



universität
wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis
**„Multi-modal Sonification of Business
Process Event Logs“**

verfasst von / submitted by
Felix Amerbauer BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of
Diplom-Ingenieur (Dipl.-Ing.)

Wien, 2017 / Vienna 2017

Studienkennzahl lt. Studienblatt /
degree programme code as it appears on
the student record sheet:

A 066 926

Studienrichtung lt. Studienblatt /
degree programme as it appears on
the student record sheet:

Masterstudium Wirtschaftsinformatik

Betreut von / Supervisor:

Univ.-Prof. Dipl.-Math. Dr. Stefanie Rinderle-Ma

Declaration of Authorship

I, Felix Amerbauer, BSc, declare that this thesis titled, “Multi-modal Sonification of Business Process Event Logs” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

Acknowledgments

I'd like to thank Tobias Hildebrandt for his continuous support and guidance throughout creating this thesis. Thanks to Stefanie Rinderle-Ma for regular check-ups and administrative help.

My girlfriend, Barbara deserves my gratefulness for keeping me on track even when this required putting aside her own interests.

To my family I'm thankful for their support during my studies, I wished we could celebrate this moment all together.

Table of Contents

1	Motivation	13
1.1	Research Question	14
1.2	Contribution	15
1.3	Structure of Thesis	16
2	Concepts and Related Work	17
2.1	Business Process Management	17
2.1.1	Event Logs	18
2.1.2	Process Mining	19
2.1.3	Process Analysis	20
2.2	Sonification	21
2.2.1	Audification	22
2.2.2	Auditory Icons	23
2.2.3	Earcons	23
2.2.4	Parameter Mapping Sonification	24
2.2.5	Model-Based Sonification	24
2.3	Existing Sonification Tools and Environments	25
2.3.1	Sonification Sandbox	25
2.3.2	SuperCollider	28
2.3.3	xSonify	29
2.4	Visualization	30
2.5	Multimodal Approaches	31
2.6	Related Work	32
2.6.1	ARKola	32
2.6.2	Grooving Factory	32
2.6.3	Business Process Sonification	32
2.7	ProM	33
2.7.1	ProM integration	34
2.7.2	Visualization Support	35
2.7.3	Sonification Support	36
3	Design Options	37
3.1	Event Logs	37
3.1.1	XES Standard Extensions	37
3.1.2	XES Custom Extensions	39
3.2	Sonification	39
3.2.1	Techniques	39
3.2.2	Technologies	40
3.2.3	Mapping	42
3.3	Visualization	42
3.3.1	Options	42
3.3.2	Technologies	43
3.3.3	Mapping	44

3.4	Interaction Design	44
4	Tool	45
4.1	Event Logs	45
4.2	Sonification	46
4.2.1	Techniques	46
4.2.2	Technologies	47
4.2.3	Sonification Structure	48
4.2.4	Default and Custom Mappings	51
4.2.5	Complete Sonification Example	55
4.3	Visualization	58
4.4	Sonification and Visualization Overview	60
4.5	Interaction Design	61
4.6	Software Architecture and Implementation	62
4.7	User Interface	66
4.7.1	Overview	66
4.7.2	Visualization	67
4.7.3	Sonification Mapping	68
4.7.4	Player	73
5	Use-Cases	75
5.1	Process Mining	75
5.2	Process Analysis	84
5.2.1	Repair example	84
5.2.2	High Frequency Logs	89
6	Conclusion	97
6.1	Future work	99
6.1.1	Sonification	99
6.1.2	Visualization	100
6.1.3	Interaction Design	100
6.1.4	Software Architecture / ProM Integration	101
A	Resources	105
A.1	Sonification Sound Effect	105
A.2	Sonification Default Instruments	105
A.3	Sonification Default Melodies	105
A.4	Sonification Default Drum Instruments	106
A.5	Sonification Default Rhythms	107
B	Use Cases: Log Files and Sonifications	108
B.1	Process Mining	108
B.1.1	Invalid Log - Not well formed	108
B.1.2	Invalid Log - No time stamp for all events	108
B.1.3	Repair example log	109
B.1.4	Repair example cleaned log	109
B.1.5	Sonification Control-Flow Perspective	109
B.1.6	Sonification Case-Related Information	113
B.1.7	Sonification Organizational-Related Information	114
B.2	Process Analysis	117

B.2.1	Low Frequency - Repair example Log	117
B.2.2	Low Frequency Sonification Case 1	117
B.2.3	Low Frequency Sonification Case 2	119
B.2.4	High Frequency Log	121
B.2.5	High Frequency Sonification No Anomaly	121
B.2.6	High Frequency Sonification Case 1	122
B.2.7	High Frequency Sonification Case 2	123
B.2.8	High Frequency Sonification Case 3	125
C	ProM Plugins	126
D	Source Code	130
D.1	Standalone Variant	130
D.2	ProM Package	130
E	Abstract	131
E.1	English	131
E.2	German	132
F	Summary - German	132

