SmartTag Design Document

Stephen Zakrewsky
Vikesh Solanki

February 21, 2008

Contents

1 Introduction **2**
1.1 Purpose of this Document 2
1.2 Scope 2
1.3 Architecture Diagram 2
1.4 Context Model 2
1.5 Definitions 3

2 Object Details **4**
2.1 ConfigParser 4
2.2 GetObject 5
2.3 FileHandlerObject 5
2.4 TagObject 6
2.5 AutomaticObject 6

3 Deterministic Protocol **7**

4 Configuration File **7**
4.1 FILENAMEMACRO 8
4.1.1 Default 8
4.2 ACCLEVEL 8

5 Jobs **9**
5.1 Rename File 9
5.2 Fix MP3 tags 9

6 Appendix **9**

List of Figures

1 Design Layout 3
2 Context Diagram 4
3 ConfigParser 4
4 GetObject 5
5 AutomaticObject 6
6 Algorithm to Gather Needed Search Criteria 7
7 Sample Configuration File 8

8 Data Sequence Diagram 10
9 Accuracy Formula 10

1 Introduction

1.1 Purpose of this Document

This document outlines all of the overall design of the SmartTag system as well as the architecture of the system. The design decisions have taken into account the constraints, attributes, interfaces and functionality of the system, all outlined in the Requirements Document. This document will be used, in conjunction with the Requirements Document, to implement this system by the developers. This document should be able to be used as a reference to any developer in the future who will be modifying the system as a guide as to how the system was initially created as well as being a tool for the developers to look into whilst implementing a part of the system. It is also to be used at a later date should the system need modifying.

1.2 Scope

This document contains the design elements involved in creating Version 1.0 of SmartTag. The document was created, and intended, for the the designers, development team and testers of the system.

1.3 Architecture Diagram

Figure 1 shows how the different parts of the system will interact. Basically the users level CLI or GUI will be in charge of creating and handling all of the objects that are needed. The first thing it should do is create a ConfigParser. Then for every file that it needs to handle it should create a FileHandler-Object by calling ConfigParser::parseFilename(). Then it should create a TagObject to edit tags and a AutomaticObject to do it automatically. See the corresponding figures 3, 4, 5.

1.4 Context Model

The process of searching for and correcting tags is shown in figure 2. The user will type out their commands through the CLI, defined in the Requirements

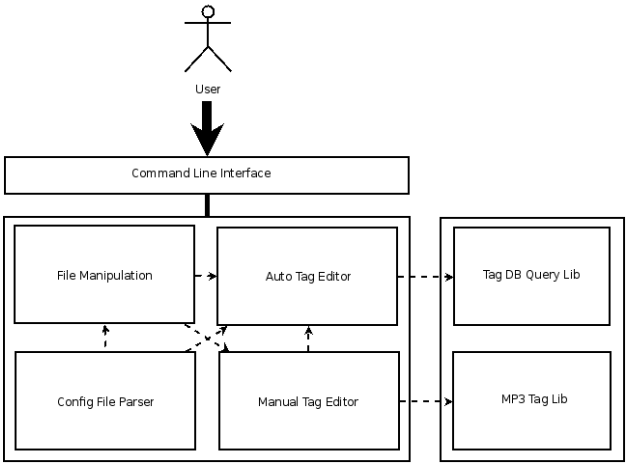

Figure 1: Design Layout

Document, and then the host computer will contact their web server, ensuring network connectivity is available, and then the search fields are sent to the external database. The results are returned and the tags are corrected. The data sequence diagram, figure 8 in the section 6, shows this process against time, and shows which process is handled by which part of the system (user, computer, server or external database).

1.5 Definitions

class A package of metadata that groups related fields and methods.

database A collection of data stored in a computer system or network

Internet The Internet is a collection of interconnected computer networks that are publicly accessible.

mp3 Currently the most popular digital audio format.

