**Special Requirements:** This operation is allowed only to the system administrator

**Preconditions:** None

**Post conditions:** The record is updated.

Fig. 5.22 gives the Use case diagram for the Use case “Edit doctor/staff” and Fig. 5.23. gives the Activity diagram.

### 5.10. Conclusion

Thus in this chapter, the software tool developed by the author for the identification of classes using OO design based on mental heuristics is discussed in a detailed manner. The following issues listed below are dealt:

- Analysis of the information contained in Use cases to extract candidate classes;
• Use of structured English notation to represent Use case textual descriptions;
• Preliminary data design and amendment of Use cases that relate not only to actors but also to Agents i.e. database tables; Further elicitation of candidate classes is made possible by including Agents in the Use case diagram.
• Activity diagrams to represent the flow of the program. Activity diagram is used to show the internal state of an object. Activity diagram expresses the post and preconditions of the Use case which is helpful for the further knowledge elicitation. Activity diagrams were drawn with and without the inclusion of Agent.
• Identification of candidate classes using the software tool developed by the author to highlight the nouns, verbs, adverbs etc. using multicolours.

Thus the construction of the Class diagrams using the software tool developed by the author is discussed. Thus automating the phase transition from analysis to design, in particular from Use cases to class diagrams is dealt in this research work by making use of the software tool developed. The knowledge elicitation i.e. the identification of classes based on OO design along with mental heuristics from Use cases is done by highlighting the nouns, verbs etc. in multicolours thus enhancing the quality of software.