CASE TOOLS

SYSTEM REQUIREMENT SPECIFICATION

Akhil Jain and Shubham Jain

Table of Contents

1. Introduction

1.1. Purpose

The main purpose of this SRS is to describe the software requirements of Computer Aided Software Engineering (CASE) Tools. It includes the functions and features of the software. Moreover it gives an overview of the design implementation of the software.

1.2. Scope

The Computer Aided Software Engineering (CASE) tool allows Software Engineers to create simple unambiguous easily maintainable and reproducible software designs. CASE tools enable software engineers to abstract away from the entanglement of source code, to a level where architecture and design become more apparent and easier to understand and modify. This tool allows automated building of software design and automatically creates the data dictionary.
It helps in automation of various activities of system development and management processes, hence increases productivity of the development team.

1.3. Glossary

CASE :	Computer Aided Software Engineering Tool
User :	Software Project Managers, Analysts and Engineer
SRS :	Software Requirement Specification
API :	Application Programming Interface
IEEE :	Institute of Electrical and Electronic Engineers

1.4. References

[IEEE] The applicable IEEE standards are published in “IEEE Standards Collection”, 2001 edition.

1.5. Overview

The rest of this document contains an overall view of **Computer Aided Software Engineering(CASE)** Tool and the specific requirements of the software.

2. Overall Description

2.1. Product Perspective

The term CASE was originally coined by software company Nastec Corporation of Southfield, Michigan in 1982 with their original integrated graphics and text editor GraphiText, which also was the first microcomputer-based system to use hyperlinks to cross-reference text strings in documents—an early forerunner of today's web page link. GraphiText's successor product, DesignAid, was the first microprocessor-based tool to logically and semantically evaluate software and system design diagrams and build a data dictionary.

This software has borrowed ideas from current CASE tools, including CiteSeerX, Jarzabek Lo and others. The software belongs to category of CASE tools that have gained popularity in the last few years.

CASE tools are designed to enhance and upgrade the computing system adopted and used. This is very important with regards to the dependence on a computer-based environment for business and/or personal pursuits.

2.2. System Environment

CASE tools consist of programming environments like IDE (Integrated Development Environment), in-built modules library and simulation tools. These tools provide comprehensive aid in building software product and include features for simulation and testing.
The software runs on Linux (Ubuntu) and also on 32 and 64-bit Windows 2007, Windows 8,8.1,10.

2.3 Functional Requirements

Structured Analysis:

The CASE tool supports a graphical interface and the following features

- The user can draw bubbles, data stores, and entities and can connect them using data flow arrows.
- Supports editing the data flow diagram.
- The user can create the diagram hierarchically.
- The user can determine balancing errors whenever required.
- The software creates the data dictionary automatically.
- Supports printing the diagram on a variety of printers.

Structured Design:

- The user can draw modules, control arrows, and data flow arrows. Also, a symbol for library modules is provided.
- The user can associate a module with some bubbles of the DFD. It can check if all the bubbles are assigned to some module and also whether each module is assigned at least one bubble.
- The user can modify his design.

- Clicking on a module shows its internal organization.
- The user can save his design and can load previously created designs.

2.4 User Classes and Characteristics

1. **Advanced professional end users**: Experienced software engineers who have knowledge of DFD and can themselves check for errors, balance faults and are familiar with the symbols and notations and the concepts of data dictionary.
2. **Inexperienced professional users:** Beginners in software engineering who intend to familiarise with the concepts of data flow and CASE tool to understand how to build and maintain software.
3. **Science/Research applications:** Maybe used by scientists, research scholars and other academic units to maintain projects, research data and information, and to plan the execution of projects.
4. **System Administrators:** People who wish to design and maintain their own systems using CASE tools.
5. **Industry**: Software industries may use this software in the design and analysis stage of their projects.

2.5 Design and Implementation Constraints

1. Every option, when selected, requires a processing time of 100 milliseconds before being executed.
2. The user is not allowed to create more than one kind of shape at a time.
3. The user can associate a name and details with every entity that he creates.
4. No specific protection in terms of encryption of data is provided.
5. All instructions, operations, diagrams must employ English as the language of usage.
6. A file is created for every diagram so there must be sufficient data on the hard disk/memory.
7. The system must have a mouse or its equivalent installed.
8. All data storage items other than the file are created and destroyed dynamically.

2.6 User Documentation

Given Separately.

2.7 Questions Asked/Answers

Structured Analysis

1. How is the diagram hierarchy created ?

There could be hierarchy within a single diagram or across diagram. For example, relationship between a whole and its parts, between generalization and specializations can be shown within the single diagram in the form of an organogram (n-ary tree). However, multiple levels of detailing could be hierarchically provided across diagrams (each representing an abstract layer over the detailed one). For example, top level gives names of classes with relationships, next level elaborates operations, next to next level illustrates attributes with visibility, etc.