Perl A general purpose programming language used for text manipulation, system administration, creating GUIs and web development.

ripping The process of copying audio or video content from a removable medium (or media stream) onto a hard disk.

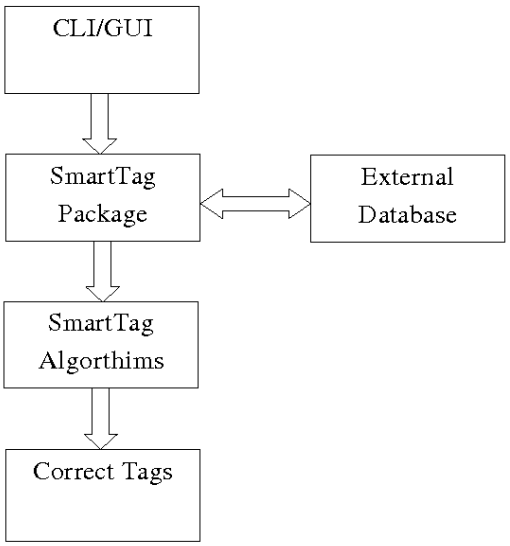

Figure 2: Context Diagram

system administrator The people in charge of, and designing updates of the system.

user The person taking advantage of the systems capabilities.

2 Object Details

2.1 ConfigParser

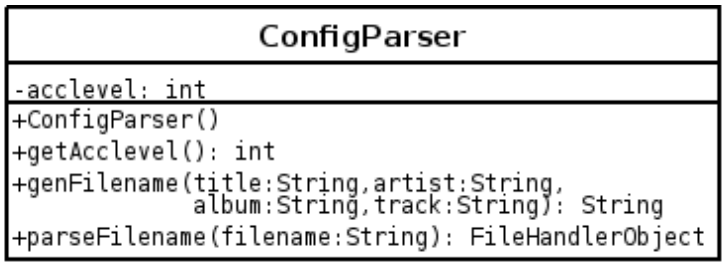

Figure 3: ConfigParser

This Object is in charge of the configuration file. It can create a filename following the FILENAMEMACRO defined in the configuration file. It can also return the ACCLEVEL or the elements defined by the FILENAMEMACRO. It it safe to be used by any classes that need it because it can not change data. There should be one ConfigParser created when the program is run, and it should exist until the program is finished.

2.2 GetObject

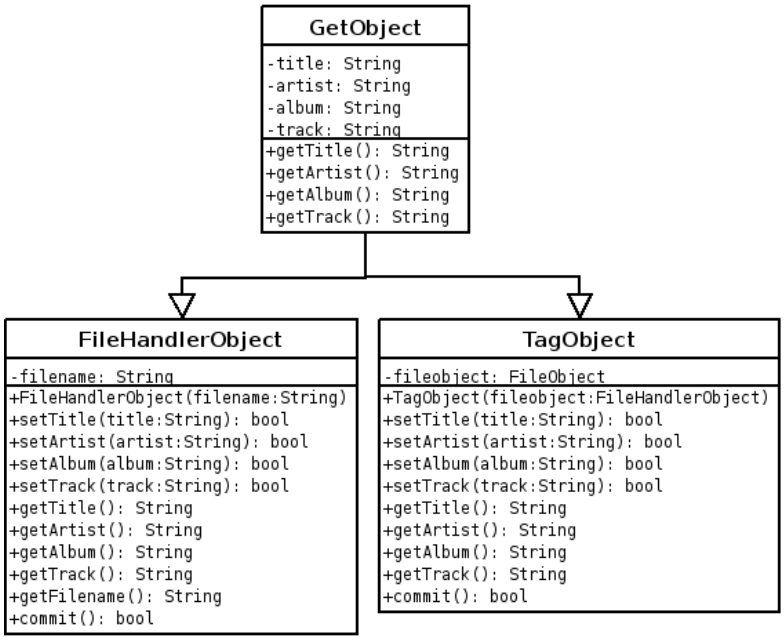

Figure 4: GetObject

GetObject is just an abstract class to access the get methods of the FileHandlerObject and the TagObject. Because it allows access to only get methods it is safe for everybody to use.

