KDD- SERVICE BASED NUMERICAL ENTITY SEARCHER

by

NAGA SOWJANYA KARUMURI

B.E., Osmania University, 2007

A REPORT

submitted in partial fulfillment of the requirements for the degree

MASTER OF SOFTWARE ENGINEERING

Department of Computing and Information Sciences
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2009

Approved by:

Major Professor
Dr. William H. Hsu
Abstract

Along with identifying the person, organizations and location names in a text, it is equally important to have data about the occurring of the events and the quantity values of the entities. These values would help us understand the timestamps, ordering and persistence of events and amounts in which quantities occur. The present system extracts this kind of numerical information from the given raw text.

The system can extract most common numerical phrases that occur in natural text but there still more numerical phrases that exist and are not being filtered by the present system. To implement this system, Stanford POS Tagger is adopted and a Numerical Phrase Extractor is built based on the output generated by the POS Tagger. The phrases extracted by the Extractor are then given to the Number-Unit/Date Pattern Recognizer that finally display the output in terms of Chunk (numerical phrase), Chunk Type (number/date), Value or Date, Unit, Unit Type, Bound and Second Date if any.

By adding more patterns and functions to the code provided, more number of different numerical phrases can be extracted.
# Table of Contents

List of Figures ................................................................................................................................ ix
List of Tables .................................................................................................................................. x
Acknowledgements ........................................................................................................................ xi
Dedication ..................................................................................................................................... xii

CHAPTER 1 - Vision Document .................................................................................................... 1
Introduction ....................................................................................................................................... 1
  Motivation ........................................................................................................................................ 1
  NLP .................................................................................................................................................. 1
  KDD- Service based Numerical Entity Searcher (KSNES) .............................................................. 2
  Terms and Conditions .................................................................................................................... 2

Project Overview ............................................................................................................................ 4
Introduction ....................................................................................................................................... 4
  Stanford POS Tagger ...................................................................................................................... 4
  Numerical Phrase Extractor ......................................................................................................... 4
  Number-Unit/Date Pattern Recognizer ......................................................................................... 4

Project Goal ..................................................................................................................................... 6
Project Purpose ............................................................................................................................... 6

Application Requirements .............................................................................................................. 6
  ARI 1 [Critical Requirement] ......................................................................................................... 7
  ARI 2 ............................................................................................................................................... 7
  ARI 3 [Critical Requirement] ......................................................................................................... 7
  ARI 4 ............................................................................................................................................... 8
  ARI 5 ............................................................................................................................................... 8
  ARI 6 [Critical Requirement] ......................................................................................................... 8
  ARI 7 [Critical Requirement] ......................................................................................................... 8
  ARI 8 ............................................................................................................................................... 8
  ARI 9 ............................................................................................................................................... 8
Architecture Elaboration Plan ................................................................. 18
Vision Document Revision ........................................................................ 18
Project Plan Revision ................................................................................ 19
Architectural Design .................................................................................. 19
Prototype Development ............................................................................ 19
Test Plan ..................................................................................................... 19
Formal Technical Inspections .................................................................... 19
Formal Requirements Specification ......................................................... 19
Software Production Plan .......................................................................... 20
Test Plan Revision ..................................................................................... 20
Architectural Design Revision ................................................................. 20
Component Design .................................................................................... 20
Final Software Executable .......................................................................... 20
Formal Technical Inspections .................................................................... 20
User’s Manual ............................................................................................ 20
Test Assessment ........................................................................................ 21
Technical Instructions for Reuse and Extension......................................... 21
Project Assessment .................................................................................... 21
CHAPTER 3 - Software Quality Assurance Plan ........................................ 22
Purpose ...................................................................................................... 22
Management .............................................................................................. 22
Organization ............................................................................................. 22
Tasks .......................................................................................................... 22
Responsibilities ......................................................................................... 23
Supervisory Committee ........................................................................... 23
Major Professor ......................................................................................... 23
Developer ................................................................................................. 23
Formal Technical Inspectors ....................................................................... 23
Documentation .......................................................................................... 23
Standards, Practices, Conventions and Metrics ......................................... 23
Documentation .......................................................................................... 23
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coding Standards</td>
<td>24</td>
</tr>
<tr>
<td>Metrics</td>
<td>24</td>
</tr>
<tr>
<td>Reviews and Audits</td>
<td>24</td>
</tr>
<tr>
<td>Testing</td>
<td>24</td>
</tr>
<tr>
<td>Problem Reporting and Corrective Actions</td>
<td>24</td>
</tr>
<tr>
<td>Tools, Technologies and Methodologies</td>
<td>24</td>
</tr>
<tr>
<td>Code and Media Control</td>
<td>25</td>
</tr>
<tr>
<td>Risk Management</td>
<td>25</td>
</tr>
<tr>
<td>Deliverables</td>
<td>25</td>
</tr>
<tr>
<td>CHAPTER 4 - Architecture Design</td>
<td>27</td>
</tr>
<tr>
<td>Introduction</td>
<td>27</td>
</tr>
<tr>
<td>Background</td>
<td>27</td>
</tr>
<tr>
<td>KDD- Service based Numerical Entity Searcher</td>
<td>27</td>
</tr>
<tr>
<td>Package View</td>
<td>27</td>
</tr>
<tr>
<td>Numerical Phrase Extractor Package</td>
<td>28</td>
</tr>
<tr>
<td>Class Description</td>
<td>28</td>
</tr>
<tr>
<td>Number-Unit/Date PatternRecognizer Package</td>
<td>29</td>
</tr>
<tr>
<td>Class Description</td>
<td>29</td>
</tr>
<tr>
<td>Sequence Diagrams</td>
<td>30</td>
</tr>
<tr>
<td>User Interacting with the System</td>
<td>30</td>
</tr>
<tr>
<td>Formal Specification</td>
<td>31</td>
</tr>
<tr>
<td>CHAPTER 5 - Component Design</td>
<td>38</td>
</tr>
<tr>
<td>Introduction</td>
<td>38</td>
</tr>
<tr>
<td>Background</td>
<td>38</td>
</tr>
<tr>
<td>KDD- Service based Numerical Entity Searcher</td>
<td>38</td>
</tr>
<tr>
<td>Package View</td>
<td>38</td>
</tr>
<tr>
<td>Numerical Phrase Extractor Package</td>
<td>39</td>
</tr>
<tr>
<td>Class Diagram</td>
<td>39</td>
</tr>
<tr>
<td>Number-Unit/Date PatternRecognizer Package</td>
<td>44</td>
</tr>
<tr>
<td>Class Description</td>
<td>45</td>
</tr>
<tr>
<td>CHAPTER 6 - Software Test Plan</td>
<td>49</td>
</tr>
</tbody>
</table>
Test Plan Identifier

Introduction

Test Items

Tested Features

General Application Related Items

Features not to be tested

POS tagger Items

Numerical Phrase Extractor Items

Number-Unit/Date Pattern Recognizer Items

Approach

Item Pass/Fail Criteria

Suspension Criteria and Resumption Requirements

Suspension Criteria

Resumption Requirements

Test Deliverables

Testing Tasks

Test Case 1: Application Items

Test Case 2: Patterns and Matching Text

CHAPTER 7 - User Manual

Introduction

Application Setup

Required Software

Running KSNES

CHAPTER 8 - Test Assessment and Evaluation

Introduction

Test Results

Complete Test Results

Test Case 1: Application Items

Overall Results

CHAPTER 9 - Project Evaluation

Introduction
Problems Encountered .............................................................................................................. 59
Determining the Patterns............................................................................................................. 59
Prioritizing the Patterns............................................................................................................. 59
Source Lines of Code (SLOC) .................................................................................................. 59
Project Duration ...................................................................................................................... 60
Lessons Learned ...................................................................................................................... 64
  Eclipse IDE .......................................................................................................................... 64
  Creation of Design Documents .......................................................................................... 64
Future Work ............................................................................................................................. 64
  Adding more patterns to the NPE Package .......................................................................... 64
  Improving NDPR Package ................................................................................................. 64
  Improving the Output Display .......................................................................................... 65
CHAPTER 10 - Technical Inspection Checklist ........................................................................... 66
  Introduction ......................................................................................................................... 66
  Items to be Inspected .......................................................................................................... 66
    UML Diagrams ................................................................................................................ 66
    Formal Specification ......................................................................................................... 66
  Formal Inspectors ............................................................................................................... 66
  Formal Inspection Checklist .............................................................................................. 66
CHAPTER 11 - Software Technical Instruction ........................................................................... 68
  Purpose ............................................................................................................................... 68
  Reusing the Project ............................................................................................................ 68
  Project Extension ............................................................................................................... 68
    Adding more patterns to the NPE Package ...................................................................... 68
    Improving NDPR Package ............................................................................................. 68
    Improving the Output Display ....................................................................................... 68
References ............................................................................................................................... 69
Appendix A - Formal Technical Inspection Letters ................................................................. 71
  Formal Technical Inspection Letter by Svitlana Volkova .................................................. 71
  Formal Technical Inspection Letter by Snehal Monteiro ................................................... 72
  Email from the Technical Inspector .................................................................................. 72
List of Figures

Figure 1.1 Project Overview ................................................................. 2
Figure 1.2 KSNE Data Flow Diagram .................................................. 5
Figure 1.3 System Use case ................................................................. 7
Figure 2.1 Project Schedule ................................................................. 14
Figure 4.1 Package View ................................................................. 27
Figure 4.2 NPE Class Diagram ......................................................... 28
Figure 4.3 NDPR Class Diagram ....................................................... 29
Figure 4.4 Sequence Diagram ......................................................... 31
Figure 5.1 Package View ................................................................. 38
Figure 5.2 NPE Class Diagram ......................................................... 39
Figure 5.3 NDPR Class Diagram ....................................................... 44
Figure 7.1 Input Screen of the Webpage ............................................. 55
Figure 7.2 Output Screen of the Webpage ......................................... 56
Figure 9.1 Phase Breakdown ......................................................... 60
Figure 9.2 Project Activity Breakdown .............................................. 61
Figure 9.3 Phase 1 Activity Breakdown ............................................ 61
Figure 9.4 Phase 2 Activity Breakdown ............................................ 62
Figure 9.5 Phase 3 Activity Breakdown ............................................ 62
Figure 9.6 Time Log ................................................................. 63
List of Tables

Table 2.1 COCOMO Effort Adjustment Factors ................................................................. 16
Table 2.2 Project Effort Adjustment Factor Values ......................................................... 17
Table 5.1 Detailed Description of the PatternMatch Class ............................................... 39
Table 5.2 Detailed Description of the MyFunctions Class ............................................... 40
Table 5.3 Detailed Description of the MyPatterns Class ................................................ 41
Table 5.4 Detailed Description of the Main Class ............................................................ 45
Table 5.5 Detailed Description of the Number Class ....................................................... 45
Table 5.6 Detailed Description of the Date Class ............................................................. 46
Table 5.7 Detailed Description of the Quantity Class ...................................................... 47
Table 6.1 Application Items Test Case ............................................................................. 52
Table 6.2 Patterns and Matching Text ............................................................................. 53
Table 8.1 Test Results Summary ..................................................................................... 57
Table 8.2 Test Application Items ..................................................................................... 58
Table 9.1 Project Duration ............................................................................................... 60
Table 10.1 Formal Inspection Checklist .......................................................................... 66
Table A.1 Technical Inspection Checklist from Svitlana ................................................. 71
Table A.2 Technical Inspection Checklist from Snehal .................................................. 72
Acknowledgements

It plunge me to exhilaration in taking the privilege in expressing my heartfelt gratitude to all those who helped, encouraged and foreseeing successful completion of my project. Ecstasy to work under gregarious guidance of Dr. William H. Hsu, to whom I would be extremely in debited for his valuable and timely suggestions.

I wish to convey my sincere thanks to Dr. Torben Amtoft and Dr. Mitchell L. Neilsen and all those who directly or indirectly contributed their assistance in finishing out his project successfully.