List of Figures

1	Business process lifecycle [45, p.12]	18
2	XES core and extensions	19
3	Sonification Sandbox mappings	26
4	Sonification Sandbox context time	27
5	Sonification Sandbox context value	28
6	xSonify UI	30
7	ProM AlphaMiner visualization using the <i>repairExample.xes</i> from http://www.promtools.org/prom6/downloads/example-logs.zip ...	35
8	ProM Dotted Chart visualization using the <i>repairExample.xes</i> from http://www.promtools.org/prom6/downloads/example-logs.zip ...	36
9	Sonification Structure	50
10	Score of sonification example	58
11	Visualization colors and shapes	59
12	ProM version	65
13	UI overview	66
14	Visualization	67
15	Properties	68
16	Global mapping	68
17	Melody mapping	69
18	Rhythm mapping	69
19	Instrument mapping	69
20	Drum instrument mapping	70
21	Volume mapping	70
22	Panning mapping	70
23	Trace selection	71
24	Log file	71
25	Staccato	71
26	Parallelism	72
27	Metrics	72
28	Player	73
29	Player duration mode	73
30	XES loading modes	74
31	XES loading error	74
32	Log summary	77
33	Dotted Chart for filtered repair example - alternative settings	80
34	Uninterrupted state for first 100 traces	85
35	Parallelism increase for increased throughput on the right side compared to uninterrupted state on the left side	86
36	Low frequency case 1 dotted chart	87
37	Low frequency case 2 dotted chart	88
38	BPMN Diagram High Frequency Log - Web shop ([20])	89
39	Zoom into a high frequency log file	91
40	Anomaly 1 prototype visualization	92
41	Anomaly 2 dotted chart visualization	94

42	Anomaly 3 prototype visualization	95
43	Anomaly 3 prototype visualization	96

List of Tables

1	Event log data used for sonification	46
2	Default instruments	52
3	Default melodies	53
4	Default drum instruments	54
5	Default rhythms	55
6	Supported XES concepts	60
7	Sonification and visualization mapping	61
8	Performance measurements	64
9	Use-cases overview	75
10	Execution times high frequency log	90
11	ProM Plugins 1/3.....	127
12	ProM Plugins 2/3.....	128
13	ProM Plugins 3/3.....	129

Listings

1	SuperCollider example	28
2	XES standard extensions: time	37
3	XES standard extensions: concept	38
4	XES standard extensions: lifecycle	38
5	XES standard extensions: organization	38
6	XES standard extensions: semantic	38
7	XES standard extensions: ID	39
8	XES standard extensions: cost	39
9	JFugue Staccato	41
10	Sonification: complete example XES	55
11	Sonification: staccato example	57
12	Sonification: example mapping details	57
13	Use case process mining: log summary	77
14	Sonification sound effect	105
15	Sonification default instruments	105
16	Sonification default melodies	105
17	Sonification default drum instruments	106
18	Sonification default rhythms	107
19	Use cases: overview invalid log not well formed XES	108
20	Use cases: overview invalid log no time stamps XES	108
21	Use cases: process mining control flow 1	109
22	Use cases: process mining control flow 2	110
23	Use cases: process mining control flow 3	111
24	Use cases: process mining control flow 4	112
25	Use cases: process mining case-related	113
26	Use cases: process mining organizational-related 1	114
27	Use cases: process mining organizational-related 2	115
28	Use cases: process analysis low frequency case 1 variant 1	117
29	Use cases: process analysis low frequency case 1 variant 2	118
30	Use cases: process analysis low frequency case 2	119
31	Use cases: process analysis high frequency no anomaly	121
32	Use cases: process analysis high frequency case 1	122
33	Use cases: process analysis high frequency case 2	123
34	Use cases: process analysis high frequency case 3	125

Glossary

API	application programming interface
BPM	business process management
BPMN	Business Process Model and Notation
DSL	domain specific language
GUI	graphical user interface
JRE	Java Runtime Environment
JVM	Java Virtual Machine
KPI	key performance indicator
MIDI	Musical Instrument Digital Interface
MXML	Mining eXtensible Markup Language
OSC	Open Sound Control
PMson	parameter mapping sonification
UGen	unit generator
UI	user interface
XES	Extensible Event Stream

1 Motivation

Business process management (BPM) started with the process orientation trend in the 1990s [45, p.4] where a business process is defined by its inputs and outputs. This allows to redesign the activities of a company by applying business process reengineering techniques. As part of the practical application of this approach, BPM systems are information systems that are aware of the processes and help to apply them. In doing so information about the actually executed processes is generated and can be accessed as event logs. While BPM started following a top down approach the generated event logs can also be analyzed without any assumptions about the underlying process models. The bottom up approach from event logs of real processes to process models is one part of the process mining [38, p.1] method and is influenced by data mining and computational intelligence. This approach is also a good fit to the increasing amount of data available to companies as subsumed under the term big data [29].

Both, business process management and its specialization process mining use visualization techniques to present the designed or mined processes or the raw process execution data. These visualization techniques have problems when process models get more complex and bigger or event logs contain more and more detailed information.

- The screen size is limited so not all information fits on one screen.
- Irregular process patterns may be difficult to identify.
- There may be a large number of parallel executed process instances and these instances have various states.
- Other weaknesses can be found in [31] and summarized in [21, p.248].

Beside visualization sonification is an alternative way to display data. As part of auditory display sonification defines how data can be rendered as sound [18, p.1]. The different steps required to render these sounds can be identified as audification, auditory icons and earcons to name only the most popular. The most important argument to apply sonification to event log input data, is the conceptual fit between sonification and time based event logs. Sonification is a temporal medium as opposed to visualization as spatial medium. While the suitability of visualization for process mining and process analysis has been formally and practically evaluated there is little research to the application of sonification for this purpose. So the well understood sonification approaches can be taken and applied. The sonification can be used together with visualization as a multimodal tool and its results can be compared to visualization and specialized algorithms.

To perform this application in practice concrete tools would be required but as most other business process related tools the popular process mining tool ProM¹ tool has no sonification support. This is in contrast to multiple plug-ins handling the visualization of event logs and mined process models. This lack of sonification support applies also to other BPM and process mining tools. On

¹ <http://www.promtools.org/doku.php>

the other hand there exist sonification tools, but these can not deal with native business process event logs as they often expect numeric data. By being limited to numeric data only part of the information in an event log can be sonified with these tools and the extraction of the data and transformation in the supported format requires additional work.