2. How are the errors balanced (what is balanced) ?

In a tree like hierarchy diagram, care should be taken not to make it skewed and to use the real estate properly. Hence the view is balanced, use of real estate is balanced etc.

Structured Design

1. What are the internal organisations of the modules ?

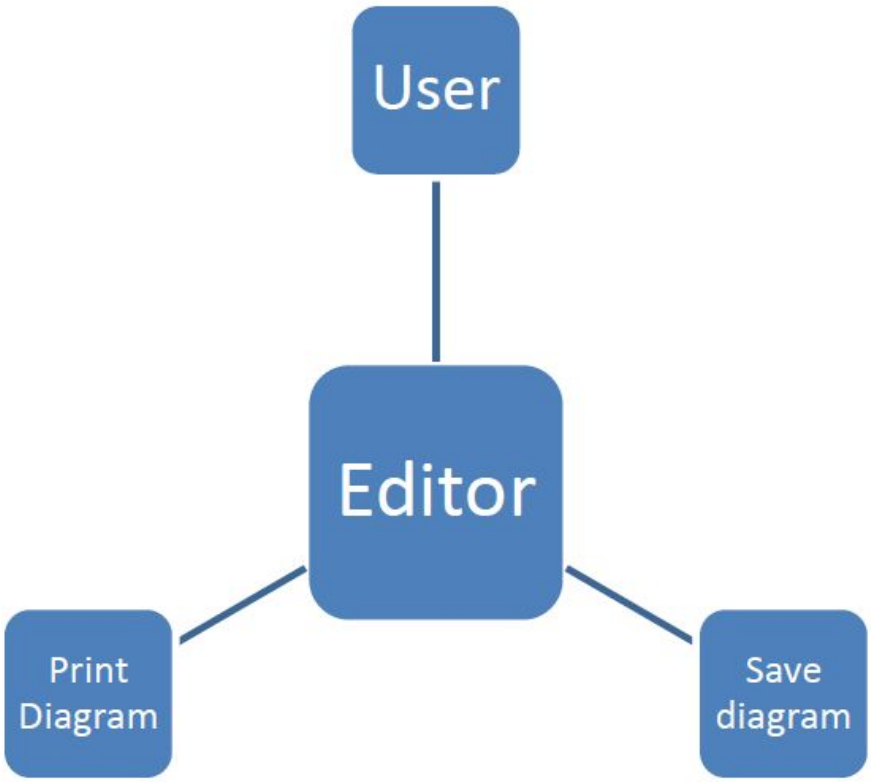
Relationships between sub-modules, linkages to hierarchical details etc.

2.8 Assumptions and Dependencies

The following are certain assumptions:

1. The user has basic knowledge of software design.
2. The CASE Tool does not have any security measures implemented. The user should use the software with caution if security is vital to him/her.
3. The user should know how to check for balancing errors.
4. The preferred operating system is Windows 2007 for best graphics.
5. The system has Java Runtime Environment installed.

3. Requirement Specification



- **Create a New Diagram**

INPUT:

The user gives the name of the diagram

PROCESS:

The CASE Tool creates a new empty diagram with name given by the user

OUTPUT:

An empty diagram with given name is displayed

- **Open a Diagram**

INPUT:

The user gives the name of the diagram to be loaded

PROCESS:

The CASE Tool looks for a previously saved diagram in the system. If it finds the diagram, it displays it

OUTPUT:

A Diagram or an Error Message saying the file doesn't exist

- **Edit Diagram**

INPUT:

The user adds new shapes, delete existing shapes, checks for balancing errors and asks for Data Dictionary.

PROCESS:

The CASE Tool creates the shapes and displays it to the user. It also automatically updates the Data Dictionary and displays it whenever required. It deletes the shapes when the user asks, and again updates the Data Dictionary again automatically. It also processes the diagram for balancing errors.

OUTPUT:

The altered Diagram is displayed, Data Dictionary is displayed and Balancing Error is shown

- **Print Diagram**

INPUT:

The user gives the print command to print the Diagram

PROCESS:

The CASE Tool looks for the printers installed to the system, if it finds one, it prints the Diagram or an Error Message is displayed

OUTPUT:

The Diagram is printed or Error Message is shown

- **Save Diagram**

INPUT:

The user gives the save command to print and name of the file to be saved.

PROCESS:

The CASE Tool saves the Diagram to file if disk space is available or an Error message is shown. It also asks the user if he/she wants to replace the file if the file is already existing.

OUTPUT:

File is saved or an Error Message is displayed

Other Requirements

Operating System:	Windows 7 and above, Linux
Code Standard:	The software will be developed in Java.
Libraries:	JRE (Java Runtime Environment)