And finally, I would like to thank my parents and brother, without whose encouragement and support, would have ever reached this stage in life.
Dedication

I would like to dedicate this project to my parents for their blessings and to my brother who initiated and encouraged me to take this project. My special gratitude to Dr. William H. Hsu and Tim Weninger for the help and support to make this project successful.
CHAPTER 1 - Vision Document

Introduction

Motivation

The motivation of the project is to develop a system that can recognize the numerical and temporal patterns in text. Learning and understanding the numbers and time related information in natural text is important in tasks that attempt to recognize semantic entailment in pairs of sentences.

Apart from this, in most of the articles, as mentioned in [5], the occurrence of events is naturally anchored in time within the narrative text. For example, certain questions like – Is Bush currently the President of America? When was India attacked by Pakistan in last century? And many other questions like this are referred by the temporal aspects of the properties of the entities being questioned. Hence it is essential to be able to extract the timestamps of the events so as to understand the changes in the world over time and order the events with respect to one another.

Similarly, it is also important to know the quantity of entities along with the temporal information. For example, certain questions like – By how many votes did Obama win the elections? How many Oscar awards are won by Steven Spielberg? What was the highest temperature recorded in the year 2008? To answer such questions and maintain statistics over the entities, it is required to save information about the numbers that appear in the news.

Named Entity Recognition (NER) helps in identifying names of persons, locations, organizations and products in text but to understand the timestamps, ordering and persistence of events it is necessary to have a Natural Language Processing (NLP) system that can extract the numerical and temporal information from natural text.

NLP

Natural language processing (NLP) is a field of computer science concerned with the interactions between computers and human (natural) languages [8]. The term natural language is used to distinguish human languages (such as Spanish, Swahili or Swedish) from formal or
computer languages (such as C++, Java or JSP). NLP may encompass both text and speech, but the current project involves text only. Some research areas in NLP are sentence understanding, probabilistic parsing and tagging, biomedical information extraction, grammar induction, word sense disambiguation and automatic question answering.

The present project belongs to the area of probabilistic parsing and tagging where a given sentence is POS tagged and then shallow parsed to extract the numerical information from the given sentence.

**KDD- Service based Numerical Entity Searcher (KSNES)**

The KSNES is such a system that should be able to identify the numerical and temporal information from natural text. The user should be able to interact with the system by entering the text in the textbox on the webpage. KSNES absorbs this text and outputs the number-unit/date on the webpage for the user to read.

**Figure 1.1 Project Overview**

![Project Overview Diagram]

**Terms and Conditions**

**Actor:** In UML, an actor is the end user of the system.

**Entity:** Information that have real existence and are distinct and independent that is presented in the examples of section 1.1.

**Data Flow Diagram:** It is used for the visualization of data processing, how data flows in a system.

**Information Extraction (IE):** It is a type of information retrieval whose goal is to automatically extract structured information, i.e. categorized and contextually and semantically well-defined data from a certain domain, from unstructured documents.
**Information Retrieval (IR):** It is the science of searching for documents, for information within documents and for metadata about documents, as well as that of searching relational databases and the World Wide Web.

**KDD(Knowledge Discovery in Databases):** A group headed by Dr. William H. Hsu whose primary focus is NLP.

**NABC:** National Agricultural Bio-Security Center.

**Named Entity Recognition (NER):** It is a subtask of information extraction that seeks to locate and classify atomic elements in text into predefined categories such as the names of persons, organizations, locations and miscellaneous entities.

**Natural Language Processing (NLP):** It is a field of computer science concerned with the interactions between computers and human (natural) languages.

**Numerical Patterns:** The phrases that represent numbers. Ex: 33 dollars, 1000 people, 145 miles, 1034.54, 843 tons etc.

**POS:** Parts-of-Speech.

**Penn Treebank Tagset:** This acts as a repository for all the tags and their meanings that are used by the POS Tagger.

**Shallow Parsing/Chunking:** It is an analysis of a sentence which identifies the constituents (noun groups, verbs, verb groups, etc.) but does not specify their internal structure, or their role in the main sentence. In this project the chunking is done to identify the numerical phrases.

**Sub-Chunker:** A module of the project that takes the numerical/temporal phrases as input and outputs the number-unit/date.

**Temporal Patterns:** The phrases that denote date/time. Ex: Dec 10th 1986, 2009, 5 May, this year, today, Monday, January etc.

**Textual Entailment:** It is the relation between two sentences where the truth of one requires the truth of the other.

**Unified Modeling Language (UML):** A standard notation used to describe real-world objects.

**Use case Diagram:** A behavioral diagram defined by UML. It provides a graphical depiction of system functionality in terms of actors.
Project Overview

This section presents the information about the overall view of the working of the system and introduces the modules that made the KSNES work.

Introduction

The block diagram in Figure 2 gives a bird’s view of the internal working of the system along with an example that shows the data at the end of each module in the system. User should be entering the text that is required to be chunked through a webpage.

Stanford POS Tagger

The text entered by the user is given to the Stanford Log-linear POS tagger that can tokenize and tag the sentence with POS such as noun, verb, adjective etc., which is developed by Stanford NLP Group. The words in the sentence are tagged by using the Penn Treebank tag set which has notations for the tags that should be assigned to words in different contexts. For example, if the input text is: I lost 33 dollars in 1999. Then the sentence would be tagged by the POS Tagger as – I/PRP lost/VBD 33/CD dollars/NNS in/IN 1999/CD. Here PRP (Personal pronoun), VBD (Verb, past tense), CD (Cardinal number), NNS (Plural noun) are some of the tags from the Penn Treebank tag set which has lot more such tags like those that describe conjunctions (and, but, or), prepositions (of, in), cardinal numbers (1, 16, 26, three), personal pronouns (I, you), possessive pronoun (your, one’s) etc.,

Numerical Phrase Extractor

The tagged output from the POS Tagger will be fed to a Numerical Phrase Extractor which filters numerical phrases from the tagged sentence. This part of the system should be able to identify the dates, the numbers, units associated with them and type to which the unit belongs. From the example discussed in the previous section, ‘33 dollars’ and ‘1999’ should be extracted using the shallow parsing technique.

Number-Unit/Date Pattern Recognizer

The extracted numerical phrase will be fed to the Number-Unit/Date Pattern Recognizer that should be able to present the number-unit/date information of the given textual input. The output for the above mentioned example would then look like this - ‘33 dollars’: type: number, value: 33, unit – dollars, unit-type – money; ‘1999’: type: date, value: 0/0/1999. Here ‘type’
specifies if it’s a number or a temporal phrase, value is the number or the date, unit is the physical unit like miles, dollars, cubic-meters etc., unit-type is the category of the unit like distance, currency, volume etc.,

Thus the required output is generated and displayed to the user on the webpage.

Figure 1.2 KSNES Data Flow Diagram
**Project Goal**

The primary focus of KSNES is to construct a system than should be able to extract the numerical data from natural language text and hence can also be called as Numerical Quantifier. The project is service based where the whole system is put on the web server and is access using a GUI which is a webpage. There are no intermediate steps for the user to perform after sending the input from the main webpage by clicking the send button; user should be able to collect the output on the same page.

**Project Purpose**

The present application is developed for the purpose of providing a good base for the KDD students to perform numerical sub-chunking. The KDD group members can use this project to collect some statistical data like finding the outbreak of the disease, number of cattle affected etc., from the web crawled data in the NABC related project. This project since made a service can be interfaced by other programs to perform numerical sub-chunking.

**Project Requirements**

In this section, we will discuss the critical requirements and other requirements that should be satisfied to accomplish a working KSNES system. Each requirement is given a unique requirement number and discussed in detail. The critical requirements in the project will be noted near the description of the requirement and the planned release that will fulfill that requirement is also mentioned.

Figure 3 is the use case diagram that visualizes the actions that a user performs to interact with the system. Here we can see that the user opens the webpage, enters the text and clicks on the send button. The request invokes the POS tagger that is set on the server which in turn corresponds with the numerical phrase extractor and Number-Unit/Date Pattern Recognizer which send the output to be displayed back on the webpage. These are the major functions of the user.

The requirements explained in the below sections are those that are to be fulfilled by the final system. They are divided based on the different modules that the system is composed of. They are – Application Requirements, POS Tagger requirements, Numerical Phrase Extractor, Number-Unit/Date Pattern Recognizer. Modularizing the project helps to do updates and
modifications to different modules of the project without affecting other modules. It will also make the modules of the system reusable and easy to understand.

**Figure 1.3 System Use case**

![System Use case diagram]

**Application Requirements**

The overall idea of the system is explained in the requirements specified in this section. The modules that are unseen to the user are explained in the coming sections. The requirements are numbered as ARI X, where ARI stands for Application Requirement Item.

**ARI 1 [Critical Requirement]**

The program shall provide a GUI for the user interaction. This is a critical requirement because the usefulness of the system would be extremely limited and complicated if done in a command line format.

**Build Release Applicability:** Final Release

**ARI 2**

The application shall be executable in a single step and no user interaction is required.

**Build Release Applicability:** Final Release

**ARI 3 [Critical Requirement]**

The application shall be started when the user enters the text into the text box in the webpage. This is a critical requirement because the input given by the user is processed and modified by application to give the final output.

**Build Release Applicability:** Final Release
ARI 4
The application shall invoke the other modules of the project when the user clicks on the send button.

**Build Release Applicability:** Final Release

ARI 5
The system shall be setup as a service to be able to access by the webpage. This is a critical requirement because the application is being developed so that other users can reuse this module.

**Build Release Applicability:** Final Release

ARI 6 [Critical Requirement]
The user shall be able to view the chunked output on the webpage. This is a critical requirement because the application’s goal is to show the output to the user.

**Build Release Applicability:** Final Release

ARI 7 [Critical Requirement]
The user shall be able to stop the running of the application after viewing his output by closing the web browser.

**Build Release Applicability:** Final Release

ARI 8
The user shall be able to run the application again with a new input ones the previously entered text is chunked.

**Build Release Applicability:** Final Release

ARI 9
The user shall be able to enter the input of any size. This is critical because the application shall not restrict the user’s freedom on the size of the input the user can to enter.

**Build Release Applicability:** Demo2, Final Release

**POS Tagger Requirements**
This section deals with the working of the tagger. This module of the project does not interact with the user but is still being mentioned so as to understand the underlying working of
this module. The requirements are numbered as PTRI X, where PTRI stands for POS Tagger Requirement Item.

**PTRI 1 [Critical Requirement]**

The POS Tagger shall be given the raw input text that user enters on the webpage. This is a critical requirement because the input is required to be tagged with POS to be processed further.

**Build Release Applicability:** Demo2, Final Release

**PTRI 2 [Critical Requirement]**

The POS Tagger shall be a service request from the main webpage. This is critical because POS tagger is a third party segment and can be run efficiently when used as a service.

**Build Release Applicability:** Demo2, Final Release

**PTRI 3**

The POS Tagger shall be able to tokenize the given text.

**Build Release Applicability:** Demo2, Final Release

**PTRI 4**

The POS Tagger shall be able to tag the words, punctuations and symbols in the sentence using the Penn Treebank Tag set.

**Build Release Applicability:** Demo1, Demo2, Final Release

**PTRI 5 [Critical Requirement]**

The POS Tagger shall be able to produce the tagged sentence. This is a critical requirement because the module should be producing the expected output.

**Build Release Applicability:** Demo1, Demo2, Final Release

**PTRI 6 [Critical Requirement]**

The POS Tagger shall be able to send the tagged sentence to the next module which is the Numerical Phrase Extractor. This is a critical requirement because the module should be passing the information for the KSNES to work further.

**Build Release Applicability:** Final Release
**Numerical Phrase Extractor Requirements**

This section presents the requirements of the phrase extractor which takes in the tagged text and filters out the numerical phrase. This module of the project does not interact with the user but is still being mentioned so as to understand the underlying working of the module. The requirements are numbered as NPERI X, where NPERI stands for Numerical Phrase Extractor Requirement Item.

*NPERI 1 [Critical Requirement]*

The Numerical Phrase Extractor shall be able to take the tagged sentence from the POS tagger. This is a critical requirement because this module requires the output of the previous module to proceed further in chunking.

**Build Release Applicability:** Final Release

*NPERI 2*

The Numerical Phrase Extractor shall be able to identify the tagged words that may be containing the numbers and the units.

**Build Release Applicability:** Demo2, Final Release

*NPERI 3*

The Numerical Phrase Extractor shall be able to identify the tagged words that may be containing the dates.

