SmartTag Requirements Document

Stephen Zakrewsky
Vikesh Solanki

March 15, 2008

Contents

List of Figures

1 Introduction

1.1 Objective

The system we are creating is a tool that edits and repairs tags for a person's MP3 collection. This manual details the system requirements for the SmartTag MP3 tagging system. The system aids the user in repairing their database of audio files by fixing each file's ID3 tags. This is achieved by filling in the incomplete information and correcting any mistakes in the current tags. If time allows us, we will provide the user with the option of attaining all details for audio files that they have ripped off a CD which contains no existing information. These standards are going to be achieved by the system, while concentrating on being flexible, easy to use, quick and accurate.

1.2 Background

Currently, the MP3 format is the most popular method of storing and downloading digital audio. Ever since 1999 people have been ripping music from CDs to their computers and listening to them in MP3 format. After the rise in popularity, electronics companies decided to release dedicated portable MP3 players. Since the rise of these players and the increase in technology, there has been a huge spike in the amount of downloads of audio in this format. The piece of data which store all of the audio details is known as the ID3 tag. This contains details such as the artist name, the album title, and the song name among other details. There are two versions of the ID3 tag: ID3v1 and ID3v2, with the latter being the latest and most widely used. The ID3v2 tag is a file of variable size that occurs at the start of the file but is not audible on the MP3. This format is in constant development to make it better and therefore it is a constantly evolving format.

2 Functional Requirements

2.1 Command Line Requirements

2.1.1 Filename

The user can run SmartTag on one or more filenames specified on the command line.

2.1.2 Directory

The user can run SmartTag for a directory specified on the command line. Only one directory should be accepted. SmartTag will only run on files in the first level of the directory. It will not follow any subsequent directories.

2.1.3 Default

The default is to run SmartTag on the users current directory. The users will be prompted to confirm before SmartTag will run, and a list of all files that will be edited will be shown. See Fig 1. Any errors in file access will show an error but will not halt the program in the event that some other files are accessible. SmartTag will prompt the user for input if ey information is missing. See Fig 2.

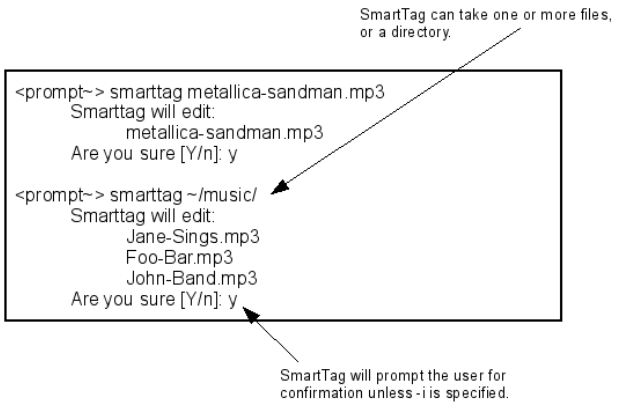

Figure 1: Running SmartSet

2.1.4 Options

2.1.4.1 -h –help If SmartTag is run with this flag a help message is shown.

2.1.4.2 -i –nointeractive This flag will not prompt for user input. This is useful if the user wants to run SmartTag for a while and leave the computer. They wouldn't want SmartTag to block after only three songs waiting for user input. Any songs that would have needed user input will be added to a list. The list will be printed on finish.

2.1.4.3 -v –verbose This flag will print extra debug messages. This is useful if the user wants to see every step that SmartTag is taking.

