Controlling Sound Parameters Using Natural Language Expressions

Jan-Torsten Milde
Fulda University of Applied Sciences, Digital Media Working Group, CS Department
milde@hs-fulda.de

ABSTRACT

This paper describes the ongoing development of a system, that allows to specify sound parameters of a virtual synthesizer using natural language. A sound generated may be described by the user with complex adjective phrases. These instructions are parsed and a semantic representation is created, which in turn is mapped to control parameters of the software synthesizer. The system system is the basis for more elaborate investigation on the relation between sound quality specifications and natural language expressions.

1. INTRODUCTION

In this research we like to investigate on the problem of describing sound quality using natural language expressions. Specifically we are interested how complex adjective phrases can be used to specify a sound. It is currently not clear, if this is possible at all. No satisfactory empirical data has been collected so far. Therefore we have started to set up a test system, which allows us to perform interactive user experiments. These user tests have been carried out during the last three months and we are still in the process of analyzing the result.

It can be observed, that musically illiterate people tend to describe sounds and music using every day language. Without detailed knowledge of musical terminology or even know how of the internal functions of a modern synthesizer (see figure 1), a large variety of *adjectives* is used to describe the sound quality instead.

A common example would be the use of the adjective *fat*. Sounds have to be fat, the bass line of a modern dance track has to be fat. If a specific song gets a mutual positive evaluation within a peer group, it will be asigned the predicate *voll fett*. The adjective *fat* has also been adopted by the marketing departments of the music industry. One of the nicest example is the string synthesizer *Streichfett* by Waldorf [1].

There seems to be a common understanding, whether a sound is fat or not. Nevertheless, when it comes to generating a sound using a standard synthesizer, there is no knob controlling the *fatness* of a sound. When designing a bass sound, fatness is sometime attributed to minimal changes of the pitch of a secondary oscillator.

2. IDENTIFYING SOUND IN CURRENT SYSTEMS

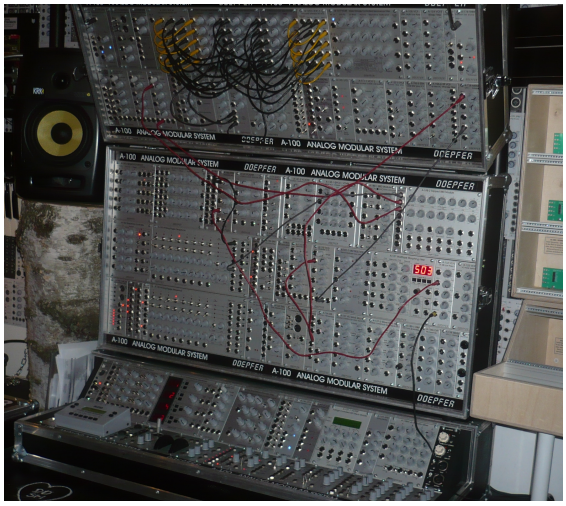

Figure 1. A modular analog synthesizer by Doepfer presented on this year's Musikmesse in Frankfurt. The impressive interface represents the internal functional complexity of the system, inviting the user to experiment with the system. Relating a certain system configuration to a specific sound or vice verse is hard.

In current systems two main approaches for identifying sounds can be distingushed: naming conventions and hierarchical meta data.

Naming conventions basically associate an arbitrary string with a certain system configuration and thus with the sound produced. In order to find the sound, the user has to enter (part of) the name string. This restricts the selection of possible sounds. As a result, the user is presented a filtered list of possible hits which then can be loaded by the system and eventually be tested by user. Figure 2 shows a standard interface with a typical input mask. Here the search is performed incrementally, thus restricting the number of possible sounds as the users types letter by letter [2].

The second approach for identifying sound quality is the use of *hierarchical meta data*. Here sound describing terms are organized in a hierarchy. General categories are recursively refined by sub categories. The MediaBay uses

[1] At least within the German speaking community the name often rises the question, whether Waldorf is really serious about this choice.

[2] The presented search interface (MediaBay) is part of the Cubase software and is connected to the halion5 system. Also it has to noted that the halion5 system is a software sampler, not a software synthesizer.

Figure 2. Sound search of the halion5 software synthesizer. The user has to type in sound names into a very tiny input field. Sounds are listed in a tabular view and can be ordered according to different column categories.

four category levels: category and subcategory, style and character. The category is organized around classic instrumental groups found in orchestras and pop and rock bands. Style and character refer to different types of music, with the finest category finally trying to somehow describe the *character* of the sound. While in the first three levels nominal phrases are used to set up distinct classes, in this category we find adjectives to capture the mood or the expression of the sound. Figure 3 shows an example of a search hierarchy.