**Build Release Applicability:** Demo2, Final Release

*NPERI 4 [Critical Requirement]*

The Numerical Phrase Extractor shall be able to produce the filtered number-unit or the date phrase. This is a critical requirement because the module should be producing the expected output.

**Build Release Applicability:** Demo2, Final Release

*NPERI 5 [Critical Requirement]*

The Numerical Phrase Extractor shall be able to send the filtered phrase to the next module which is the Number-Unit/Date Pattern Recognizer. This is a critical requirement because the module should be passing the information for the KSNES to work further.
Build Release Applicability: Final Release

*Number-Unit/Date Pattern Recognizer Requirements*

This module takes the extracted numerical phrase and determines the number, unit, date, unit-type in it and presents it to the user on the webpage. This module of the project does not interact with the user but is still being mentioned so as to understand the underlying working of the module. The requirements are numbered as NDPRRI X, where NDPRRI stands for Number-Unit/Date Pattern Recognizer Requirement Item.

**NDPRRI 1 [Critical Requirement]**

The Number-Unit/Date Pattern recognizer shall be able to take the extracted phrase from the Numerical Phrase Extractor. This is a critical requirement because this module requires the output of the previous module to proceed further in chunking.

**Build Release Applicability:** Demo1, Demo2, Final Release

**NDPRRI 2**

The Number-Unit/Date Pattern recognizer shall be able to identify the numbers, units and unit-type if present in the phrase.

**Build Release Applicability:** Demo1, Demo2, Final Release

**NDPRRI 3**

The Number-Unit/Date Pattern recognizer shall be able to identify the date, month and the year if present in the given phrase.

**Build Release Applicability:** Demo1, Demo2, Final Release

**NDPRRI 4 [Critical Requirement]**

The Number-Unit/Date Pattern Recognizer shall be able to produce the number, unit corresponding to it and the type to which the unit belongs to if it’s a number phrase. And if it is a temporal phrase then the module should be able to display the date based on the number, month and year information in the phrase. This is a critical requirement because the module should be producing the expected output.

**Build Release Applicability:** Demo1, Demo2, Final Release

**NDPRRI 5 [Critical Requirement]**
The Number-Unit/Date Pattern Recognizer shall be able to display the value, unit, and unit-type to the user on the webpage. This is a critical requirement because the module should be able to produce the output since it’s the final output expected output of the KSNES system.

**Build Release Applicability:** Final Release

**Assumptions**

- The user will have to interact with the system using webpage.
- The system requires Java 1.5 or later to be installed on the computer running the application.
- The user will need a minimum of 512 MB of memory.
- The user will need to have a computer with a minimum speed of 1.6 GHz.

**Constraints**

- POS Tagger is not implemented as a part of this project. A Stanford Log-linear POS Tagger, developed by Stanford University NLP Group, is used that takes the raw text and provides a POS tagged sentence.
- Numerical Quantifier developed by Ben Sapp, UIUC is used as a base to develop the Number-Unit/Date Pattern Recognizer that outputs the number-unit/date information of the given numerical phrase.
- The whole system should be set up as a service on a server so that anyone can interface to it or use it.

**Environment**

- Eclipse 3.3.0 and Java version 1.4 will be used to develop the Numerical Phrase Extractor.
- Eclipse 3.3.0 and Java version 1.4 will be used to run the POS Tagger
- Version control will be handled using Tortoisesvn.
- Number-Unit/Date Pattern Recognizer will be developed using GNU C++.
CHAPTER 2 - Project Plan

Test Breakdown

Project Phases

The Unified Process divides the project into three phases: Inception, Elaboration and Production.

Inception Phase

The inception phase is the first and smallest phase in the project. In this phase the project scope, use cases diagrams, functional requirements, project overview, project design, risks in the project and preliminary project schedule are analyzed before beginning the project development and documented in three different documents – Vision Document, Project Plan and Software Quality Assurance Plan.

Vision Document outlines the scope, purpose, goals and functional requirements of the project. In the Project Plan document, effort estimation and project schedule details are logged. The list of the required documentation, standards and conventions, steps to maintain quality of the project are recorded in the Software Quality Assurance Plan.

This phase is completed when the developer gets all the documentation to be reviewed and approved by the supervisory committee. To complete the phase, an initial prototype developed by the developer is given as the first presentation.

Elaboration Phase

In the Elaboration Phase the architectural design plan of the system is developed. The main objective of this phase is to capture the healthy majority of the system requirements using OCL diagrams and develop a test plan to specify the testing techniques as well as the method of documenting, tracking and debugging. Along with this, the documents developed in the inception phase are updated as per the supervisory committee suggestions from the first presentation. Two formal technical inspectors, who are the fellow MSE students, will perform technical inspections of the architectural design and provide feedback from their findings.
Including the suggestions given by the supervisory committee in the previous presentation, a second prototype will be developed in this stage by modifying the first prototype that will support the key system functionalities and exhibit the right behavior in terms of performance, scalability and cost.

The second version of the documentation is submitted to the supervisory committee for changes and approval. The second presentation is given at the end of this phase with the demonstration of a stable and executable architecture baseline.

**Production Phase**

The Production phase is the largest phase in the project where in the complete system is build and coded to develop an executable release of the software. The developer also documents the code and deliverables like User Manual, Project Evaluation, and Test Logs etc.

The developer submits the executable version of the project and the supporting documents for the supervisory committee for reviewing and approval. The final presentation will be given at the end of this phase with a demonstration of the complete project.

**Project Schedule**

The Gantt chart below presents the schedule for the KDD-Numerical Sub-Chunker project. A separate document is submitted for better view of the chart.

**Figure 2.1 Project Schedule**
Cost Estimate

To estimate the effort, cost and schedule for a project many different measures can be taken but the most efficient method is the COCOMO II model as it considers the factors like complexity, reliability, database and memory usage, experience of the developer into consideration into account during calculations. As these are the major factors that would affect the development of a software project this model is considered to be the right choice to estimate the development cost.

**COCOMO II**

Original COCOMO model has been very successful, but it does not apply to newer software development practices as well as it does to traditional practices. COCOMO II is the latest major extension to the original COCOMO model published in 1981.

COCOMO II is being developed by Dr. Barry Boehm and his students at USC and it targets the software projects of the 1990s and 2000s, and will continue to evolve over the next few years. It consists of three sub-models each one offering increased fidelity the further along one is in the project planning and design process. Listed in increasing fidelity, these sub-models can be explained as follows:

- The Application Composition Model – Suitable for projects built with modern GUI-builder tools based on new Object Points.
- The Early Design Model – This model can be used to get a rough estimate of a project’s cost and duration before determining its entire architecture. It uses a small set of new cost drivers and new estimating equations based on unadjusted function points or KSLOC.
- The Post-Architecture Model – This is the most detailed COCOMO II model. It is used before we develop the architecture.

The present KSNES project can be considered as The Post-Architecture Model as we can see its architecture from Vision document. The following formulae are used for the cost estimation:

\[
\text{Effort} = 2.45 \times \text{EAF} \times (\text{KSLOC})^{1.09}
\]
Time = 2.5 * (Effort)$^{0.38}$

Where in:
- Effort = the number of person months (PM)
- Time = Duration time in months for project
- KSLOC = Estimated number of source lines of code for the project (expressed in thousands)
- EAF = Effort Adjustment Factor

Effort Adjustment Factors are 15 in number and their values differ within a given range. Each adjustment factor may fall in one of the categories – very low, low, nominal, high, very high, extra high and based on these categories a value is given to EAF. EAF is the product of the values of the 15 factors shown in the below table:

**Table 2.1 COCOMO Effort Adjustment Factors**

<table>
<thead>
<tr>
<th>Identifier</th>
<th>EAF</th>
<th>Possible Range Of Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELY</td>
<td>Required Software Reliability</td>
<td>0.75 – 1.40</td>
</tr>
<tr>
<td>DATA</td>
<td>Size of Application Database</td>
<td>0.94 – 1.16</td>
</tr>
<tr>
<td>CPLX</td>
<td>Complexity of the Product</td>
<td>0.70 – 1.65</td>
</tr>
<tr>
<td>TIME</td>
<td>Run-time Performance Requirements</td>
<td>1.00 – 1.66</td>
</tr>
<tr>
<td>STOR</td>
<td>Memory Constraints</td>
<td>1.00 – 1.56</td>
</tr>
<tr>
<td>VIRT</td>
<td>Virtual Machine Volatility</td>
<td>0.87 – 1.30</td>
</tr>
<tr>
<td>TURN</td>
<td>Required Turnabout Time</td>
<td>0.87 – 1.15</td>
</tr>
<tr>
<td>ACAP</td>
<td>Analyst Capability</td>
<td>1.46 – 0.71</td>
</tr>
<tr>
<td>AEXP</td>
<td>Applications Experience</td>
<td>1.29 – 0.82</td>
</tr>
<tr>
<td>PCAP</td>
<td>Software Engineer Capability</td>
<td>1.42 – 0.70</td>
</tr>
<tr>
<td>VEXP</td>
<td>Virtual Machine Experience</td>
<td>1.21 – 0.90</td>
</tr>
<tr>
<td>LEXP</td>
<td>Programming Language Experience</td>
<td>1.14 – 0.95</td>
</tr>
<tr>
<td>TOOL</td>
<td>Use of Software Tools</td>
<td>1.24 – 0.82</td>
</tr>
<tr>
<td>MODP</td>
<td>Use of Modern Software Practices</td>
<td>1.24 – 0.83</td>
</tr>
<tr>
<td>SCED</td>
<td>Required Development Schedule</td>
<td>1.23 – 1.10</td>
</tr>
</tbody>
</table>
Based on the KDD-Numerical Sub-Chunker project these above defined factors can be assigned a value in the range specified above. In the table given below each identifier is classified and given a value. The reason for choosing a specific value for each factor is also mentioned in the table.

**Table 2.2 Project Effort Adjustment Factor Values**

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Classification</th>
<th>Value</th>
<th>Reasoning</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELY</td>
<td>Low</td>
<td>0.88</td>
<td>Project is not safety critical, and does not have to be completely reliable</td>
</tr>
<tr>
<td>DATA</td>
<td>Low</td>
<td>0.94</td>
<td>Data used by the project is only the user’s input which is relatively small in size</td>
</tr>
<tr>
<td>CPLX</td>
<td>High</td>
<td>1.02</td>
<td>Processing natural text and extracting the required features from them are a complicated</td>
</tr>
<tr>
<td>TIME</td>
<td>Nominal</td>
<td>1.00</td>
<td>Response time is important but not a overly critical issue</td>
</tr>
<tr>
<td>STOR</td>
<td>Nominal</td>
<td>1.00</td>
<td>Tagging and Numerical Phrase Extraction will not require high memory usage</td>
</tr>
<tr>
<td>VIRT</td>
<td>Low</td>
<td>0.87</td>
<td>Low complexity of the hardware and software</td>
</tr>
<tr>
<td>TURN</td>
<td>Low</td>
<td>0.87</td>
<td>The turnaround time on results is low as the project is developed by a single developer</td>
</tr>
<tr>
<td>ACAP</td>
<td>High</td>
<td>0.86</td>
<td>Developer has 3+ years experience in Software Engineering</td>
</tr>
<tr>
<td>AEXP</td>
<td>Nominal</td>
<td>1.00</td>
<td>Developer has 2+ years experience in applications development</td>
</tr>
<tr>
<td>PCAP</td>
<td>Nominal</td>
<td>1.00</td>
<td>Developer has 6 month of experience in the application area</td>
</tr>
<tr>
<td>VEXP</td>
<td>Low</td>
<td>1.10</td>
<td>Developer has 6 months of experience and is learning Java Virtual Machine</td>
</tr>
<tr>
<td>LEXP</td>
<td>Low</td>
<td>1.15</td>
<td>Developer has 6 months of experience and is learning Java</td>
</tr>
<tr>
<td>TOOL</td>
<td>Nominal</td>
<td>1.00</td>
<td>Moderate experience with the tools being used</td>
</tr>
</tbody>
</table>
EAF is the product of the values of the adjustment factors. From the above table we have the calculated value of EAF as 0.58. To make the system work, coding is required for making the GUI, patching the modules in the system, making the client-server system work. Based on this the estimated size of the project will be around 1 KLOC.