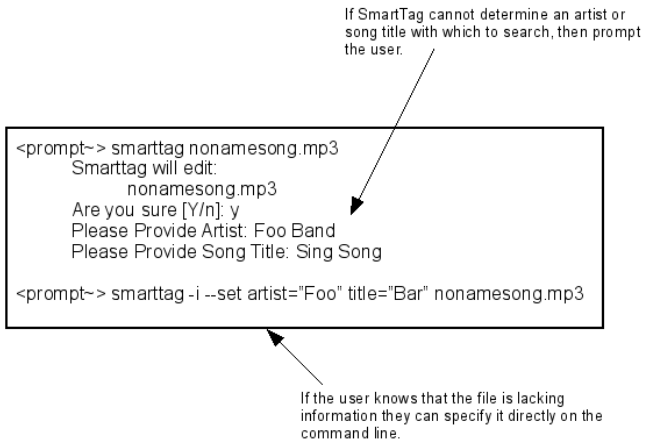

Figure 2: MP3 Tag User Input

2.1.4.4 -s –set This flag is used to set MP3 tags by hand. This will set the tags for all MP3s provided on the command line, or directory. It will still do an automatic search for the correct tags. This is useful if the user knows that certain MP3s will fail because they are missing tags. The user can specify those tags at the start of SmartTag . See Fig 2. The following can be specified with this flag:

artist= sets artist tag

album= sets album tag

title= sets track title tag

track= sets track number tag

noauto do not automatically look up tag information.

2.2 File Requirements

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. See Section 2.4. The ".mp3" filename extension should not be included in FILENAMEMACRO. See Fig 3.

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

2.3 Results Requirements

Rename File SmartTag will rename a user's files using the correct information. It will follow the guidelines provided by the FILENAMEMACRO. See Section 2.4.

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

2.4 Configuration File Requirements

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 time. See Fig 3.

```
# 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 3: Sample Configuration File

FILENAMEMACRO= This is used to tell SmartTag how the user would like their filenames to look, and more importantly, how SmartTag should parse the filename. See 2.2 for the default.

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.

3 Non-Functional Requirements

3.1 System Constraints

Database Constraints The constraints regarding the database are inherited from the fact that rather than using our own custom-made database, we will initially be relying on an online database. This will cause the system to not be as quick as it possibly could be as it will have to search this online database. It also means that the system will rely on being connected to the Internet. If it is not connected to the Internet then the system will not be able to access the database.

Depth Constraints The dilemma with a database containing information on the music industry is that it is almost impossible to have data for every CD or song has ever been recorded and released. This is the main reason an online database was used rather than creating and using our own database. Even then, it is a never-ending task to have details on every song ever produced. This results in the system never being certain to be completely full or completely accurate.

3.2 Development Process Constraints

Within the development process there are also limitations which cause the system not be as complete as possible. These restrictions are time and experience.

Time Constraints Time will play a role in our development stages as we do not have a huge amount of it. This results in us not being able to compromise our own database or a GUI for the system. This results in the system not being as user-friendly as we would have liked and also results in a slower system. We would have liked time to create a GUI so that it ensures the system is easier to use and also makes sure that the user doesn't require knowledge of how to use the CLI.

3.3 Experience Constraints

Our lack of experience in creating and designing real-world systems is a massive hindrance to us. If we had developed systems for external stakeholders

before, we would have had a huge advantage. Also, the extent of our programming knowledge is not too large either. Combined we have a good knowledge of only the languages C, C++, Java, Perl and VHDL.

4 System Evolution

As this is only a six week project, we are unable to create a full system. This system will be expandable for the stakeholder by numerous methods. These methods will add additional features that will help boost the speed of the system as well as aiding the ease of use of the system. The main method will be for them to have their own database rather than rely on an online database. This will result in a quicker system but will take longer to produce and take a while for it to be completely accurate as it will require the stakeholder filling in details for millions of albums.

4.1 Local Database

This addition would ensure that the system would be quicker at returning results. This would be accomplished as the system would not have to be connected to the Internet to retrieve the search results. The only time the system has to connect to the Internet is to update the database. This also enables the stakeholder to have complete control over their system. This would please them immensely as they wouldn't have to rely on another, possible unreliable, service.

4.2 Create A GUI

We would like to add a proper GUI to the system. This would ensure that the usability of the system is increased immensely. This would be due to the increased ease-of-use of the system. This therefore enables the system to be used by a wider range of end-users as the end-user would not have to need knowledge of using the CLI. This would tie in well with a custom made database but would also take a while to implement.

4.3 Edit Multiple Directories

In future editions of the system, we would like to introduce the ability to edit multiple directories - A feature that makes the system even easier for the user to use. Even though this functionality would be possible to program now, the usefulness is outweighed by the overhead of the work involved, for example, running SmartTag over an entire file system.