Both approaches are easy to understand and effective to organize large quantities of sounds, but still make it very hard to find a specific sound. An obvious problem are sound names. These names are almost completely arbitrary. Sounds are described either metaphorically or by the underlying technology or by specific music genre and in many other ways. A good example is the guitar sound named "Brian Will Rock You", which obviously refers to a sound comparable to the one found in the song "we will rock you" by Queen with the guitar played by Brian May. How would one ever come up with idea to search for "Brian" ?

The meta hierarchies are also problematic. Again category names are arbitrary. This is not so much of a problem, as the number of terms is usually quite small. On the other hand categories are used to clearly discriminate the sounds. Either a sound is in a category or it is not. When you are trying to find that "fat bass" with the category system, then you might be faced with fact, that you simply do not know, which category the sound is in: it could be an analog upright bass, a monophonic bass using FM synthesis or a filtered brass sound.

As a result both approaches force you to painfully step through thousands of sounds on your hard disk desperately seeking for the right one for your current arrangement. Even worse, you have used the sound before, but just cannot remember the name or the category.

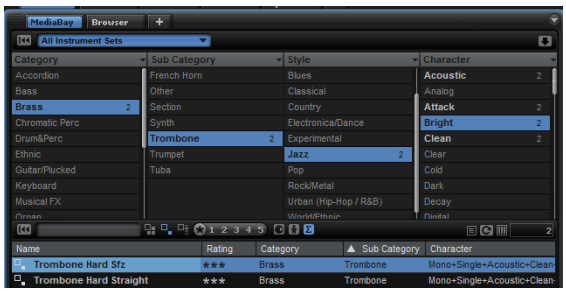

Figure 3. Hierarchical sound search for the halion5 software synthesizer. Categories are presented in a tabular way, starting with highest hierarchical level in the first column. For each level a list of possible values can be selected by the user. The hierarchy filters the result list by performing a logical AND operation.

3. THE ARCHITECTURE OF THE SOFTWARE SYNTHESIZER

In order to perform interactive user tests we developed a software synthesizer that incorporates both a standard control interface and a natural language input system, allowing to describe the sound using complex adjective phrases.

The test system is based on a (simple) standard synthesizer that has been implemented using the *Processing* programming environment (see [1], [2]). This environment is specifically suited for the development of visually attractive, interactive systems. It is often used within the field of computer based generative art and also offers a large number of extension libraries. These include a simple to use audio library (*Minim*), allowing to easily setup a modular synthesizer consisting of the standard components: oscillators, filters, amplifiers.

For the test system we developed a monophonic synthesizer. The system consists of a DCO generating five variations of waveforms (sine, square, saw, triangle and noise). The pulse width of the square waveform can be adjusted. In addition it is possible to create a waveform by summing the first n harmonics with random weights.

The signal is routed into the DCF, a simulation of a 24dB Moog resonance filter. The filter provides three filter characteristics: a low pass filter, a high pass filter and and a band pass filter. In addition, the cutoff frequency can be set and the resonance strength can be specified.

Once the signal has been filtered it is sent to the DCA. As usual, this step allows to control the loudness progression of the final signal.

The DCF and the DCA are each being further controlled by a four step ADSR envelope which is triggered by note-on/note-off events. Two more LFO have been incorporated into the system. One LFO modulates the DCO in order to realize a vibrato, the second LFO is used to further modulate the frequency of the DCO, the cutoff frequency of the DCF and the gain of the DCA [3] .

The final output of the oscillator-filter-amplifier chain is sent to an effect unit consisting of a simple delay. Here

[3] To be precise here: the gain is controlled independent from the DCA in gain module chained behind the DCA.

the delay time and the amplification factor can be set. The complete system architecture is displayed in figure 4.

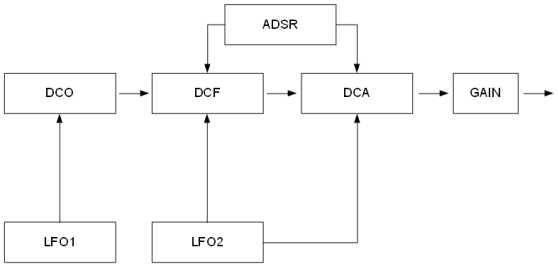

Figure 4. The system architecture of the synthesizer system. The architecture follows a standard configuration found in many systems: the oscillator generates the basic wave form, which is filtered by a DCF. A subsequent DCA controls the volume. In addition 2 LFOs and an ADSR further control volume and modulation of the generated sound.