Using these figures, Effort and Time can be calculated as:

\[
\text{Effort} = 2.45 \times 0.58 \times 1^{1.09} = 1.42
\]
\[
\text{Time} = 2.5 \times 1.42^{0.38} = 2.85
\]

From the above calculations we can see that the COCOMO model estimates the effort to be 1.42 person months to be required to complete the project. From the Time value calculation the project should be taking around 2.85 months to complete it and is almost near to the project schedule shown in the Gantt chart.

There is a limitation with the COCOMO II model since there could be a slight variation in the above calculated values based in the misjudged EAF values, increased project complexity and scope of the project.

**Architecture Elaboration Plan**

Before presenting the second presentation, elaboration phase comes to end by completing all the required documents and artifacts. The details about the documents to be submitted are as follows:

**Vision Document Revision**

A revised version of the Vision Document is required created to include the changes suggested by the committee after the first presentation. This version also includes the complete list of requirements which are ranked based on the priority. This is submitted to the major professor for approval.
**Project Plan Revision**

Project Plan document is revised as per the feedback from the committee members in the first presentation. Based on the changes in the schedule and the cost estimation, the Gantt chart and the COCOMO estimates are updated. This is submitted to the major professor for approval.

**Architectural Design**

The Architectural Design document explains the components and the scenarios of the present system using UML diagrams. This is submitted to the major professor for approval.

**Prototype Development**

Based on the suggestions from the supervisory committee during the first project presentation and by adding new functionalities a prototype is developed. This prototype would be an extended version of the prototype developed in the inception phase. This is submitted to the major professor for approval.

**Test Plan**

To ensure that all the requirements specified in the Vision document are satisfied, a test plan is developed that outlines all the testing activities required. This document also contains the instructions on how to evaluate the product and will be submitted to the major professor for approval.

**Formal Technical Inspections**

Two yet to be determined fellow MSE students will act as the formal inspectors of the project. These inspectors will use a formal inspection checklist that will be produced during the Elaboration phase based on their findings.

**Formal Requirements Specification**

Using the USE (UML – based Specification Environment) tool, at least one of the modules of the project will be specified using OCL. This is submitted to the major professor for approval.
Software Production Plan

Test Plan Revision
As per the suggestions from the committee members after presentation 2, the changes will be made to the test plan and will be submitted to the major professor.

Architectural Design Revision
The revision of the Architectural Design will be done after the second presentation as per the suggestions from the committee members. The major professor will approve the changes to the document.

Component Design
UML will be used to convey detailed information about the software components in the component design document. It will include all the attributes and methods for the classes in the project and this document is submitted to the major professor for approval.

Final Software Executable
The prototype developed during the Architecture Elaboration Phase will be developed and improved with additional features during the Production Phase. The additional features would be the required functionalities and the suggestions from the committee members.

Formal Technical Inspections
Two MSE students will provide feedback to the project after inspecting the formal specifications. Both the inspector will report their formal inspection checklist based on their findings.

User’s Manual
To help the user to use the system, a user manual will be provided by the software developer. This manual will guide to use the different sections of the system and lists the various troubleshooting problems and solutions.
**Test Assessment**

At the end of the software development, the developer will run the test cases logged in the Test Plan document and will record the results. The Test Assessment document contains the results of running these tests.

**Technical Instructions for Reuse and Extension**

The developer also provides a written copy of the guidelines that help other developers to use the current system for other MSE projects. The document also contains the features that could be added to the project to adapt it for different uses.

**Project Assessment**

Project assessment is the document that the developer writes at the completion of the software development and the testing about his/her opinion on the project. The document will describe the issues that went well, the scope of improvement and the things that did not work. This will contain the final metrics for the project.
CHAPTER 3 - Software Quality Assurance Plan

Purpose

The purpose of this document is to define the process to ensure that the software to be developed is of high quality. This document also lists the tools and the techniques used to develop the application and conclude with the list of all the deliverables at each phase.

Management

Organization

Supervisory Committee
- Dr. William H. Hsu
- Dr. Torben Amtoft
- Dr. Mitchell L. Neilsen

Major Professor
- Dr. William H. Hsu

Developer
- Naga Sowjanya Karumuri

Formal Technical Inspectors
- Snehal Monteiro
- Svitlana Volkova

Tasks

All the project tasks performed are recorded in detail in the Project Plan document. The Project Plan document will be revised as per the suggestions from the supervisory committee after the first presentation and changes are also made to the Gantt chart as per the changes in the schedule.
Responsibilities

Supervisory Committee

The primary responsibility of the committee members is to attend all the three project presentations and provide feedback to the developer based on the project progress.

Major Professor

Major Professor holds two responsible and the first is to act as a supervisory committee member. The second responsibility is to supervise and evaluate the project with the developer on a weekly basis.

Developer

The main responsibility of the developer is to produce the product with in time with better performance along with documentation. Developer has to maintain the time log, meet the major professor every week and discuss the project for feedback.

Formal Technical Inspectors

The duty of the formal technical inspectors is to inspect the architecture, design and source code of the project and submit their findings as a formal report.

Documentation

The official documentation requirements for MSE projects are defined at the MSE webpage: http://mse.cis.ksu.edu/online/mse-portfolio.htm. The documents that are to be submitted are listed in Section 12 of this document.

All the documents related to the project can be found on the below web location: http://people.cis.ksu.edu/~sowji/100jiMSE/.

Standards, Practices, Conventions and Metrics

Documentation

IEEE standards will be followed for all applicable documentation throughout the project.
**Coding Standards**

For the code written in Java, Java naming conventions will be followed. Source code API will be generated using Javadoc. For the code written in C++, follows the C++ coding standards and style guide.

**Metrics**

COCOMO II will be used to estimate the project cost in terms of time and effort.

**Reviews and Audits**

At the end of each phase, the documentation, source code and executable products will be evaluated by the supervisory committee. Two fellow MSE students would act as the formal inspectors and submit their findings as a report after analyzing the architecture, design and source code.

**Testing**

Completed details about the test procedures and the expected results are all listed in the Test Plan. Since the present project consists of different modules, modular testing can be done at the initial stages. Once the whole system is built, the overall performance can be tested. The results are manually checked with the expected results. It is the developer’s responsibility to fix the bugs using different methods and document them in the test plan.

**Problem Reporting and Corrective Actions**

All the problems detected during the development of the system will be recorded in the Software Problem Report spreadsheet. Each problem detected will be recorded by defining the parameters – problem description, time consumed to fix the bug, the correction actions taken. If there are any such problems that are not solved they will be brought to the notice of the major professor and discussed.

**Tools, Technologies and Methodologies**

The following tools are used for coding, testing and documenting the reports:

- Eclipse IDE – for software development
- Java – for software development
GNU C++ – for software development
JSP – for server side coding
Apache Tomcat – for web server
Microsoft Word – for documentation development
Microsoft Excel – for risk and problem report tracking and time logs
Microsoft PowerPoint – for project presentation creation
Microsoft Project – for drawing the Gantt chart (project planning)
Microsoft Visio – for software design development
Microsoft Front Page – for developing the project webpage
Open Office – for document conversation to PDF
JUnit – for testing the java code

**Code and Media Control**

All the source code being developed will be controlled using the Tortoisesvc –Subversion Control system. All the source code is maintained on the developer’s personal computer.

All documents will be maintained on the developer’s personal computer with filenames associated with the version numbers and the dates on which they are created. Each document includes a change log and all the documents are put on the project’s webpage:

http://people.cis.ksu.edu/~sowji/100jiMSE/

**Risk Management**

Software risks will be logged in a Software Risk Reporting and Mitigation spreadsheet as well discussed with the major professor during the weekly meeting hours.

**Deliverables**

The list below presents the deliverables at each phase of the project:

**Phase I**

- Vision Document
- Project Plan
- Prototype Demonstration
- Software Quality Assurance Plan
• Time Log
• Presentation

**Phase II**

• Vision Document
• Project Plan
• Software Requirements Specification
• Architecture Design
• Test Plan
• Software Risk Reporting and Mitigation Document
• Technical Inspection Checklist
• Executable Architecture Prototype
• Action Items
• Time Log
• Presentation

**Phase III**

• Component Design
• Source Code
• Executable Project
• User Manual
• Formal Technical Inspection Letters
• Project Evaluation
• Software Problem Reports
• Time Log
• Presentation
CHAPTER 4 - Architecture Design

Introduction

The purpose of this document is to provide the architectural design of KSNES. The documentation presents the class and sequence diagrams and explanation is provided for each diagram. The last section has the formal specification of the Numerical Phrase Extractor and the Number-Unit/Date Pattern Recognizer.

Background

The purpose of the project is to extract the numerical information from the given raw text and present the values, units and the numerical phrase type. The whole system is set as a service on the server and a webpage is created for the user to enter the raw text and view the required information.

KDD- Service based Numerical Entity Searcher

Package View

As discussed in the Vision document, the present system has client server architecture. The webpage acts as the client and the KSNES system acts as the server. In this document, details of Numerical Phrase Extractor and Number-Unit/Date Pattern Recognizer are presented since the developer has worked on these modules and the POS Tagger is taken from a third party and hence not documented.

In the below package view, we see the three main modules of the project that are called by a controller class so that they work together in sequence to process the raw text and extract the numerical phrases from it.

We see that Number-Unit/Date Phrase Recognizer depends on Numerical Phrase Extractor and this in turn depends on POS Tagger.

Figure 4.1 Package View
Class Description

- **PatternMatch**: This class starts the NPE package working by calling a method from MyFunctions to start reading the given tagged input file.

- **MyFunctions**: This class holds the functions to read from the input file, writes the map values to the output file, delete the files and parse the line with the patterns available in the MyPatterns class.

- **MyPatterns**: This class contains all the patterns and each pattern takes the input line and when found any match, is written to the map. This map is printed back to the file by calling a function from MyFunctions class.
Class Description

- **Main**: This is the class that reads the input from the file and calls the number or the date functions based on the type of the input.
- **Number**: This class parses the given number phrase and evaluates the value, unit, unit-type and the bound corresponding to that phrase.
- **Date**: This class parses the given date phrase and prints the date as mm/dd/yyyy. If there is any second date it is also parsed as mm/dd/yyyy.
• **Quantity:** This class has the basic functions defined that are used by the number and the date classes.

**Sequence Diagrams**

The system is more like a black box to the user and hence the user has very less to interact with the system but the below diagram explains how the user system interaction is taking place. The message passing between the user and the system are pseudo as it has been set by one of the KDD members. Hence the below sequence diagram explains the process happening between the user and the system but do not present the actual methods that have been invoked during the flow.

**User Interacting with the System**

**Prerequisites:**

- KSNES is set as a service on the server.
- A webpage is designed for the user to access the system through it.

**Sequence of Events:**

1. User opens the browser
2. User opens the KSNES webpage
3. User enters the raw text into the text box in the page
4. User clicks the send button
5. User sees a new webpage displaying the given text, chunks, chunk types, values, units, unit-types and bounds.
6. User may close the browser or enter a new raw text again.