As starting point for evaluating the sonification options the Sonification Handbook [18] offers an extensive overview about sonification and its application. Concrete sonification tools are described in [43,5,27]. Research for applying sonification to business processes can be found in [15,46] and more closely related to the topic of this thesis in [21,22,23,19]. For the practical application of sonification ProM is the only viable alternative considered. While it describes itself as process mining tool, it can also be used for process analysis and its extensibility with plugins allow to apply sonification to existing event logs.

1.1 Research Question

After the overview of the relevant aspects of business processes, sonification, visualization and the already existing research a more concrete formulation of the research questions for this thesis can be presented. The application of sonification for business process mining and business process analysis. This sonification is used together with visualization in a multimodal approach.

As shown in Section 2.6 there is little known research that is directly related to sonification for business process event logs. As the research question is further limited to process mining and analysis, the process mining related research loses relevance. In the remaining work only Hildebrandt, in [22, p.4ff], shows concrete examples for sonification based on event logs. But the visualization aspect of the multimodal approach is missing completely. The existing sonification tools, shown in Section 2.3, are either too specific to use all the data available in an event log or so generic that there are no well understood ways how a sonification for an event log may be performed. The main problem is the missing knowledge of these tools about the hierarchical structure and the meaning of the standard extensions found in a typical Extensible Event Stream (XES) event log. This applies also to the visualization approaches of these tools, as they are also not aware of the XES structure.

After showing that there are little to none concrete research and practical tools that apply a sonification first multimodal approach for process mining and analysis, this thesis has to handle three main aspects in its research question.

1. Are sonification techniques and concrete mappings useful to perform typical process mining and process analysis tasks? If so which insights can be gained by these techniques and mappings? Which properties of the process data are beneficial for sonification?
2. Which kind of visualization fits to a multimodal approach where the focus is on sonification?

3. Both sonification and visualization have to be configured and applied to concrete logs. Log data may have to be loaded and filtered. The sonification needs to be played and the visualization shown. All these activities require user interactions, leading to the question how the tool can best support the user in these interactions.

The results of this thesis are therefore primarily the chosen sonification and visualization techniques and mappings described in Section 3 and how they are put into practice in the executable prototype. The supported user interactions, to answer the third part of the research question, are answered by the concrete user interface (UI) of the prototype. The suitability of these design decisions is evaluated by performing typical process mining and process analysis tasks in Section 5. The use cases cover typical aspects but are not comprehensive so that additional use cases may lead to new requirements for multimodal process mining and analysis. The benchmark for the suitability of these concepts are concrete existing visualization tools available in ProM.

The expected outcome of the use case evaluation is a general suitability of sonification supported by visualization for the tasks typically solved by means of visualization alone. But it is to be expected that not all questions can be solved with the current implementation status of the tool as the visualization approaches are much more mature.

Concrete expected results for the three research questions are: For sonification, techniques can be found that allow to reflect changes in multiple data dimensions. For these techniques, concepts that one may expect the typical user to be already familiar with, like everyday sounds, music as opposed to random sounds should be easier to remember and identify. The supplementary visualization should cover at least all sonification mapping dimensions. Finally the UI interactions are organized in similar ways as existing ProM packages dealing with visualization. This includes support to deal with large event logs by offering zooming and filtering capabilities optimized for sonification.

1.2 Contribution

This thesis is most closely related to the work done in [21,22,23,19]. In contrast to other sonification applications, it discusses sonification mappings for event log data found in the standard extensions as described by the XES [17] standard. This allows for the first time to experiment with sonification for the data found in these logs without the need to transform this data as it would be required for existing sonification tools. The sonification is focused on these plain event logs and a visualization similar to the one found in [34] is chosen to support the creation of the sonification mapping in a multimodal manner.

The evaluation of the suitability of the proposed sonification is done for typical process mining and process analysis use cases. These use cases are exemplary for process mining and process analysis and allow to draw first conclusions about the suitability of this approach. Technical limitations of the tool

presented in this thesis don't allow to find sonification for all tasks. These shortcomings may serve as starting point for further evaluations of business process sonification.

For the sonification, the suitability of the most common techniques is evaluated for the given event log input data. The chosen techniques can be configured by identifying useful mapping concepts. For the visualization the focus is on identifying additional aspects that make the visualization more helpful as an accompanying technique for the visualization. For both sonification and visualization technical limitations and best practices are identified when implementing a multimodal tool using the ProM technology stack. For the UI interactions the basic concepts are taken from existing sonification tools and ProM visualization tools. These concepts get extended by the practical requirements when using the multimodal tool to solve concrete use cases.

1.3 Structure of Thesis

The remainder of this thesis is structured as followed. Section 2 covers how the business process data is generated and which basic sonification technologies are available. Together with a presentation of existing research, tools and environments this allows to define the research question with more precision. The design and implementation of the tool are shown in Sections 3 and 4 where the ideas for a multimodal event log analysis are laid out and concrete implementation aspects are considered. The suitability of this tool is evaluated in Section 5 for process mining and process analysis. Section 6 sums up the results of the work done in this thesis and shows possible starting points how to build upon these results. The remaining two content-related sections, Section A and Section B, serve as reference for design aspects of the tool and list details how the use-cases are sonified and visualized. All other sections are required to cover formal aspects of a thesis and don't contain new insights on the subject of this thesis.

