**CHAPTER 8**

**DETAILED DESIGN**

**8.1 Application architecture**

The Input 🡪 Process 🡪 Output model is followed for each of the following stages:

* Stylistic evaluation and generation of comments
* Display of comments
* Text highlighting
* Adherence to templates
* Conformity to target readability
* Opening an existing file
* Saving an edited file
  + 1. **Stylistic evaluation and generation of comments**

Input Process Output

**Text for evaluation**

**Evaluate against rules of style**

**Comments**

***Fig 8.1.1*** *Generation of comments*

The text is taken for evaluation against a predefined set of rules of style. Comments are generated for every violation of rules of style. Each comment has a priority depending on the importance of that rule of style.

* + 1. **Display of comments**

Input Process Output

**Comments**

**Display of comments as in list in GUI**

**Colour-coded comment list**

***Fig 8.1.2*** *Display of comments*

The comments generated after evaluation of the text should be displayed to the user. They appear as a list. The colour of each comment depends on the priority assigned to it. The user can view comments of only one priority or all comments.

* + 1. **Text highlighting**

Input Process Output

**Clicked comment**

**Text search**

**Highlighting of text**

***Fig 8.1.3*** *Text highlighting*

When the user clicks on a comment, the text to which the comment refers gets highlighted. Even if the text is not visible on the screen, the file scrolls down so that the text is visible when it is highlighted.

* + 1. **Adherence to templates**

Input Process Output

**Template option**

**Displaying the template for user to insert text**

**Text in editor window**

***Fig 8.1.4*** *Adherence to templates*

When the user chooses a template to follow, the template appears along with boxes to the user to enter textual data. Upon completion of this action, the text along with help text appears in the editor window. The user can further edit this.

* + 1. **Conformity to target readability**

Input Process Output

**Target readability**

**Evaluation of readability**

**Readability highlighting**

***Fig. 8.1.5*** *Conformity to target readability*

The user selects a target readability score. The next time the user requests for readability analysis, the results are highlighted if the text does not lie in the target readability range. In case of auto evaluation of readability, the results are highlighted as and when the user types in the editor window.

* + 1. **Opening an existing file**

Input Process Output

**File name**

**Opening file**

**File contents in editor window**

***Fig.8.1.6*** *Opening an existing file*

If the user wishes to open an existing file he may search in the system directory and chose a file. That file is read and its contents are visible in the editor window.

* + 1. **Saving an edited file**

Input Process Output

**Text**

**Transferring contents to a file**

**File with text**

***Fig 8.1.7*** *Saving an edited file*

When the user wishes to save his work, if the file already exists, then it’s contents are replaced by the current text in editor window. However, if the file does not exist, then the user must choose a file name and the directory in which he wishes to save the file. The file data can now be accessed outside the application.

**8.2 Modules**

Modules are major components of design that may translate to a function or a class in code. Enumeration of modules makes the further implementation phase easier.

**8.2.1 Parsing of sentences**

The user types his document in the editor window or opens an existing document. Once he is satisfied with his work, he can ask for his document to be evaluated stylistically. After reading of the text, it is split according to sentences, then words and finally each word is tagged with its appropriate part of speech.

Return tags

End

Tag resulting words

Start

Read text

Split text per ‘\n’

Split result into sentences

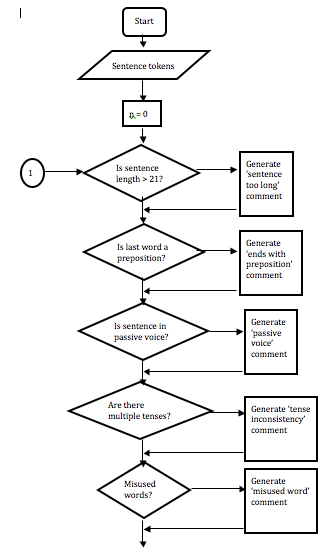
Split result into words

***Fig.8.2.1*** *Sentence tokenization and tagging*

**8.2.2** **Generation of comments**

Once the text is parsed and each word tagged with the appropriate part of speech, comment generation occurs. The comments refer to non-conformance of a sentence to a rule of style. The following types of comments are generated:

* Length of a sentence: The number of words for a moderate sentence length is 21. Beyond this number, the sentence is considered too long.
* Ending with a preposition: Ending a sentence with a preposition is not considered a good practice. Non-conformance to this rule gives rise to an error.
* Usage of passive voice: Using passive voice results in an indirect main object. This makes the sentence weak and should not be used frequently.
* Tense inconsistency: In sentences sticking to a single tense gives clarity of writing. It removes a possibility of confusion.
* Misused words: Several words in English are similar sounding. For example: effect and affect, then and than. One can be mistakenly used for the other.
* Usage of Indian English: Several phrases have been accommodated into Indian English and the original/correct phrase is usually unknown. So, such phrases are detected with alternative phrase suggestions.
* Floweriness: It is the number of descriptive words per sentence. Descriptive words refer to adverbs and adjectives.
* Obscurity: It is the number of rare or infrequent words of English per sentence.