**Post-Conditions:** User sees a new webpage displaying the given text, chunks, chunk types, values, units, unit-types and bounds.
Figure 4.4 Sequence Diagram

Formal Specification

model KSNES
   --
   -- NPE PACKAGE
   --

class Readfile
   attributes
      rline: String
   end

class Writefile
   attributes
      wline: String
class Map
attributes
    linenum: Integer
    value: String
end

class Datetemp
end

class PatternMatch
attributes
    mfunction: MyFunctions
end

class PatternMatch
attributes
    mfunction: MyFunctions
end

class MyFunctions
attributes
    tmp: String
    iline: String
    fchunk: String
    passon: String
    rf: Readfile
    wf: Writefile
    mchunkorder: Map
operations
    startparsing()
    postchunks()
    deletefile()
    cleanchunk(ichunk: String)
    parsesline(taggedline: String)
end

class MyPatterns
attributes
    chunk: String
mf: MyFunctions
chunkorder: Map
operations
  p_abtfrac(tline: String): String
  p_age(tline: String): String
  p_ampm(tline: String): String
  p_and(tline: String): String
  p_anydate(tline: String): String
  p_btwfrm(tline: String): String
  p_btwfrmd(tline: String): String
  p_date(tline: String): String
  p_days(tline: String): String
  p_centuary(tline: String): String
  p_hyphenww(tline: String): String
  p_hyphennumnum(tline: String): String
  p_in(tline: String): String
  p_mids(tline: String): String
  p_months(tline: String): String
  p_numunit(tline: String): String
  p_per(tline: String): String
  p_percentinches(tline: String): String
  p_ratio(tline: String): String
  p_tty(tline: String): String
  p_twmy(tline: String): String
  p_xbits(tline: String): String
  p_yrange(tline: String): String
end

--
-- NDPR PACKAGE
--

class main
attributes
  infile: Readfile
operations
  batch_process(infile: String)
  main(argc: Integer, argv: Set(String))
end

class number
attributes
  unit: String
  cat: String
  value: Integer
  inputnum: String
operations
  get_units()
  is_equal(number: Integer): Boolean
  print()
end

class date
attributes
  month: Integer
  day: Integer
  year: Integer
  format: String
  modifier: String
  second_date: Datetemp
  inputdate: String
operations
  get_modifier()
  make_relative_date(dt:Integer)
  is_equal()
  print()
end

class quantity
attributes
  dbl2string: String
  int2string: String
  type: String
  phrase: String
  bound: String
operations
  match(s: String, pattern: String, v: Set(String))
  subst(s: String, pattern: String, replace: String)
  split(s: String, delim: String): Set(String)
  convert_to_integer(phrase: String): String
  get_bound()
  is_equal()
-- ASSOCIATIONS

-- This association will be not appear in the class diagram but
-- is seen as the relation between the packages

association Packagerelations between
  PatternMatch[1] role usedby
  main[1] role connectsto
end

--

-- NPE PACKAGE

association PatternUseFunctions between
  PatternMatch[1] role calledbypatternmatch
  MyFunctions[1] role maincallsfuncs
end

association FunctionsCallPatterns between
  MyFunctions[1] role patternSCALLfuncitons
  MyPatterns[1] role functionscallpatterns
end

--

-- NDPR PACKAGE

association InputMaybeNumber between
  main[1] role inputmaybenumber
  number[1] role maincallsnum
end

association InputMaybeDate between
  main[1] role inputmaybedate
date[1] role maincallsdate
end

association NumberUseQuantity between
    number[1] role quantitybynum
    quantity[1] role numberusequantity
end

association DateUseQuantity between
    date[1] role quantitybydate
    quantity[1] role dateusequantity
end

association NumberUseUnitslist between
    number[1] role maybecontainedby
    unitslist[1] role containsunit
end

constraints

--
-- The output of the NPE Package is the input for the NDPR Package
--

context PatternMatch
inv passonfile:
    maincallsfuncs.rf = connectsto.infile

--
-- The input to the NDPR Package is either a number or a date
--

context main
inv dateornumber:
    ((infile.rline = maincallsnum.inputnum) or
        (infile.rline = maincallsdate.inputdate)) and
    (maincallsnum.inputnum <>
     maincallsdate.inputdate)
--
-- The outputs written to the output file are those put in the
-- map after processing the input file
--

context MyFunctions
inv printcorrectoutput:
    mchunkorder=functions.callpatterns.chunkorder
    and wf.wline = mchunkorder.value

--
-- If the unit is not mentioned in the list then the unit is -
-- misc
--

context number
inv unitmatch:
    (containsunit.lunit = unit) or unit = 'Misc'
CHAPTER 5 - Component Design

Introduction

The purpose of this document is to provide the component design of KSNES. The documentation will present the class diagrams and tables that present the description of the attributes and methods.

Background

The purpose of the project is to extract the numerical information from the given raw text and present the values, units and the numerical phrase type. The whole system is set as a service on the server and a webpage is created for the user to enter the raw text and view the required information.

KDD- Service based Numerical Entity Searcher

Package View

As discussed in the Vision document, the present system has client server architecture. The webpage acts as the client and the KSNES system acts as the server. In this document, details of Numerical Phrase Extractor and Number-Unit/Date Pattern Recognizer are presented since the developer has worked on these modules and the POS Tagger is taken from a third party and hence not documented.

In the below package view, we see the three main modules of the server as three packages that work together in sequence to process the raw text and extract the numerical phrases from it.

We see that Number-Unit/Date Phrase Recognizer depends on Numerical Phrase Extractor and this in turn depends on POS Tagger.

Figure 5.1 Package View
**Class Diagram**

- **PatternMatch**: This class starts the NPE package working by calling a method from MyFunctions to start reading the given tagged input file.

**Table 5.1 Detailed Description of the PatternMatch Class**

<table>
<thead>
<tr>
<th>Class</th>
<th>Visibility</th>
<th>Extends</th>
<th>Implements</th>
</tr>
</thead>
<tbody>
<tr>
<td>PatternMatch</td>
<td>public</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>Attribute</td>
<td>Visibility</td>
<td>Type</td>
<td>Other</td>
</tr>
<tr>
<td>mfunction</td>
<td>private</td>
<td>MyFunction</td>
<td></td>
</tr>
</tbody>
</table>
- **MyFunctions**: This class holds the functions to read from the input file, writes the map values to the output file, delete the files and parse the line with the patterns available in the MyPatterns class.

### Table 5.2 Detailed Description of the MyFunctions Class

<table>
<thead>
<tr>
<th>Class</th>
<th>Visibility</th>
<th>Extends</th>
<th>Implements</th>
</tr>
</thead>
<tbody>
<tr>
<td>MyFunctions</td>
<td>public</td>
<td>none</td>
<td>none</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Visibility</th>
<th>Type</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>tmp</td>
<td>private</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>iline</td>
<td>private</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>fchunk</td>
<td>private</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>passon</td>
<td>private</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>fis</td>
<td>private</td>
<td>FileInputStream</td>
<td></td>
</tr>
<tr>
<td>bis</td>
<td>private</td>
<td>BufferedInputStream</td>
<td></td>
</tr>
<tr>
<td>dis</td>
<td>private</td>
<td>DataInputStream</td>
<td></td>
</tr>
<tr>
<td>outputfile</td>
<td>private</td>
<td>FileWriter</td>
<td></td>
</tr>
<tr>
<td>bw</td>
<td>private</td>
<td>BufferedWriter</td>
<td></td>
</tr>
<tr>
<td>mchunker</td>
<td>private</td>
<td>Map&lt;Integer, String&gt;</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>Visibility</th>
<th>Parameters</th>
<th>Returns</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>startparsing</td>
<td>public</td>
<td>void</td>
<td>void</td>
<td>Reads each line from the input file and send it to parseline function</td>
</tr>
<tr>
<td>postchunks</td>
<td>public</td>
<td>void</td>
<td>void</td>
<td>Writes data on to the output file</td>
</tr>
<tr>
<td>deletefile</td>
<td>public</td>
<td>void</td>
<td>void</td>
<td>Deletes the previously generated file containing the chunks</td>
</tr>
</tbody>
</table>
- **MyPatterns**: This class contains all the patterns and each pattern takes the input line and when found any match, is written to the map. This map is printed back to the file by calling a function from MyFunctions class.

**Table 5.3 Detailed Description of the MyPatterns Class**

<table>
<thead>
<tr>
<th>Class</th>
<th>Visibility</th>
<th>Extends</th>
<th>Implements</th>
</tr>
</thead>
<tbody>
<tr>
<td>MyPatterns</td>
<td>public</td>
<td>none</td>
<td>none</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Visibility</th>
<th>Type</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>chunk</td>
<td>private</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>p_words</td>
<td>private</td>
<td>Pattern</td>
<td></td>
</tr>
<tr>
<td>mf</td>
<td>private</td>
<td>MyFunctions</td>
<td></td>
</tr>
<tr>
<td>chunkorder</td>
<td>private</td>
<td>Map&lt;Integer,String&gt;</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>Visibility</th>
<th>Parameters</th>
<th>Returns</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_abtfrac</td>
<td>public</td>
<td>String</td>
<td>String</td>
<td>Extracts numerical phrases like ‘about two-thirds of the vote’, ‘millions of unregulated dollars’</td>
</tr>
<tr>
<td>p_age</td>
<td>public</td>
<td>String</td>
<td>String</td>
<td>Extracts numerical phrases like ‘27 year-old’, ‘two clear opportunities’, ‘a 20-story’</td>
</tr>
<tr>
<td>p_ampm</td>
<td>public</td>
<td>String</td>
<td>String</td>
<td>Extracts numerical phrases like ‘3:00 a.m.’, ‘4:15 p.m. CST’</td>
</tr>
<tr>
<td>------------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>-------------------------------------------------------------</td>
</tr>
<tr>
<td>p_and</td>
<td>public</td>
<td>String</td>
<td>String</td>
<td>Extracts numerical phrases like ‘4,952 children and adolescents’</td>
</tr>
<tr>
<td>p_btwfrmd</td>
<td>public</td>
<td>String</td>
<td>String</td>
<td>Extracts numerical phrases like ‘from 200 to 300 miles’, ‘151 to 200 people’, ‘from 7.5 percent to 10.5 percent’</td>
</tr>
<tr>
<td>p_date</td>
<td>public</td>
<td>String</td>
<td>String</td>
<td>Extracts numerical phrases like ‘18 April 2008’</td>
</tr>
<tr>
<td>p_days</td>
<td>public</td>
<td>String</td>
<td>String</td>
<td>Extracts weekdays like ‘Monday’, ‘Sunday’</td>
</tr>
<tr>
<td>p_centuary</td>
<td>public</td>
<td>String</td>
<td>String</td>
<td>Extracts numerical phrases like ‘17(^{th}) centaury’</td>
</tr>
<tr>
<td>p_hyphenww</td>
<td>public</td>
<td>String</td>
<td>String</td>
<td>Extracts numerical phrases like ‘million-dollar home’, ‘thirty-three dollars’</td>
</tr>
<tr>
<td>p_hyphennumnum</td>
<td>public</td>
<td>String</td>
<td>String</td>
<td>Extracts numerical phrases like ‘the 20-20 match’, ‘a 3-2 lead’</td>
</tr>
<tr>
<td>----------------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>p_in</td>
<td>public</td>
<td>String</td>
<td>String</td>
<td>Extracts numerical phrases like ‘9 in 10 people’</td>
</tr>
<tr>
<td>p_mids</td>
<td>public</td>
<td>String</td>
<td>String</td>
<td>Extracts numerical phrases like ‘mid-1990s’, ‘the early 1990s’, ‘1980s’</td>
</tr>
<tr>
<td>p_months</td>
<td>public</td>
<td>String</td>
<td>String</td>
<td>Extracts month names like ‘Jan’, ‘May’, ‘Dec’</td>
</tr>
<tr>
<td>p_numunit</td>
<td>public</td>
<td>String</td>
<td>String</td>
<td>Extracts numerical phrases like ‘33 USD’, ‘3,400 miles’, ‘four season tickets’, ‘dlrs 3.4 billion’</td>
</tr>
<tr>
<td>p_per</td>
<td>public</td>
<td>String</td>
<td>String</td>
<td>Extracts numerical phrases like ‘$33 per day’, ‘100 miles per hour’</td>
</tr>
<tr>
<td>p_percent</td>
<td>public</td>
<td>String</td>
<td>String</td>
<td>Extracts numerical phrases like ‘39%’, ‘0.5-1%’, ‘about 90%’</td>
</tr>
<tr>
<td>p_ratio</td>
<td>public</td>
<td>String</td>
<td>String</td>
<td>Extracts numerical phrases like ‘one of the five people’, ‘two of the groups’, ‘89 percent of people’, ‘3 out of 5 people’</td>
</tr>
<tr>
<td>p_tty</td>
<td>public</td>
<td>String</td>
<td>String</td>
<td>Extracts numerical phrases like ‘today’, ‘tomorrow’,</td>
</tr>
<tr>
<td>Method</td>
<td>Access</td>
<td>Return Type</td>
<td>Parameter Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>--------</td>
<td>-------------</td>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>p_twmy</td>
<td>public</td>
<td>String</td>
<td>String</td>
<td>‘yesterday’, ‘noon’</td>
</tr>
<tr>
<td>p_xbits</td>
<td>public</td>
<td>String</td>
<td>String</td>
<td>Extracts numerical phrases like ‘this year’, ‘next month’, ‘last week’</td>
</tr>
<tr>
<td>p_yrange</td>
<td>public</td>
<td>String</td>
<td>String</td>
<td>Extracts numerical phrases like ‘320GB’, ‘500TB’</td>
</tr>
</tbody>
</table>

**Number-Unit/Date Pattern Recognizer Package**

Figure 5.3 NDPR Class Diagram
**Class Description**

- **Main**: This is the class that reads the input from the file and calls the number or the date functions based on the type of the input.