We tried to design the user interface of the test system according to current industry standards. Almost every physical system on the market is using dedicated hardware controls (knobs, sliders, buttons etc.) to interact with the central internal parameters of the sound generating system. This approach has been adopted by software synthesizers: the majority of the virtual instruments displays a user interface which copies the look and feel of a r̈ealsÿstem. As our test system has a relatively simple internal structure, the complexity of the control interface is not very high. Nevertheless it provides all the elements, that are found in bigger systems. We think, that we have found a good level of abstraction for the test system. It is versatile enough to generate an interesting variety of sounds, still simple enough to be explained to the test participants.

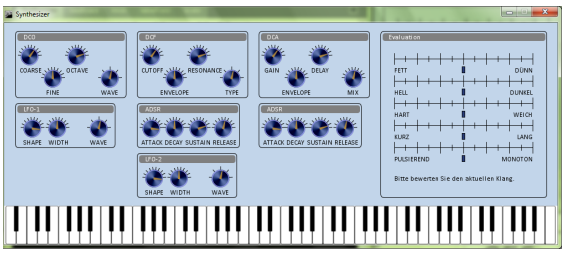

Figure 5. The interface of the test system. The graphical user interface mimics the physical appearance of a simple analog synthesizer, thus creating compatibility with standard commercial systems. The evaluation unit is used to collect user data during the naming process.

4. SOUND AND LANGUAGE

As a starting point for the development of the natural language interface, we generated a list of adjectives (see table 1) based on our own experience in describing sounds. This list has not been empirically checked. It is solely based on personal preferences. While this approach might be questionable, we found that the number of adjectives quickly rises and becomes quite diverse, when taken from user interviews. This result indicates that it might not be feasible to allow unrestricted use of every day language for the sound description. Instead controlled language (see [3]) could be used, reducing the vocabulary to a predefined subset.

System parameter	*Adjective*	*Antonomy*
volume	laut	leise
filtering, waveform	hell	dunkel
filtering, waveform	schrill	dumpf
modulation, vibrato	pulsierend	monoton
ADSR, waveform	hart	weich
modulation, filtering	rhytmisch	melodis
ADSR, delay	kurz	lang
modulation	ansteigend	abfallend

Table 1. A subset of the list of basic adjective pairs. The table lists adjectives with their semantic counterparts (antonym) and shows how they could be related to sound parameters of the test system.

Once the basic set of adjectives had been chosen, we tried to expand the possible vocabulary. This was achieved by using the GermaNet system (see [4]). GermaNet is the German WordNet version. In this data base more then 100000 German lexical items are stored with their conceptual and lexical relations encoded. With this system it becomes quite simple to find adjectives with related meaning (synonyms) and adjectives with opposite meaning (antonyms) . Using this approach, we were able to extend the basic list of descriptive adjectives to more than 150.

4.1 NLP analysis and generation

The standard user interface of the test systems has been extended by incorporating a natural language input field. Using this field, the user is able to enter complex adjective phrases (currently in German). These phrases are parsed by the underlying NLP (*Natural Language Processing*) system and a (complex) attribute value matrix is set up. This matrix is used as an internal semantic representation of the user's sound description and will be mapped onto the sound parameters.

The underlying parser is using a lexicon driven approach. The lexicon is a list of words, which are associated with certain attribute value pairs, describing their semantic value. In our case semantic value refers to sound quality parameters. These parameters will later be mapped onto system parameters.

We are providing a very loose phrase structure grammar for the sound descriptions. Basically a user is allowed to enter any combination of adjectives followed by an optional category name. This combination is referred to as an AP (*Adjective Phrase*). It is possible to build more complex instructions by connecting multiple APs with conjunction (*and*) or disjunctions (*or*). Finally, adjectives can be further differentiated by modifiers. These could be negations (*not*) or gradations (*more, less*). So far we do not process the adjectives themselves. Comparative or superlative

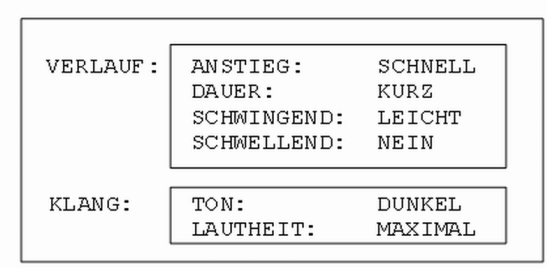

Figure 6. Attribute Value Matrix. In this example information of different parts of the user's sound description has been collected. The unification process integrates the parts, joins compatible information and thus fills the slots of the AVM.

word forms have to be stored in the lexicon as a discrete lexeme. The simplified underlying grammar could be described by the following phrase structure rules:

```
S -> AP*
AP -> AP CONJ AP
AP -> ADJS N
AP -> ADJS
ADJS -> ADJS, ADJS
ADJS -> MOD ADJ
ADJS -> ADJ

ADJ -> lexicon_lookup
N -> lexicon_lookup
CONJ -> lexicon_lookup
MOD -> lexocon_lookup
```

The lexicon lookup process will collect the attribute-value data of the given adjective, noun or modifier. These partial attribute value matrices (AVM) are the combined by the system to a single big AVM. As the constituent structure created by the parser is relatively shallow, we mostly ignore it when unifying the AVM. An exception to this rule are the modifiers. These have to be combined with the adjectives they modify, hence the phrase structure is used here to guide the unification process (see [5]).