Generate ‘Indian English detected’ comment

Indian English detected?

End

Return comments, and statistics

n == number of sentence tokens?

n = n + 1

Obscurity += rare word count

Floweriness += adjective, adverb count

**8.2.3 Template adherence**

Context creation is done with the help of recommendation context creation is expected to be done by context users. This is done in 2 steps:

* Naming
* Properties identification

8.2.2.1 Naming

* User enters the description of the contexts. Using this description, a term is found for that description by making a call to the reverse dictionary.
* The context description is saved in the context tree/database.
* The reverse dictionary returns n terms (usually 10) that will match the description given by the user.
* If the user agrees with the results retrieved, he uses one of the terms as the context name.
* If the user doesn’t agree with the results retrieved, he may enter his own name or the user will repeat the procedure with some different description. In such a case, the context database will be updated.

8.2.2.2 Properties identification

Context is some situation or circumstance which is useful for the user. To make it useful, right users have to be found. In order to do that, the context is expressed in terms of properties. There are two types of properties.

* Quantitative Properties
* Qualitative Properties

Quantitative Properties are directly measurable.

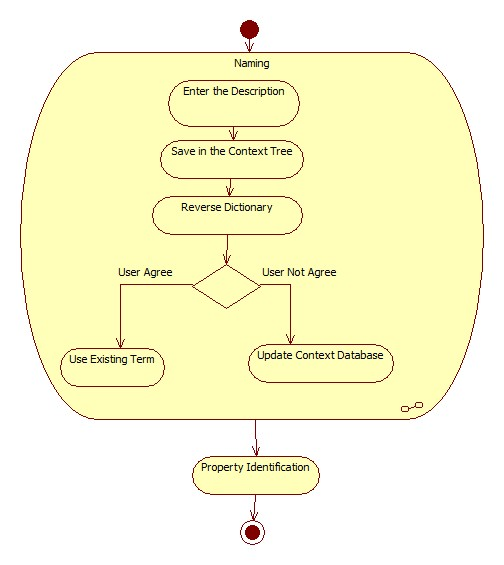
* The system will recommend the available attributes.
* User chooses one of the attribute as the property name.
* Use enters values. These values can be:
  + Tuple, List
  + Range (Numerical)
  + Single value
* Attribute, along with values, is stored in the context database.
* The same process is repeated for the required number of times.

Qualitative Properties are not directly measurable.

* User enters the description of the property.
* Using the reverse dictionary look up, the property names are suggested to the user.
* The property description is saved in the context database.
* If the user accepts any of the suggested names, it is stored in the database.
* If the user doesn’t agree with the results, the user will provide a name, or repeat the same procedure with different description, and the context database gets updated.

The following assumptions are made:

* Reverse dictionary look up can be made.
* Context database is kept updated.
* Description of the context name, along with description of attributes, is important in order to generate congruent networks.



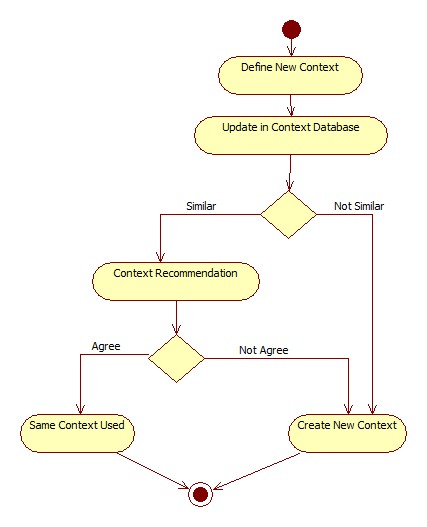
***Fig. 8.2.2*** *Activity diagram for creating of new contexts*

**8.2.4 Readability score**

* When a user wants to define a new context, he enters description of the context.
* This description is compared with the description of the existing contexts in the database.
* If the descriptions are found to be similar, these contexts will be recommended to the user, so that the user can select the existing context.
* Recommendation also includes properties of the context, along with their description.
* If the agrees, the same context is used. A congruent network would have already been generated, and the same congruent network will be used with some updates.

The following assumptions are made:

* Facility exists to compare a context and find similar contexts.
* Facility exists to update congruent networks.