**Table 5.4 Detailed Description of the Main Class**

<table>
<thead>
<tr>
<th>Class</th>
<th>Visibility</th>
<th>Extends</th>
<th>Implements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main</td>
<td>public</td>
<td>none</td>
<td>none</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>Visibility</th>
<th>Parameters</th>
<th>Returns</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>batch_process</td>
<td>private</td>
<td>String</td>
<td>void</td>
<td>Reads the input file</td>
</tr>
<tr>
<td>main</td>
<td>private</td>
<td>int, char</td>
<td>int</td>
<td>Main function that calls number and date classes</td>
</tr>
</tbody>
</table>

- **Number**: This class parses the given number phrase and evaluates the value, unit, unit-type and the bound corresponding to that phrase.

**Table 5.5 Detailed Description of the Number Class**

<table>
<thead>
<tr>
<th>Class</th>
<th>Visibility</th>
<th>Extends</th>
<th>Implements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>public</td>
<td>none</td>
<td>none</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Visibility</th>
<th>Type</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>unit</td>
<td>private</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>cat</td>
<td>private</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>private</td>
<td>double</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>Visibility</th>
<th>Parameters</th>
<th>Returns</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>get_units</td>
<td>private</td>
<td>none</td>
<td>void</td>
<td>Verifies if the current unit is in the units list</td>
</tr>
<tr>
<td>is_equal</td>
<td>private</td>
<td>Number</td>
<td>bool</td>
<td>Verifies if it is the exact value or else does it has words like ‘above’, ‘around’, ‘more than’</td>
</tr>
<tr>
<td>print</td>
<td>private</td>
<td>none</td>
<td>void</td>
<td>Prints the chunk, chunk-type, value,</td>
</tr>
</tbody>
</table>
- **Date**: This class parses the given date phrase and prints the date as mm/dd/yyyy. If there is any second date it is also parsed as mm/dd/yyyy.

**Table 5.6 Detailed Description of the Date Class**

<table>
<thead>
<tr>
<th>Class</th>
<th>Visibility</th>
<th>Extends</th>
<th>Implements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>public</td>
<td>none</td>
<td>none</td>
</tr>
</tbody>
</table>

**Attribute Visibility Type Other**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Visibility</th>
<th>Type</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>month</td>
<td>private</td>
<td>int</td>
<td></td>
</tr>
<tr>
<td>day</td>
<td>private</td>
<td>int</td>
<td></td>
</tr>
<tr>
<td>year</td>
<td>private</td>
<td>int</td>
<td></td>
</tr>
<tr>
<td>format</td>
<td>private</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>modifier</td>
<td>private</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>second_date</td>
<td>private</td>
<td>Date</td>
<td></td>
</tr>
</tbody>
</table>

**Function Visibility Parameters Returns Actions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Visibility</th>
<th>Parameters</th>
<th>Returns</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>get_modifer</td>
<td>public</td>
<td>void</td>
<td>void</td>
<td>Checks if the given phrase is relative, regular or difference phrase</td>
</tr>
<tr>
<td>make_relative_date</td>
<td>public</td>
<td>int</td>
<td>void</td>
<td>helper function to make the correct date out of a relative term like ‘today’, ‘yesterday’, ‘last week’</td>
</tr>
<tr>
<td>is_equal</td>
<td>public</td>
<td>Date</td>
<td>bool</td>
<td>Verifies if it is the exact value or else does it has words like ‘above’, ‘around’, ‘more than’</td>
</tr>
<tr>
<td>print</td>
<td>public</td>
<td>void</td>
<td>void</td>
<td>Prints the chunk, chunk-type, date, modifier, relative date if any</td>
</tr>
</tbody>
</table>

46
- **Quantity**: This class has the basic functions defined that are used by the number and the date classes.

### Table 5.7 Detailed Description of the Quantity Class

<table>
<thead>
<tr>
<th>Class</th>
<th>Visibility</th>
<th>Extends</th>
<th>Implements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quantity</td>
<td>public</td>
<td>none</td>
<td>none</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Visibility</th>
<th>Type</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>mapcmp</td>
<td>private</td>
<td>Struct</td>
<td></td>
</tr>
<tr>
<td>dbl2string</td>
<td>private</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>int2string</td>
<td>private</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>private</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>phrase</td>
<td>private</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>bound</td>
<td>private</td>
<td>char</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>Visibility</th>
<th>Parameters</th>
<th>Returns</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>match</td>
<td>public</td>
<td>String, string, vector&lt;string&gt;</td>
<td>bool</td>
<td>matches string and returns matched subexpressions in a vector of strings</td>
</tr>
<tr>
<td>subst</td>
<td>public</td>
<td>string, string, string</td>
<td>string</td>
<td>a substitution regex function: takes string s and replaces every instance of &lt;pattern&gt; with &lt;replace&gt;</td>
</tr>
<tr>
<td>split</td>
<td>public</td>
<td>string, string</td>
<td>vector&lt;string&gt;</td>
<td>a split regex function:</td>
</tr>
<tr>
<td>convert_to_integer</td>
<td>public</td>
<td>string</td>
<td>string</td>
<td>Converts words to number values</td>
</tr>
<tr>
<td>get_bound</td>
<td>public</td>
<td>void</td>
<td>void</td>
<td>Based on the words like ‘above’, ‘around’, ‘more than’ determines if the bound to be ‘=’, ‘&gt;’, ‘&lt;’, ‘~’</td>
</tr>
<tr>
<td>is_equal</td>
<td>public</td>
<td>void</td>
<td>void</td>
<td>Verifies if it is the exact value or else does it have words like ‘above’, ‘around’, ‘more than’</td>
</tr>
</tbody>
</table>
CHAPTER 6 - Software Test Plan

Test Plan Identifier

KSNES-Validation-V-1.0

Introduction

The document provides the methods that will be used to test the KSNES. The project allows the user to give in the raw text and presents the numbers, units and the dates, if any, in the given input. The data analyzed is displayed on the webpage for the user to view it. Each task is treated as a separate module of the system and will be tested with respect to the associated requirements described in the vision document.

Test Items

The following items will be tested:

- General Application Related Items

Tested Features

All features listed below will be tested.

General Application Related Items

- **ARI 1-** The program shall provide a GUI for the user interaction.
- **ARI 2-** The application shall execute in a single step with no user interaction.
- **ARI 3-** The application shall start when user enters the text into the text box in webpage.
- **ARI 4-** The application shall invoke the other modules of the project when the user clicks on the send button.
- **ARI 5-** The system will be setup as a service accessible over the K-State network.
- **ARI 6-** The user shall be able to view the chunked output on the webpage.
• **ARI 7**- The user shall be able to stop the running of the application after viewing the output by closing the web browser.

• **ARI 8**- The user shall be able to run the application again with a new input ones the previously entered text is chunked.

• **ARI 9**- The user shall be able to enter the input of any size.

**Features not to be tested**

Testing on the following requirements will not be tested because certain parts of the system are adopted from third parties who have already tested the systems. Most part of the system is a black box to the user and hence a tester will not be testing it. The system will be tested by the developer while developing the system.

**POS tagger Items**

• **PTRI 1**- The POS Tagger shall be given the raw input text that user enters on the webpage.

• **PTRI 2**- This POS Tagger shall be a service request from the main webpage.

• **PTRI 3**- The POS Tagger shall be able to tokenize the given text.

• **PTRI 4**- The POS Tagger shall be able to tag the words, punctuations and symbols in the sentence using the Penn Treebank Tag set.

• **PTRI 5**- The POS Tagger shall be able to produce the tagged sentence. This is a critical requirement because the module should be producing the expected output.

• **PTRI 6**- The POS Tagger shall be able to send the tagged sentence to the next module which is the Numerical Phrase Extractor.

**Numerical Phrase Extractor Items**

• **NPERI 1**- The Numerical Phrase Extractor shall be able to take the tagged sentence from the POS Tagger.

• **NPERI 2**- The Numerical Phrase Extractor shall be able to identify the tagged words that may be containing the numbers and the units.

• **NPERI 3**- The Numerical Phrase Extractor shall be able to identify the tagged words that may be containing the dates.
• **NPERI 4-** The Numerical Phrase Extractor shall be able to produce the filtered number-unit or the date phrase.

• **NPERI 5-** The Numerical Phrase Extractor shall be able to send the filtered phrase to the next module which is the Number-Unit/Date Pattern Recognizer.

**Number-Unit/Date Pattern Recognizer Items**

• **NDPRRI 1-** The Number-Unit/Date Pattern Recognizer shall be able to take the extracted phrase from the Numerical Phrase Extractor.

• **NDPRRI 2-** The Number-Unit/Date Pattern Recognizer shall be able to identify the numbers, units and unit-type if present in the phrase.

• **NDPRRI 3-** The Number-Unit/Date Pattern Recognizer shall be able to able to identify the date, month and the year if present in the given phrase.

• **NDPRRI 4-** The Number-Unit/Date Pattern Recognizer shall be able to produce the number, unit corresponding to it and the type to which the unit belongs to if it’s a number phrase. And if it is a temporal phrase then the module should be able to display the date based on the number, month and year information in the phrase.

• **NDPRRI 5-** The Number-Unit/Date Pattern Recognizer shall be able to display the value, unit, and unit-type to the user on the webpage.

**Approach**

Testing will be done by running KSNES in a separate series of actions. The sequence of actions, expected result and the requirements of each test case is mentioned.

**Item Pass/Fail Criteria**

A test case is said to be passed if it meets the requirements in the vision document and it is considered to be failed if does not meet the requirements described in the vision document.

**Suspension Criteria and Resumption Requirements**

**Suspension Criteria**

Testing will be halted if a test case fails while testing. The reason for the failure and a suggested solution will be logged near the test case.
**Resumption Requirements**

The failed test cases will be rerun from the beginning of the test after logging the test in the test log with the possible solution to the problem. Other independent test cases are run parallel with the effort to fix problems that occur in independent areas.

**Test Deliverables**

The provided Test Log will document all the test cases and record if the test case is passed or failed. Comments will be added when a test case has failed and the reason and the solutions are documented.

**Testing Tasks**

**Test Case 1: Application Items**

Table 6.1 Application Items Test Case

<table>
<thead>
<tr>
<th>Step#</th>
<th>Action Performed</th>
<th>Expected Outcome</th>
<th>Requirements Met</th>
<th>Pass/Fail</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Tester opens the webpage in a browser</td>
<td>Observe the KSNES webpage open Tester sees a text box, send button</td>
<td>ARI 1 ARI 2</td>
<td>Pass</td>
</tr>
<tr>
<td>2</td>
<td>Enters the text in the box and clicks the send button to send the request to the service</td>
<td>Observes the raw text to be sent by the tester on clicking the send button The request is sent to the server</td>
<td>ARI 3 ARI 4 ARI 5</td>
<td>Pass</td>
</tr>
<tr>
<td>3</td>
<td>The tester views chunks, value, units and unit type on the webpage</td>
<td>The output is seen on a webpage explaining the chunks, values, units and unit-types</td>
<td>ARI 6</td>
<td>Pass</td>
</tr>
<tr>
<td>4</td>
<td>The tester may close the browser in between or after viewing the output</td>
<td>The application stops and the browser closes</td>
<td>ARI 7</td>
<td>Pass</td>
</tr>
<tr>
<td>5</td>
<td>The tester may enter a new input</td>
<td>The text box appears again for the</td>
<td>ARI 8</td>
<td>Pass</td>
</tr>
</tbody>
</table>
The tester may enter the input string of any length.