2 Concepts and Related Work

A detailed discussion of the developed tool and its concepts is done in Section 2 and Section 3. But before this the input data producer, business processes, the resulting event logs and analysis and discovery methods are discussed in Section Business Process Management and the main output of the tool, the Sonification (2.2) are introduced. Along with discussing the principles behind sonification follows an overview of the existing sonification tools and environments Sonification Sandbox (2.3.1), SuperCollider (2.3.2) and xSonify (2.3.3). These solutions are not targeted at a specific domain and therefore not for business process data so the main interest in evaluating them is to find basic sonification approaches that can also be applied to business process sonification. Before the research question, hinted at in the motivation chapter, is specified in all details the related work section discusses research done related both to sonification and business processes. ARKola (2.6.1) and Grooving Factory (2.6.2) both apply sonification to business process data. While ARKola is focused on monitoring support with auditory icons Grooving Factory uses audification as sonification technique. Business Process Sonification (2.6.3) as the name implies is closely related to the topic of this thesis even though the concrete application is to some extent focused on monitoring and not process mining and analysis. But as opposed to the other related work the developed ideas are not put to practice in the form of an executable tool. Other topics covered in this section are the multimodal companion to sonification Visualization (2.4), a introduction to Multimodal Approaches (2.5) and the concrete platform ProM (2.7) to build upon the tool.

2.1 Business Process Management

Business processes consist of activities [45, p.5]. Process and activity models can be executed leading to concrete business process instances and activity instances. The motivation for describing an enterprise in the form of business processes is to narrow the gap between the organizational business aspect and the used information technology [45, p.4]. The same processes that are used to plan an organization may be later executed in the information systems. The information systems responsible for executing the business processes are called BPM systems [45, p.6]. BPM deals with the design, administration, configuration, enactment and analysis of business processes [45, p.5].

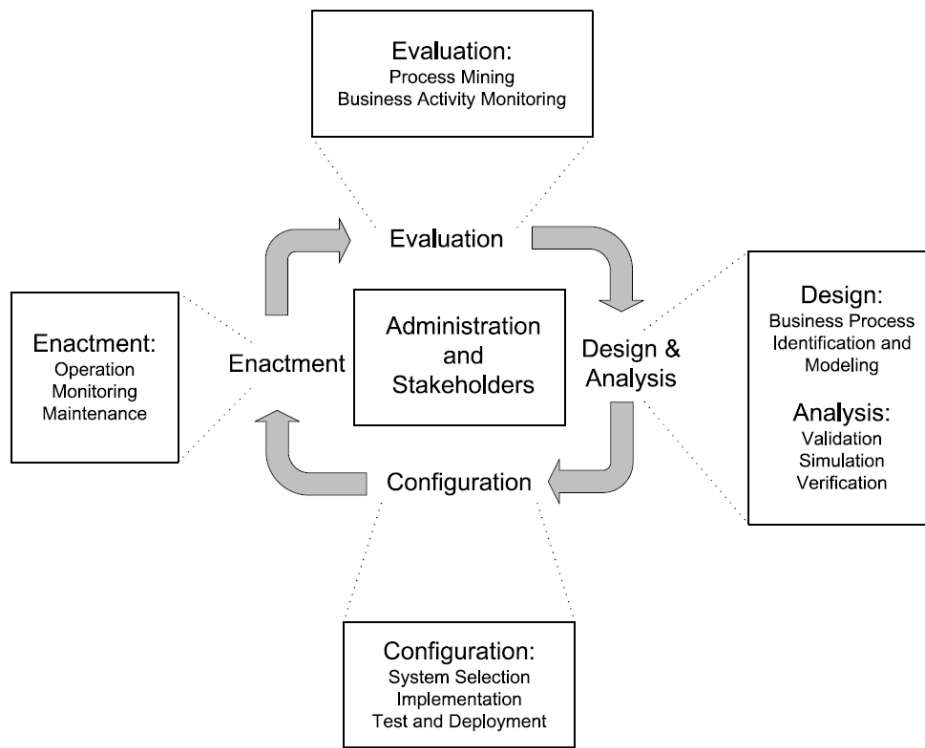


Fig. 1: Business process lifecycle [45, p.12]

A business process is used in multiple contexts and may change over time. Figure 1 shows the business process lifecycle with its four main phases that try to capture all these different aspects. Relevant for the prototype described here are the evaluation phase [45, p.15] and the analysis part of the design & analysis phase. Here the results of the process execution, event logs are evaluated either against an existing process model (see Section 2.1.3) or a new process model is created from these logs (see Section 5.1).

2.1.1 Event Logs

XES² is a domain independent, extensible XML-based standard for event logs [17]. It is described here because the sonification prototype is implemented as plugin for the ProM tool and this tool uses XES as default format for event logs. Event logs in general capture data when business processes are executed. This data has the following properties.

- Hierarchical, a process may be executed multiple times and each instance may consist of multiple activities.

² <http://www.xes-standard.org/>

- Chronological, each activity happens at a specific time so a time stamp is usually attached to each event.
- Widely used knowledge, like information about the state, the user or the role of an event can be saved in a standardized way by using a fixed set of standard extensions.
- Domain specific knowledge, that can't be mapped to one of these standard extensions, can be covered by defining custom extensions.
- Typed information with optional human-interpretable semantics allow for a precise representation of the data produced by the business processes.