2.3 FileHandlerObject

The FileHandlerObject is in charge of getting and changing the filename of an mp3 file. The elements of the file defined by FILENAMEMACRO should

be already instantiated by the call to ConfigParser::parseFilename(). This is how FileHandlerObjects should be created. A call to the FileHandlerObject's commit method will commit any changes and destroy the object. It is important to destroy objects using the highest level commit. For example, if a FileHandlerObject is created and then passed to a TagObject then the TagObject's commit should be called not the FileHandler's commit.

2.4 TagObject

The TagObject is in charge of getting and changing the mp3 tags of an mp3 file. A call to the TagObject's commit method will commit any changes and destroy the object. It is important to destroy objects using the highest level commit. For example, if a FileHandlerObject is created and then passed to a TagObject then the TagObject's commit should be called not the FileHandler's commit.

2.5 AutomaticObject

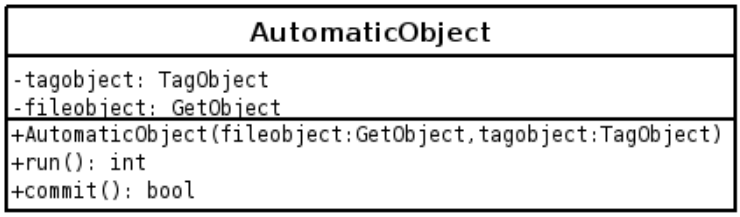

Figure 5: AutomaticObject

The AutomaticObject in in charge of implementing the algorithms to determine the correct tag information automatically. These algorithms are shown in figures 6 in section 2.5 and 9 in section 6. This will be done when the run method is called. The program should check the return value every time the run method is called. It should then decide to commit or change values and rerun. The returned value will be an integer specifying errors. Those errors are:

0 Success

1 No Suitable Title

2 No Suitable Artist

3 No Suitable Match

4 No Search Results

5 Network Error

3 Deterministic Protocol

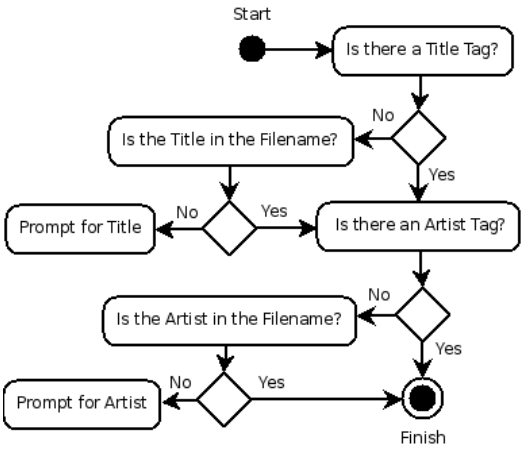

Figure 6: Algorithm to Gather Needed Search Criteria

SmartTag will use an algorithm to determine how to gather the title and artist information which is needed to search for MP3 tag information. This algorithm is shown in figure 6. Data collected will be kept in order to determine how accurate the searched MP3 tag information is. Once a query is complete a list of matched albums will be returned. SmartTag will use another deterministic algorithm to decide if it has found a suitable match. That algorithm is shown in figure 9. A count is kept and compared to a user defined value ACCLEVEL in the configuration file. See figure 7.

4 Configuration File

SmartTag will use a user configuration file for a few personal choice options. This file will be created with the defaults when SmartTag is run for the first

```
# This is the Default User Configuration File for SmartSet
# It was automatically generated on 2/03/08-13-18
# See man smartset for documentation.
#

# Macro used to parse filenames.
# Tokens:
#       $ARTIST$
#       $TITLE$
#       $TRACK$
#       $ALBUM$

FILENAMEMACRO=$ARTIST$-$TITLE$
ACCLEVEL=5
```

Figure 7: Sample Configuration File

time. See figure 7.