**Test Case 2: Patterns and Matching Text**

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Numerical Phrase</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>I lost 33 dollars since 1998</td>
<td>33 dollars</td>
<td>p_numunit, p_btwfrm</td>
</tr>
<tr>
<td>At just 12-years-old, he enrolled as a freshman at F.I.U. in Miami.</td>
<td>12-years-old</td>
<td>p_age</td>
</tr>
<tr>
<td>The 20&quot; iMac is cheaper at $1200, but doesn't carry as much value: It</td>
<td>20&quot; $ 1200 2GB RAM</td>
<td>p_percentinches,</td>
</tr>
<tr>
<td>comes standard with only 2GB RAM.</td>
<td></td>
<td>p_numunit, p_xbits</td>
</tr>
<tr>
<td>Volunteers bring in a heavy crane for work on a bridge to Polihale</td>
<td>last month</td>
<td>p_twmy</td>
</tr>
<tr>
<td>State Park on Kauai last month.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>As for those who do not invest, around 40% say capitalism is better</td>
<td>around 40 % more than</td>
<td>p_percentinches,</td>
</tr>
<tr>
<td>while more than 25% prefer socialism.</td>
<td>25%</td>
<td>p_percentinches</td>
</tr>
<tr>
<td>A residual force of between 35,000 to 50,000 troops will remain until</td>
<td>between 35,000 to 50,000</td>
<td>p_btwfrmd</td>
</tr>
<tr>
<td>December 31, 2011.</td>
<td>troops December 31, 2011</td>
<td>p_anydate</td>
</tr>
<tr>
<td>6 January 2006, Rome - Millions of people are on the brink of starvation</td>
<td>6 January 2006 Millions of</td>
<td>p_date p_abtfrac</td>
</tr>
<tr>
<td>in the Horn of Africa due to recent severe droughts.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sowjanya will be giving her final MSE presentation on Tuesday from</td>
<td>Tuesday 12:00 13:00 CST</td>
<td>p_days p_numunit p_</td>
</tr>
<tr>
<td>12:00 to 13:00 CST.</td>
<td></td>
<td>ampmp</td>
</tr>
<tr>
<td>Current listings of 2,000 children and adults who are reported missing, including in-depth coverage of high-profile cases.</td>
<td>2,000 children and adults</td>
<td>p_and</td>
</tr>
<tr>
<td>----------------------------------------------------------</td>
<td>--------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Between 1990 and 1998, there were an estimated 22,000 bird-aircraft collisions in the United States.</td>
<td>Between 1990 and 1998 22,000 bird-aircraft collisions</td>
<td>p_btwfrm p_numunit</td>
</tr>
<tr>
<td>A six-bedroom mansion in the Beverly Hills, Calif. postal region that rocker Adam Duritz once owned has been placed on the market for $8,250,000.</td>
<td>six-bedroom mansion $8,250,000</td>
<td>p_hypenww p_numunit</td>
</tr>
<tr>
<td>Peterson's three capped the surge, giving New Orleans a 64-51 lead</td>
<td>Three 64-51 lead</td>
<td>p_numunit p_hyphennunnum</td>
</tr>
<tr>
<td>1 in every 8 women is effected by breast cancer.</td>
<td>1 in every 8 women</td>
<td>p_in</td>
</tr>
<tr>
<td>She became an exotic dancer at Scores in New York City in the mid-1990s.</td>
<td>mid-1990s</td>
<td>p_mids</td>
</tr>
<tr>
<td>Economists predict the recession will end in September, but views for the labor market remain bleak, a Wall Street Journal survey found</td>
<td>September</td>
<td>p_months</td>
</tr>
<tr>
<td>Big Savage Mountain, at night fall, with only a headlamp and sharing a no-berm road with cars and pickups doing 60 miles per hour.</td>
<td>60 miles per hour</td>
<td>p_per</td>
</tr>
<tr>
<td>The 10 out of 10 team however bring then a combined experience of 100 years within the entertainment industry.</td>
<td>10 out of 10 team 100 years</td>
<td>p_ratio p_numunit</td>
</tr>
<tr>
<td>Yesterday was the last day to pay the spring semester fee.</td>
<td>Yesterday</td>
<td>p_tty</td>
</tr>
<tr>
<td>The 1970-71 season was the best of competitive football in England</td>
<td>1970-71</td>
<td>p_yrange</td>
</tr>
</tbody>
</table>
CHAPTER 7 - User Manual

Introduction

This document describes how to use the KSNES. This project is set up as a service and is accessed by the user through a webpage.

Application Setup

Required Software

- A web browser is sufficient to use this service.

Running KSNES

- The application is setup as a service on the CIS server and is made accessible to the users through the URL http://viper.cis.ksu.edu:11603/numerical/. Using this service is as simple as using a general webpage.
- On opening the above link in a browser a webpage as shown in the below screenshot is seen. User enters the raw text in the text box and click the submit button to invoke the service.

Figure 7.1 Input Screen of the Webpage
After a while a new webpage, similar to the one in the below screenshot, appears to the user displaying the input, chunk, value, unit, unit-type, date, second date etc.

**Figure 7.2 Output Screen of the Webpage**

- The user can either enter new text to run the application again or can close the browser to abort the application.
CHAPTER 8 - Test Assessment and Evaluation

Introduction
This document presents the functionality testing results of the KSNES project. Since the system is set as a service to the user a black box testing is performed.

Test Results

Table 8.1 Test Results Summary

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Main Functionality Tested</th>
<th>Pass/Fail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Case 1</td>
<td>Application Functionality</td>
<td>Pass</td>
</tr>
<tr>
<td>Test Case 2</td>
<td>POS Tagger Functionality</td>
<td>Pass</td>
</tr>
<tr>
<td>Test Case 3</td>
<td>Numerical Phrase Extractor Functionality</td>
<td>Pass</td>
</tr>
<tr>
<td>Test Case 4</td>
<td>Number-Unit/Date Pattern Recognizer Functionality</td>
<td>Pass</td>
</tr>
</tbody>
</table>

The testing for the test case 1 is done as shown in the next section for the remaining test cases it is done manually by the developer as the whole system is a black box for the user.

Complete Test Results

Test Case 1: Application Items
This test case tests the basic application items
Prerequisites: The system should be set up on the server so as to be accessible to the user through a webpage.
Date Performed: 04/11/09
Issues Found: None
Comments: Test ran perfectly
Table 8.2 Test Application Items

<table>
<thead>
<tr>
<th>Step#</th>
<th>Action Performed</th>
<th>Expected Outcome</th>
<th>Requirements Met</th>
<th>Pass/Fail</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Tester opens the webpage in a browser</td>
<td>Observe the KSNES webpage open</td>
<td>ARI 1</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tester sees a text box, send button</td>
<td>ARI 2</td>
<td>Pass</td>
</tr>
<tr>
<td>2</td>
<td>Enters the text in the box and clicks the send button to send the request to the service</td>
<td>Observes the raw text to be sent by the tester on clicking the send button</td>
<td>ARI 3</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The request is sent to the server</td>
<td>ARI 4</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ARI 5</td>
<td>Pass</td>
</tr>
<tr>
<td>3</td>
<td>The tester views chunks, value, units and unit type on the webpage</td>
<td>The output is seen on a webpage explaining the chunks, values, units and unit-types</td>
<td>ARI 6</td>
<td>Pass</td>
</tr>
<tr>
<td>4</td>
<td>The tester may close the browser in between or after viewing the output</td>
<td>The application stops and the browser closes</td>
<td>ARI 7</td>
<td>Pass</td>
</tr>
<tr>
<td>5</td>
<td>The tester may enter a new input again</td>
<td>The text box appears again for the user allowing to give a new input</td>
<td>ARI 8</td>
<td>Pass</td>
</tr>
<tr>
<td>6</td>
<td>The tester may enter the input string of any length</td>
<td>Tester enters the string of any length</td>
<td>ARI 9</td>
<td>Pass</td>
</tr>
</tbody>
</table>

**Overall Results**

KSNES passed the formal qualification testing and is ready for the final MSE presentation.
CHAPTER 9 - Project Evaluation

Introduction

This document speaks about the experiences I had while working on KSNES project in this semester. This document includes a time log analysis, a source code analysis and the problems encountered and lessons learned. There is also a section that presents the possible extensions that can be made to the project.

Problems Encountered

While working on the project there has been certain problems that I have encountered and they are explained below:

Determining the Patterns

There has been lot number of ways that a numerical phrase could occur in text. Initially lot of patterns have been designed to match each kind of numerical phrase but later on after severe testing the number of patterns are decreased such that one pattern can filter more than one type of numerical phrases.

Prioritizing the Patterns

There are certain numerical phrases which are detected by more than one pattern. To make the numerical phrases picked by a more suitable pattern the patterns are given priorities and determining this priority had been a bit difficult at the initial stages of designing the patterns.

Source Lines of Code (SLOC)

SLOC of 1000 is estimated at the end of the each phase and at the end of Phase 1 based on other numerical phrase extractors. At the end of the phase two, a new estimate of 800 SLOC is made because the coding is decreased by using regular expressions to parse the numerical phrases.

Initially it is assumed that the interfacing the gluing the modules would take some coding but since it done by one of the KDD members, that source lines of code is ignored at the end of
phase 2. Overall, I think I made appropriate estimated from the beginning of the project and I would definitely like to improve the project to make it more efficient and effective.

**Project Duration**

The following table shows the estimated dates and the actual dates of the presentation in all the three phases. The actual completion dates are almost close to the estimated date schedule.

**Table 9.1 Project Duration**

<table>
<thead>
<tr>
<th>Phase</th>
<th>Expected Completion Phase</th>
<th>Actual Completion Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>February 26, 2009</td>
<td>February 24, 2009</td>
</tr>
<tr>
<td>2</td>
<td>March 26, 2009</td>
<td>March 31, 2009</td>
</tr>
<tr>
<td>3</td>
<td>April 14, 2009</td>
<td></td>
</tr>
</tbody>
</table>

The pie charts below show the time spent on the project at each phase.

**Figure 9.1 Phase Breakdown**

From the above graph, we can see that more time is spent in Phase 1 as it is the important phase where in one has to analyze the whole project and plan. Phase 2 took a considerable amount of time for coding and the Phase 3 took less time as most part of the work is done in the final phase.
From the above graph, we can see that more time is spent in coding and documentation as it is the important activity in any project. Reading and Presentation are the other two activities that took a good part of the remaining time.

**Figure 9.3 Phase1 Activity Breakdown**
From the above graph, we can see the time distribution of the project activities in Phase1. As it is the initial phase more time is spent documenting the findings and presenting the ideas to the committee.

**Figure 9.4 Phase2 Activity Breakdown**

![Phase 2 Activity Breakdown](image)

From the above graph, we can see the time distribution of the project activities in Phase2. As it the second phase more time is spent in coding and reading the project related material.

**Figure 9.5 Phase3 Activity Breakdown**

![Phase 3 Activity Breakdown](image)
From the above graph, we can see the time distribution of the project activities in Phase 3. As most of the code is done in Phase 2, the corresponding documentation and the overall project performance assessment is done.