4.2 Relating to sound parameters

Once the AVM has been constructed the final step of actually modifying the sound parameters of the synthesizer has to be performed. Again the mapping between these two structures could be freely negotiated. In order to get a working system configuration, we have used a set of 50 basic sounds in 7 different categories. We have tried to find appropriate description using the system's grammar and have then mapped the resulting AVM to the actual parameters of the system.

The AVM stores the information in a set of categories (see table 2). These categories try to generalize the information of multiple adjectives thus integrating the intended effect of the user's sound description.

AVM	*Sound parameter*
LAUTHEIT:	volume
ANSTIEG:	percussive
TON:	harmonics
DAUER:	duration
SCHWINGEND:	modulation
TON:	feedback
SCHWINGEND: + TON:	vibrato
SCHWELLEND:	tremolo

Table 2. Mapping between the semantic categories and the sound parameters of the test system. The categories are created as the result of the analysis of the natural language expression. Currently the categories take discrete values.

4.3 User tests

During the last three months a set of user tests had been conducted. In order to elicitate the data, two consecutive tests had to be taken by each of the participants.

1. The participants have been asked to describe the sound quality of a number of predefined sounds.

2. The participants have been asked to create a sound, following a a given description.

A total of 25 participants (aged 21 to 27) have taken the tests, 15 male students, and 12 female students. The participants were introduced to the software synthesizer and were able to experiment with the system for 60 minutes. Then each contestant was asked to fill out a questionaire that was used to collect the relevant personal meta data. Finally the general hearing ability of the contestants was checked, specifically the perceived pitch range of each person had been measured.

After that a 10 minute break in a silent environment took place. This was followed by the first test. Here a set of 30 different sound was presented to the participant. Each sound had to be described with a set of predefined adjectives. In order to estimate the influence of each adjective, the participants were asked to grade the adjective with the build in user interface of the software synthesizer. In the second phase of the test, the participants were asked to describe the differences between two sounds. They were able to switch back and forth between the sounds and had to describe the differences, again with a graded set of predefined adjectives.

The second experiment took place after two weeks. A set of 15 descriptions of the first experiments had been selected for each participants. Five of those descriptions were taken from the participant's data set. The remaining 10 descriptions were a set of randomly chosen descriptions presented to all of the participants of the second experiment. The student were asked to recreate a sound that resembled that description.

The complete data set has been stored in an XML annotated data base. We are currently preprocessing the data in order to analyze and visualize the results.

5. CONCLUSIONS

In this paper we describe the development of a test system for specifying sound parameters using natural language descriptions. The test system is fully functional: the software synthesizer allows for the dynamic creation of a wide range of different sounds. The graphical user interface follows current industry standards and makes it simple to modify all the sound relevant parameters. By integrating a natural language parser, sound description can be entered, resulting in parameter changes of the system. A set of user tests have been carried out. The data has still to be analyzed.

With the system a very good experimental basis has been created to further investigate the relation of natural language descriptions and sound parameters. If sound description could be reliably mapped to sound configurations then more elaborate multi modal user interfaces for virtual instruments could be developed.

Acknowledgments

This research has been funded by the internal research commission of the Fulda University of Applied Sciences.

6. REFERENCES

[1] C. Reas and B. Fry, *Processing: A Programming Handbook for Visual Designers and Artists*. The Mit Press, 2007.

[2] M. Vail, *The Synthesizer: A Comprehensive Guide To Understanding, Programming, Playing, And Recording The Ultimate Electronic Music Instrument*. Oxford University Press, 2014.

[3] T. Kuhn, "A survey and classification of controlled natural languages," *Computational Linguistics*, vol. 40, no. 1, pp. 121–170, March 2014. [Online]. Available: http://www.mitpressjournals.org/doi/abs/10.1162/COLI_a_00168

[4] C. Kunze and L. Lemnitzer, "Germanet - representation, visualization, application." in *Proc. LREC 2002, main conference, Vol V*, Gran Canaria, 2002, pp. 1485–1491.

[5] S. Müller, *Head-Driven Phrase Structure Grammar: Eine Einführung*, 3rd ed., ser. Stauffenburg Einführungen. Tübingen: Stauffenburg Verlag, 2013, no. 17. [Online]. Available: http://hpsg.fu-berlin.de/~stefan/Pub/hpsg-lehrbuch.html