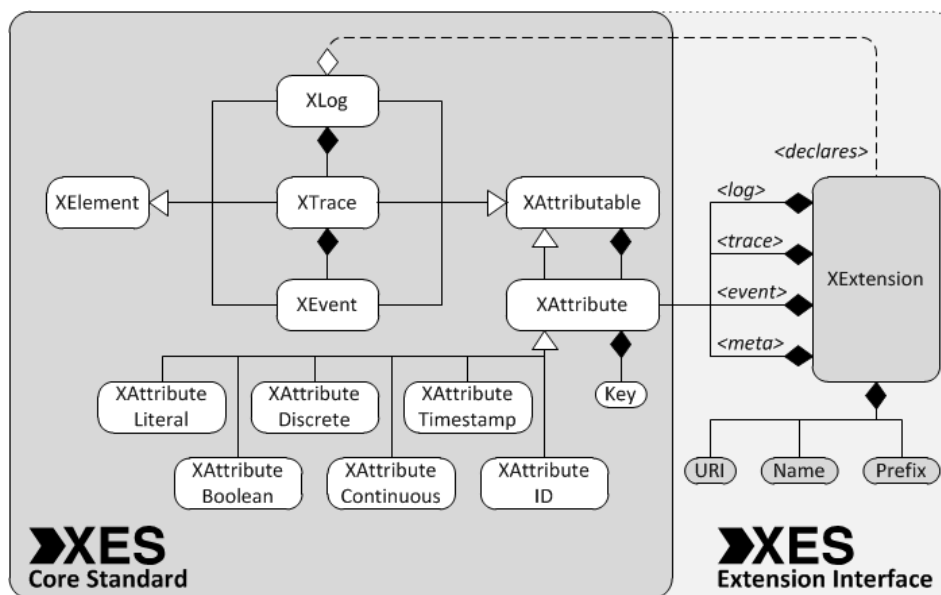


Fig. 2: XES core and extensions

Figure 2³) shows how XES as XML dialect covers these requirements for event logs. The hierarchy of process, instance and activity is mapped to *XLog*, *XTrace* and *XEvent*. Chronological and other information is stored in *XAttributes*. For example the chronological information stored in each event, a time stamp, can be mapped to an *XAttribute* of the type *Timestamp*.

2.1.2 Process Mining

Process Mining supports discovering, monitoring and improving business processes [37, p.99:1]. The input for these tasks is real-world process data in the

³ <http://www.xes-standard.org/openxes/start>

form of event logs. The main difference to top down BPM approaches is the lack of hand-made models. Process mining discovers these models in the event logs using several algorithms like the α -algorithm. The resulting model then has a direct conformance to the event logs used for its discovery. Each event in the event log can be mapped to an activity in the process model.

[38, p.4] distinguishes three different basic types of process mining.

1. Process discovery uses an event log and generates a process model from it. This may lead to much more complex process models also known as spaghetti processes [37, p.10] as opposed to the usually conceptually cleaner processes when applying a top down approach.
2. Conformance checking compares the actual event logs against an existing model and has the purpose to identify deviations in both directions.
3. Model enhancement is similar to conformance checking but focuses on extending an existing model so that this extended model fits to event logs.

Another way to distinguish process mining is the analyzed perspective. The most prominent perspective is the control-flow perspective focusing on the ordering of the activities and resulting in process models often using the Petri net or Business Process Model and Notation (BPMN) notation. Another perspective is the organizational perspective focusing on organizational information like persons or roles available in the event logs. Important for this thesis is also the time perspective as sonification as a temporal medium can benefit from time stamp data in the events. The suitability of process mining depends also on the current phase according to the BPM life-cycle. [38, p.5] shows how for other phases than the diagnosis phase process mining can be used. When dealing with real world event logs they often need to be prepared before they can be used as they may be incomplete or incoherent [38, p.10]. Even when the log quality is sufficient, the log may be too small or too large, requiring an additional preprocessing step before the data can be used.

2.1.3 Process Analysis

Process mining concentrates on the discovery of process models, model conformance checks and model enhancements. Process analysis focuses on identifying issues in the as-is process [11, p.22].

Qualitative process analysis can be done with multiple techniques as shown by [11, p.185ff]. It can be analyzed which steps of a process add value, leading to the removal of non-value adding steps. Another aspect is to identify the root cause of issues and distinguish causing and contributing factors. The resulting issues need to be collected and prioritized.

Quantitative process analysis allows to calculate performance measures [11, p.213ff]. The first step is to define the performance measures of a process like time, cost, quality and flexibility. When quantitative data about the activities is available flow analysis can help measure the performance of a process. In cases of resource contention queuing theory allows to deal with the effects of queues. The most popular quantitative process analysis technique today is simulation

where many instances of a process with different properties are generated and get executed using a simulation tool. These executions result in quantitative measures like cycle times or waiting times.

2.2 Sonification

[18, p.1] describes *auditory display* as all aspects of human-machine interaction related to audio. This includes technical aspects like the number and the placement of speakers. *Sonification* on the other hand is a part of the auditory display and describes how the audio is rendered depending on the input data. Sonification is an interdisciplinary field as it requires an understanding of amongst other things physics, acoustics and musicology.

When sonification is applied to a specific use case, like business process sonification, the resulting sonification can be characterized in different ways. There are different techniques (shown in the Sections 2.2.1 to 2.2.5) that may be adequate to varying degree for the use case. Knowledge about these techniques is important but it may be difficult to know when to choose which technique and how these techniques should be combined. For this purpose a sonification design space map [9] can be useful. It distinguishes sonifications along three dimensions.

1. The number of data points that influence the sonification.
2. The number of data properties (dimensions) that influence the sonification.
3. The number of parallel streams presented to the listener.

These dimensions help to choose between *Discrete Point Data Representation*, *Continuous Data Representation* and *Model-Based Data Representation*. For example model-based data representation requires a higher number of data properties than discrete point and continuous data representation. But even when making a design change within one data representation type using this map can be used to identify the position before and after the design change. For example one could change a discrete data data point representation sonification from containing only a single stream to parallel streams. Applied to business process event logs this could mean applying the data property trace belonging available in every business process event to the stream on which it should be sonified. This can either add the trace belonging to the sonification if it wasn't considered so far or it may free this data property (dimension) when the information is now part of the stream.

A list of design choices relevant for common sonification implementations contains:

- The data type can set a matching sonification technique. [18, p.18] distinguishes between nominal data and ordinal data where nominal data has no meaning beside a group membership whereas ordinal data is ordered to some quantity.
- [18, p.24] describes how to choose which data dimensions get mapped to which sound dimensions. For example [42, p.212] evaluated that loudness

is better suited to represent temperature than pitch which itself is better suited than tempo.