Figure 9.6 Time Log

<table>
<thead>
<tr>
<th>Date</th>
<th>Start</th>
<th>End</th>
<th>Task Description</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/26/18</td>
<td>16:13</td>
<td>18:19</td>
<td>0.48 Discussion</td>
<td></td>
</tr>
<tr>
<td>1/27/18</td>
<td>18:30</td>
<td>20:45</td>
<td>0.48 Reading</td>
<td></td>
</tr>
<tr>
<td>1/28/18</td>
<td>21:00</td>
<td>23:15</td>
<td>0.25 Testing</td>
<td></td>
</tr>
<tr>
<td>1/29/18</td>
<td>23:15</td>
<td>25:30</td>
<td>0.55 Documentation</td>
<td></td>
</tr>
<tr>
<td>1/30/18</td>
<td>25:30</td>
<td>27:45</td>
<td>0.65 Reading</td>
<td></td>
</tr>
<tr>
<td>1/31/18</td>
<td>27:45</td>
<td>29:59</td>
<td>0.65 Testing</td>
<td></td>
</tr>
<tr>
<td>2/1/18</td>
<td>29:59</td>
<td>32:15</td>
<td>0.85 Documentation</td>
<td></td>
</tr>
<tr>
<td>2/2/18</td>
<td>32:15</td>
<td>34:30</td>
<td>0.85 Reading</td>
<td></td>
</tr>
<tr>
<td>2/3/18</td>
<td>34:30</td>
<td>36:45</td>
<td>0.85 Testing</td>
<td></td>
</tr>
<tr>
<td>2/4/18</td>
<td>36:45</td>
<td>38:55</td>
<td>0.85 Documentation</td>
<td></td>
</tr>
<tr>
<td>2/5/18</td>
<td>38:55</td>
<td>40:30</td>
<td>0.85 Reading</td>
<td></td>
</tr>
<tr>
<td>2/6/18</td>
<td>40:30</td>
<td>42:15</td>
<td>0.85 Testing</td>
<td></td>
</tr>
<tr>
<td>2/7/18</td>
<td>42:15</td>
<td>44:30</td>
<td>0.85 Documentation</td>
<td></td>
</tr>
<tr>
<td>2/8/18</td>
<td>44:30</td>
<td>46:30</td>
<td>0.85 Reading</td>
<td></td>
</tr>
<tr>
<td>2/9/18</td>
<td>46:30</td>
<td>48:30</td>
<td>0.85 Testing</td>
<td></td>
</tr>
<tr>
<td>2/10/18</td>
<td>48:30</td>
<td>50:15</td>
<td>0.85 Documentation</td>
<td></td>
</tr>
<tr>
<td>2/11/18</td>
<td>50:15</td>
<td>52:15</td>
<td>0.85 Reading</td>
<td></td>
</tr>
<tr>
<td>2/12/18</td>
<td>52:15</td>
<td>54:30</td>
<td>0.85 Testing</td>
<td></td>
</tr>
<tr>
<td>2/13/18</td>
<td>54:30</td>
<td>56:15</td>
<td>0.85 Documentation</td>
<td></td>
</tr>
<tr>
<td>2/14/18</td>
<td>56:15</td>
<td>58:15</td>
<td>0.85 Reading</td>
<td></td>
</tr>
<tr>
<td>2/15/18</td>
<td>58:15</td>
<td>60:15</td>
<td>0.85 Testing</td>
<td></td>
</tr>
<tr>
<td>2/16/18</td>
<td>60:15</td>
<td>62:15</td>
<td>0.85 Documentation</td>
<td></td>
</tr>
<tr>
<td>2/17/18</td>
<td>62:15</td>
<td>64:15</td>
<td>0.85 Reading</td>
<td></td>
</tr>
<tr>
<td>2/18/18</td>
<td>64:15</td>
<td>66:15</td>
<td>0.85 Testing</td>
<td></td>
</tr>
<tr>
<td>2/19/18</td>
<td>66:15</td>
<td>68:15</td>
<td>0.85 Documentation</td>
<td></td>
</tr>
<tr>
<td>2/20/18</td>
<td>68:15</td>
<td>70:15</td>
<td>0.85 Reading</td>
<td></td>
</tr>
<tr>
<td>2/21/18</td>
<td>70:15</td>
<td>72:15</td>
<td>0.85 Testing</td>
<td></td>
</tr>
<tr>
<td>2/22/18</td>
<td>72:15</td>
<td>74:15</td>
<td>0.85 Documentation</td>
<td></td>
</tr>
<tr>
<td>2/23/18</td>
<td>74:15</td>
<td>76:15</td>
<td>0.85 Reading</td>
<td></td>
</tr>
<tr>
<td>2/24/18</td>
<td>76:15</td>
<td>78:15</td>
<td>0.85 Testing</td>
<td></td>
</tr>
<tr>
<td>2/25/18</td>
<td>78:15</td>
<td>80:15</td>
<td>0.85 Documentation</td>
<td></td>
</tr>
<tr>
<td>2/26/18</td>
<td>80:15</td>
<td>82:15</td>
<td>0.85 Reading</td>
<td></td>
</tr>
<tr>
<td>2/27/18</td>
<td>82:15</td>
<td>84:15</td>
<td>0.85 Testing</td>
<td></td>
</tr>
<tr>
<td>2/28/18</td>
<td>84:15</td>
<td>86:15</td>
<td>0.85 Documentation</td>
<td></td>
</tr>
<tr>
<td>2/29/18</td>
<td>86:15</td>
<td>88:15</td>
<td>0.85 Reading</td>
<td></td>
</tr>
<tr>
<td>3/1/18</td>
<td>88:15</td>
<td>90:15</td>
<td>0.85 Testing</td>
<td></td>
</tr>
<tr>
<td>3/2/18</td>
<td>90:15</td>
<td>92:15</td>
<td>0.85 Documentation</td>
<td></td>
</tr>
<tr>
<td>3/3/18</td>
<td>92:15</td>
<td>94:15</td>
<td>0.85 Reading</td>
<td></td>
</tr>
<tr>
<td>3/4/18</td>
<td>94:15</td>
<td>96:15</td>
<td>0.85 Testing</td>
<td></td>
</tr>
<tr>
<td>3/5/18</td>
<td>96:15</td>
<td>98:15</td>
<td>0.85 Documentation</td>
<td></td>
</tr>
<tr>
<td>3/6/18</td>
<td>98:15</td>
<td>100:00</td>
<td>0.85 Reading</td>
<td></td>
</tr>
</tbody>
</table>

63
The above figure is the time log of the project.

**Lessons Learned**

There has been a lot of learning for this project and I am sure that this knowledge would be really helpful in future:

**Eclipse IDE**

I have worked on C++ in my undergrad projects and I have never used Java or Eclipse apart from using them for very small project in some of the graduate courses. This is the first time I have developed a complete application using both Java and the Eclipse IDE. It took some time on learning the new interface and the language but once I started working on it I could realize the efficiency and power of this tool. Learning this tool has been a good learning experience and added confidence to work on more projects in Java.

**Creation of Design Documents**

In all the projects I have worked till now, more emphasis was given on implementing and a very basic report is written. But by working on this project I could get an opportunity to go through the whole software development life cycle and develop required documents at each phase. It is a good learning experience to do the implementation along with intense documentation using Microsoft Visio and USE OCL.

**Future Work**

These are the areas in which there could be a scope of improvement.

**Adding more patterns to the NPE Package**

There are certain numerical phrases that are not being filtered by the current system and hence some more new patterns can be added to the system to make it more effective. These newly added patterns would be added in such a way that they do not clash with previous patterns.

**Improving NDPR Package**

In order to analyze and process the filtered numerical phrases from the NPE package, more conditions had to be added to classes in the NDPR package. These conditions should be added so that they do not interfere with the previous conditions.
Improving the Output Display

Displaying the output can be done in a more effective manner by assigning different colors to the date and number types.
CHAPTER 10 - Technical Inspection Checklist

Introduction

This document provides a checklist to be used in the technical inspection of KSNES project. It provides a guideline for the inspectors to follow to ensure that the Architectural Design Document and the USE formal specification model are both complete and correct.

Items to be Inspected

Technical inspectors refer Vision Document 1.0 and Architectural Design for technical inspection.

UML Diagrams

- Class Diagrams
- Sequence Diagrams

Formal Specification

- OCL Model

Formal Inspectors

- Svitlana Volkova (svitlana@ksu.edu)
- Snehal Monterio (snehalm@ksu.edu)

Formal Inspection Checklist

The below table shows the items that are considered by the inspectors for verifying the correctness of the

Table 10.1 Formal Inspection Checklist

<table>
<thead>
<tr>
<th>Item #</th>
<th>Inspection Item</th>
<th>Pass/Fail/Partial</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>TI- 1</td>
<td>All the symbols used in the class diagrams are according to the UML standards</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TI-2</td>
<td>All the classes in the class diagram are clear as to what they represent in the architecture design document</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>---------------------------------------------------------------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TI-3</td>
<td>The symbols used in the sequence diagram correspond to UML standards</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TI-4</td>
<td>Sequence diagram matches class diagram</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TI-5</td>
<td>All the classes in the OCL model are represented in the class diagram</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TI-6</td>
<td>The multiplicities in the OCL model have been depicted in the class diagram</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TI-7</td>
<td>All the requirements in the Software Requirements Specification have been covered in the Architecture Design Document</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The Technical Inspection Checklist from the technical inspectors is put in Appendix A.
CHAPTER 11 - Software Technical Instruction

Purpose

This document presents the details about reusing and improving the current project. For project reuse, it explains the file types to be read and modified. For project extensions, the paper explains how the features can be done to the project.

Reusing the Project

KSNES is set up on the server of the KIIAC and access to this can be available by contacting the administrator of that webpage. Since the whole system is set as a service, a different interface can be used to connect to it.

Project Extension

As explained in the Project Evaluation Report the following changes can be made to the current system in order to improve its efficiency.

Adding more patterns to the NPE Package

There are certain numerical phrases that are not being filtered by the current system and hence some more new patterns can be added to the system to make it more effective. These newly added patterns would be added in such a way that they do not clash with previous patterns.

Improving NDPR Package

In order to analyze and process the filtered numerical phrases from the NPE package, more conditions had to be added to classes in the NDPR package. These conditions should be added so that they do not interfere with the previous conditions.

Improving the Output Display

Displaying the output can be done in a more effective manner by assigning different colors to the date and number types.
References

[1] Bennett, N., He, Q., Powell, K., & Schatz, B. Extracting Noun Phrases for all of MEDLINE, CANIS - Community Architectures for Network Information Systems, Graduate School of Library and Information Science, University of Illinois at Urbana-Champaign, IL, USA.


[10] Roth, D., Sapp B & Lev-Arie, R. Numerical and Temporal Quantities in Textual Entailment, Department of Computer Science, University of Illinois Urbana-Champaign, IL, USA.


Appendix A - Formal Technical Inspection Letters

The Technical Inspection Letters submitted by the technical inspectors are included in this section.

**Formal Technical Inspection Letter by Svitlana Volkova**

Table A.1 Technical Inspection Checklist from Svitlana

<table>
<thead>
<tr>
<th>Item #</th>
<th>Inspection Item</th>
<th>Pass/Fail/Partial</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>TI- 1</td>
<td>All the symbols used in the class diagrams are according to the UML standards</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>TI- 2</td>
<td>All the classes in the class diagram are clear as to what they represent in the architecture design document</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>TI- 3</td>
<td>The symbols used in the sequence diagram correspond to UML standards</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>TI- 4</td>
<td>Sequence diagram matches class diagram</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>TI- 5</td>
<td>All the classes in the OCL model are represented in the class diagram</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>TI- 6</td>
<td>The multiplicities in the OCL model have been depicted in the class diagram</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>TI- 7</td>
<td>All the requirements in the Software Requirements Specification have been covered in the Architecture Design Document</td>
<td>Pass</td>
<td></td>
</tr>
</tbody>
</table>
# Formal Technical Inspection Letter by Snehal Monteiro

## Table A.2 Technical Inspection Checklist from Snehal

<table>
<thead>
<tr>
<th>Item #</th>
<th>Inspection Item</th>
<th>Pass/Fail/Partial</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>TI- 1</td>
<td>All the symbols used in the class diagrams are according to the UML standards</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>TI- 2</td>
<td>All the classes in the class diagram are clear as to what they represent in the architecture design document</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>TI- 3</td>
<td>The symbols used in the sequence diagram correspond to UML standards</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>TI- 4</td>
<td>Sequence diagram matches class diagram</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>TI- 5</td>
<td>All the classes in the OCL model are represented in the class diagram</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>TI- 6</td>
<td>The multiplicities in the OCL model have been depicted in the class diagram</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>TI- 7</td>
<td>All the requirements in the Software Requirements Specification have been covered in the Architecture Design Document</td>
<td>Pass</td>
<td></td>
</tr>
</tbody>
</table>

## Email from the Technical Inspector

Dr. Hsu and committee members,

After carefully inspecting Sowjanya Karumuri’s project reports and design documents, I verify that all the documents and UML Diagrams are complete and consistent with the standards.

Sincerely,

Snehal Monteiro

Graduate Student in Software Engineering