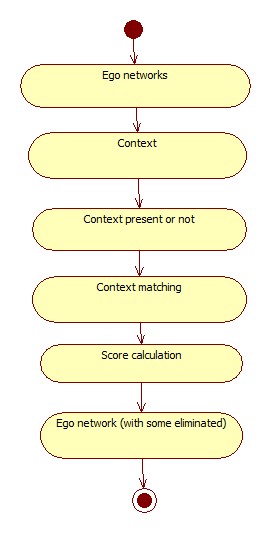
***Fig. 8.2.3*** *Activity diagram for using an existing context*

**8.2.5 Elimination of ego networks**

* For the given context, identify the attributes that describes that context best
* Context matching is done between the attributes of the context and the attributes of the ego node in the egocentric network
* If CA ∩ NA < Threshold, then eliminate the ego network

CA

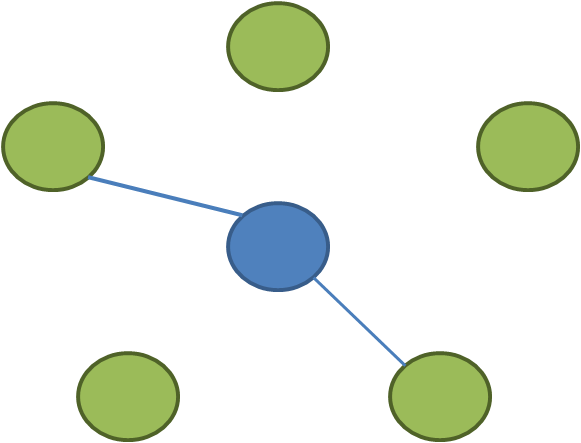
* Only the ego networks whose score is above this threshold value are considered for the next steps.



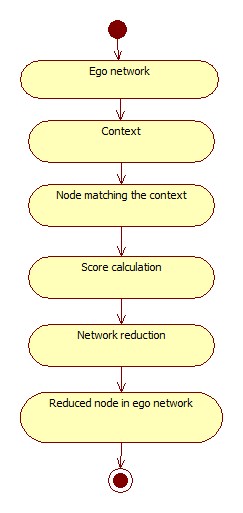
***Fig. 8.2.4*** *Activity diagram for elimination of Ego networks*

8.2.5 Reducing the number of nodes in each of the remaining egocentric network

* For each of the remaining egocentric network
* Nodes whose score is less than the threshold are eliminated and we are left with the ego centric network for the particular context.
* Hence, we derive the context specific egocentric network
* At the end of this step we would be left with certain number of context specific egocentric network



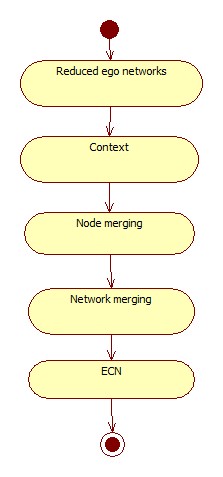
***Fig.8.2.5.1*** *Ego network with nodes reduced*



***Fig.8.2.5.2*** *Activity diagram for reduction of nodes in ego network*

8.2.6 Generation of ECN

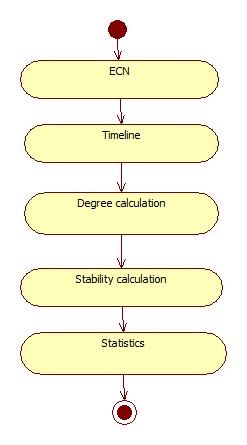
* We consider all the contest specific egocentric networks for this step.
* A network of all the nodes, in the remaining networks, is formed to generate the egocentric congruent network.



***Fig.8.2.6*** *Activity diagram for generation of ECN*

8.2.7 Determine the degree and stability of ECN

* Degree tells us to what extent the nodes of the ECN are congruent with each other
* Stability of the ECN tells us the number of nodes that “join” or “leave” the congruent network, is the specified timeline
* To simulate this, the user profile information is updated, and based on this modification, the ECN regenerated for the same context. The number of nodes that “come into “ECN or “go out” of the ECN, tell us about the stability
* Higher this number, lower is the stability of the ECN

****

***Fig. 8.2.7*** *Activity diagram for measurement of degree and stability of ECN*

**8.3 Database Design**

The number and type of comments are fixed in the application. Another candidate for usage of databases is misused words/phrases. But the number of misused words/phrases is fixed and more will not be added during program execution. Hence there is no reason to bring in databases.

However, data must still be stored. Misused words/phrases are stored in the form of a list of tuples.

Misused words:

[(‘word1’, ‘tag1’, ‘replacementWord1’),

(‘word2’, ‘tag2’, ‘replacementWord2’),

…

(‘wordN’, ‘tagN’, ‘replacementWordN’)]

Misused phrases:

[(‘misusedPhrase1’, ‘correctPhrase1’),

(‘misusedPhrase1’, ‘correctPhrase1’),

…

(‘misusedPhraseN’, ‘correctPhraseN’)]