4.1 FILENAMEMACRO

FILENAMEMACRO is a string that SmartTag will parse for key tokens in order to gather information from a user's filenames.

Tokens will include:

$ARTIST$

$ALBUM$

$TITLE$

$TRACK$

The macro will be a configurable option via a user configuration file. The ".mp3" filename extension should not be included in FILENAMEMACRO. See figure 7.

4.1.1 Default

By default SmartTag will use a FILENAMEMACRO = $ARTIST$-$TITLE$.

4.2 ACCLEVEL

This is the accuracy level, a number of 0, 5, or 10. This is used to adjust how sensitive SmartTag is when it comes to selecting the queried data to change files.

0 This level will essentially pick the first element that is returned from the search query. The file need not have an album tag or a track length.

5 This is the default. This will work with out an album tag or a track length, but there must be one if not both.

10 This level will only match if an album tag is specified and both the album tag and track length tag match a returned value.

5 Jobs

5.1 Rename File

SmartTag will rename a user's files using the correct information. It will follow the guidelines provided by the FILENAMEMACRO. See section 4.

5.2 Fix MP3 tags

SmartTag will also fix all of the important MP3 tags. Those tags include the artist, title, track, and album. SmartTag will also add a comment tag of "SmartTag-version-number" for future reference.

6 Appendix

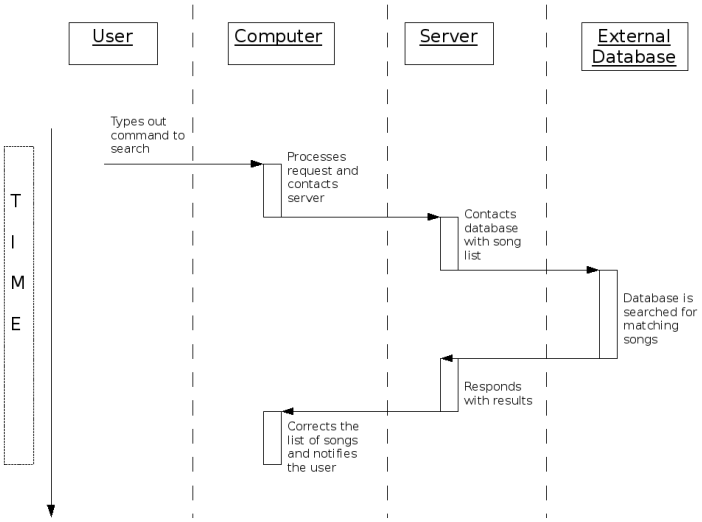

Figure 8: Data Sequence Diagram

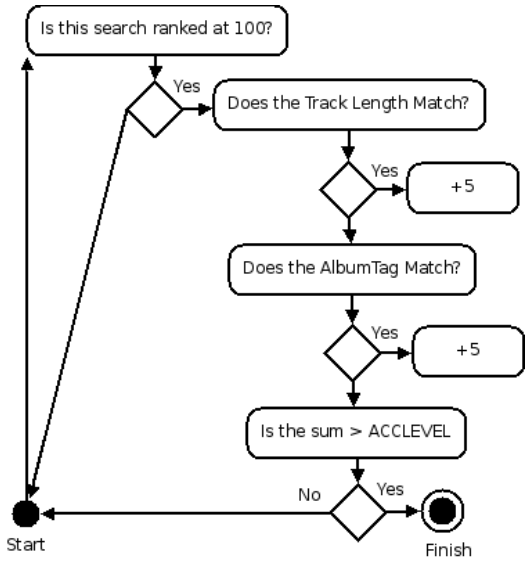

Figure 9: Accuracy Formula