- Scaling described in [18, p.25] takes care of mapping the input range of the data property to the output range of the sound property. For example when mapping a temperature range from 0 - 50° Celsius to pitch a naive approach would just multiply the temperature by 400 resulting in a pitch range from 0 to 20.000 Hz. But this wouldn't make good usage of the human aural sense which is most sensitive between 1000 and 5000 Hz.
- [33, p.1067] "Context refers to the purposeful addition of non-signal information to a display" and can be applied to sonification in the form of repeating clicks like a metronome.

Concrete sonifications contain at least one of the five following sonification techniques listed in [18, p.299ff].

2.2.1 Audification

Audification is defined as a direct translation of data waveform into sound [28, p.xxvii]. According to [18, p.302f] four types of data can be used as input for sonification.

1. *Sound Recording Data*: Today mainly digital recordings played with a Digital-to-Analog converter.
2. *General Acoustical Data*: Elastomechanic measurements that are interpreted acoustically, for example from a sonar.
3. *Physical Data*: Physical processes that have no mechanical domain resulting in unfamiliar soundscapes.
4. *Abstract Data*: Data from a non physical system, for example from the stock market.

Abstract data is the only type fitting to business process sonification as the event logs themselves are based on abstract processes. To further evaluate if audification might be appropriate for business process sonification [18, p.319f] defines eight preconditions from which the most critical are listed here.

- At least a few thousands samples are required.
- Best fitted to data based on physical laws that would normally be outside the frequency range of the human hearing.
- Frequency or rhythm changes might be spotted easier than drifting.
- Data screening allows to get a quick overview of a data set.

The main obstacle to using audification is the the required amount of data. Many business processes contain only single- or double-digit number of events and therefore audification could only be used if many instances of the process would be sonified.

2.2.2 Auditory Icons

Auditory icons are sounds similar to everyday sounds [18, p.325] and should therefore already be familiar to the listener, as they have a predefined meaning, that doesn't require an explicit explanation. They were first used to complement visual UI events with sounds. The SonicFinder [14, p.5] extends the Macintosh file manager Finder. For example a selection with the mouse or keyboard in the file manager produces a hitting sound. The type of the selection (file, folder etc.) is mapped to a sound source like wood or metal and the size of the selected element is reflected by the pitch of the generated sound. This gradual adaption of the sound to changes in the causing event is a property of parametric auditory icons. Other adaptations could be for example changing the volume or filter specific frequencies [18, p.331]. Another aspect to categorize auditory icons is to distinguish *fully formed objects* and *evolutionary objects* [4, p. 2.2]. While fully formed objects are completely created at the time when the playback starts evolutionary objects may change according to input only available after the playback started.

For the concrete design of auditory icons in the domain of business process sonification several decisions would have to be made.

- The usage of recording and/or synthesized sounds for auditory icons.
- Parametric auditory icons as opposed fixed sounds allow to change with the input data. Possible transformation effects may be loudness and volume.
- [30, p. 71] created a methodology for designing auditory icons that not only considers how to design auditory icons but also how the icons get identified and how the the learnability success rate can be evaluated.

The ARKola simulation described in detail in Section 2.6.1 uses auditory icons for process monitoring and collaboration.

2.2.3 Earcons

[2, p. 7] defines earcons as "*... abstract, synthetic tones that can be used in structured combinations to create sound messages to represent parts of an interface.*". [18, p.339] emphasizes the musical aspect where scales are affected by the input data. The main difference to auditory icons is the missing predefined connection between the sonification and the conveyed information. This connection has to be learned in some way by the user of the earcons. Tests by [13] showed that unfamiliar earcons without training have only recognition rates around 10%.

[1, p. 28ff] originally proposed earcons and how to categorize them. [18, p.340ff] gives a more up-to-date overview about earcons.

- *One-Element Earcons*: Similar to non-parameterized auditory icons, but as they are earcons, they have no intuitive predefined meaning. An Example is smartphone notification sounds where different sounds may be assigned to different event types but without a structured relationship between the sounds.

- *Compound Earcons*: Combine one-element earcons to convey more information. [2, p.59f] gives an example by assigning an earcon to each of the words *create*, *destroy*, *file* and *string*. Now each of the two verbs could be combined with each of the two nouns to get a total of four compound earcons.
- *Transformational Earcons*: Rules define how data attribute values affect sound attribute values. Sound attribute values could be timbre, melody or register.
- *Hierarchical Earcons*: Use a set of rules like for transformational earcons but additionally use an inheritance hierarchy to pass properties from top-level elements to elements below.

When designing earcons [12] provide concrete recommendations for the auditory attributes timbre, register, pitch, rhythm, duration, tempo, intensity, spatial location and how to build compound earcons.

2.2.4 Parameter Mapping Sonification

“Parameter Mapping Sonification (PMSon) involves the association of information with auditory parameters for the purpose of data display.” [18, p.363ff]. For example for the boiling process of water in a tea kettle, PMSon would allow for many different sonifications.

- Only when the boiling point is reached a special sound is played.
- The temperature in the tea kettle could be mapped to the frequency of the sonification and a special sound is played when the boiling point is reached.
- Additional to the rising frequency various intermediate stages are indicated by specific sounds. For example the five stages of boiling water according to Chinese traditions.

A strength of PMSon is its ability to map different data dimensions to the different sound dimensions like volume or balance. This gives the designer of a PMSon many options but may also lead to overwhelming or hard to learn sonifications.

2.2.5 Model-Based Sonification

Model-Based Sonification [18, p.399ff] takes into account how the user interacts with an environment that produces an acoustic response. Acoustic responses are ubiquitous in the physical world when a person interacts with nature and gets acoustic feedback about it. Computer interfaces on the other hand are often limited to visual interfaces and don’t provide acoustic feedback. This is the motivation for the model-based sonification technique. When designing such a sonification it’s important to be aware of the distinction between passive sound where the listener does not influence the sound creation and active sounds where the listener influences the sound creation. Active sounds can be further differentiated in sounds where the user is influencing the sound creation by intention and other cases where the sound creation is not directed by the user. An example for the first case is the playing of a musical instrument while the sound of a tennis player sliding on a clay court is not intentional.

[18, p.419ff] shows four application fields for model-based sonification from which exploratory data analysis and process monitoring are applicable to business processes. Exploratory data analysis tries to cover the gap between the input data model and the preferred way the human perception recognizes patterns in data. For process monitoring the excitation usually performed by the listener has to be taken over by the data.

2.3 Existing Sonification Tools and Environments

2.3.1 Sonification Sandbox

Sonification Sandbox⁴ is described in 2003 by [43] but the software development still continues. The features described here refer to version 6.2beta from December 2014. This tool allows to sonify and visualize numeric data. The data input must be in the form of a spreadsheet. Exactly one column must be mapped to the time dimension of the sonification and visualization. The values in this column must increase monotonically from top to bottom so that a simple transformation to relative time values is possible. The main difference to business process sonification is the numeric data input, e.g. temperature measurements over a time period, instead of structured events available in an event log. The visualization is a dot plot with the time column on the x-axis and the value from the other column on the y-axis. Additionally a progress bar helps to keep track about the current state of the sonification.

⁴ http://sonify.psych.gatech.edu/research/sonification_sandbox/

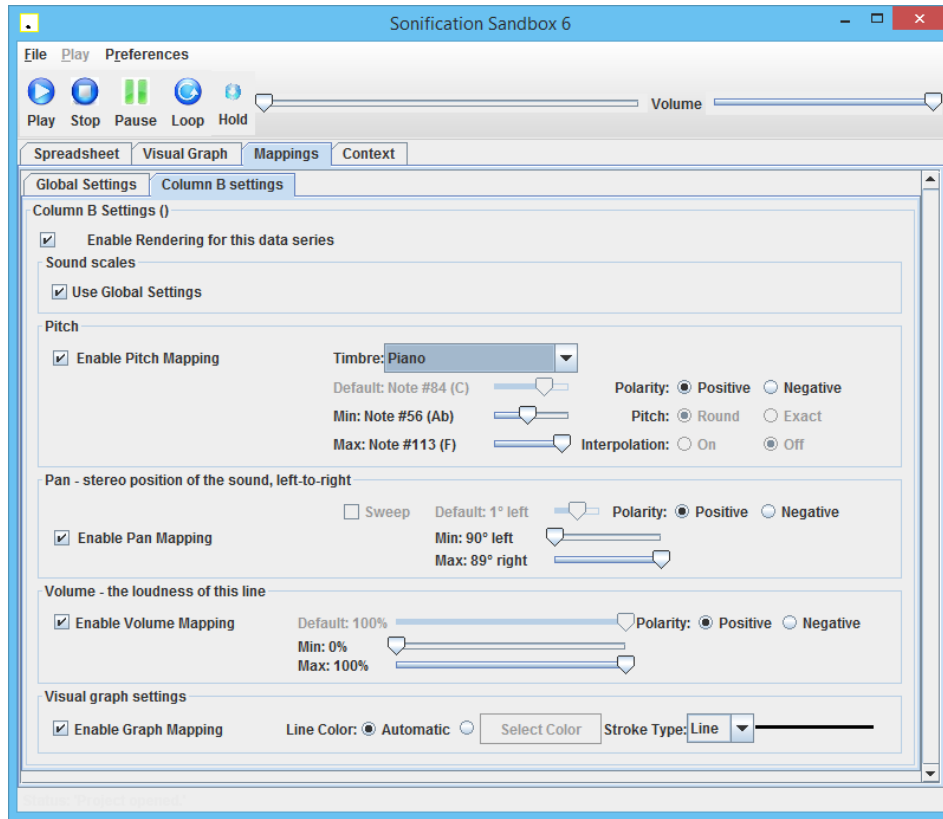


Fig. 3: Sonification Sandbox mappings

Figure 3 shows the player control and the mapping options. Beside the usual controls for play, stop, pause and volume a looper allows to listen to specific points repeatedly and thereby get a better understanding of a period. *Global Settings* are applied to all data while column specific settings apply only to a single column. In the *Global Settings* the sonification speed can be set by specifying the total duration of the sonification and the progress bar can be switched on and off.

The column specific settings allow to map the following sonification options:

- Choosing an instrument from the available MIDI instruments on the computer running the software.
- A detailed mapping how changes in the column data influence the pitch, pan and volume of the sonification.

Also the visualization can be customized:

- Toggle the visualization for a specific column.
- Modify the concrete visualization.

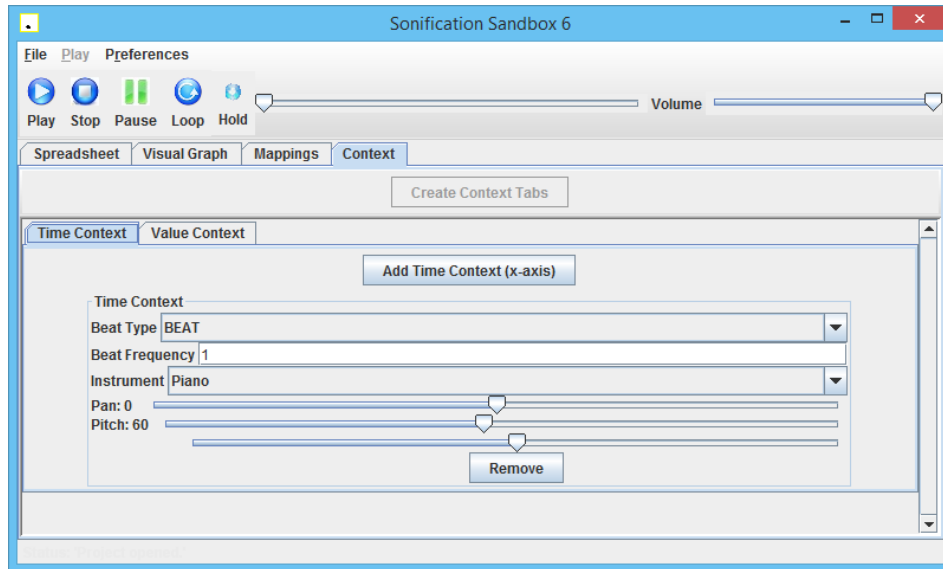


Fig. 4: Sonification Sandbox context time

Figure 4 allows to add context to the sonification comparable e.g. to tick-marks that can be found in a visual graph. The first context option allows to repeat a signal at regular intervals, comparable to a metronome.

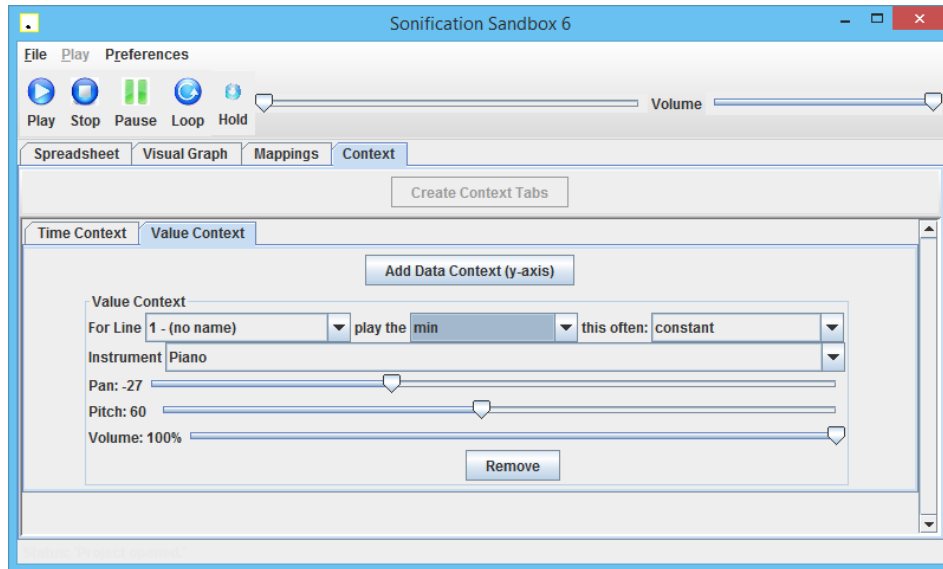


Fig. 5: Sonification Sandbox context value

Figure 5 on the other hand adds context depending on the concrete values of specified columns. For example when the maximum value of a column is reached a notification is played.

The feature analysis of the Sonification Sandbox provides valuable ideas for the sonification and visualization of business process events.

2.3.2 SuperCollider

SuperCollider⁵ provides a programming environment including a special language (sclang) for real time audio synthesis and algorithmic composition and delivers also an IDE for this programming language. It even allows the user to build custom graphical user interfaces (GUIs). The programming language sclang is dynamically typed and supports object-oriented and functional programming concepts. It allows to interact with the generated sound and the GUI and can consider various input events for the sound and GUI calculation. The following example taken from the homepage shows how the position of the mouse can influence the generated sound in real time.

Listing 1: SuperCollider example

```
// modulate a sine frequency and a noise amplitude with another sine
// whose frequency depends on the horizontal mouse pointer position
{
  var x = SinOsc.ar(MouseX.kr(1, 100));
```

⁵ <http://supercollider.github.io/>

```

SinOsc.ar(300 * x + 800, 0, 0.1)
+
PinkNoise.ar(0.1 * x + 0.1)
}.play;

```

SuperCollider is not limited to a specific domain for the generated sounds. Beside defining new sonifications there also more than 1000 (December 2015) unit generators (UGens)⁶ ranging from small building blocks like a random number generator⁷ that may be used in a sonification to modify a property of the generated sound to synthetic instruments like a piano⁸.

SuperCollider can also be integrated with other sonification tools using the Open Sound Control (OSC) protocol⁹. OSC uses network based messages to exchange sonification information. It has several advantages compared to the Musical Instrument Digital Interface (MIDI).

- It is network based and therefore doesn't require specific hardware.
- Higher precision parameter values avoid abrupt unintended changes in sonification.
- A client can determine the capabilities of a server.
- The message routing allows to determine the recipients of a message with pattern matching.

The main disadvantage compared to MIDI is the missing standardization of the message semantics but that also allows for a greater extensibility.

2.3.3 xSonify

xSonify¹⁰ is a sonification tool and was originally developed to allow visually-impaired students to make better use of space science data provided by the NASA [5]. A relatively up to date description of its features along with two use cases for space science data can be found in [10, p.133ff].

- It can handle up to 2-dimensional data from text file typically in the form of time series.
- The sonification options are pitch, volume and rhythm.
- Playback controls including setting the speed, loops and time points are supported.
- Loaded data can be pre-processed by limiting the range of values that should be considered for the sonification or by transforming the data.

A screenshot of the application taken from [10, p.134] is shown in Figure 6.

⁶ <http://doc.sccode.org/Browse.html#UGens%3ERandom>

⁷ <http://doc.sccode.org/Classes/IRand.html>

⁸ <http://doc.sccode.org/Classes/OteyPiano.html>

⁹ <http://opensoundcontrol.org/introduction-osc>

¹⁰ <http://sourceforge.net/projects/xsonify/>

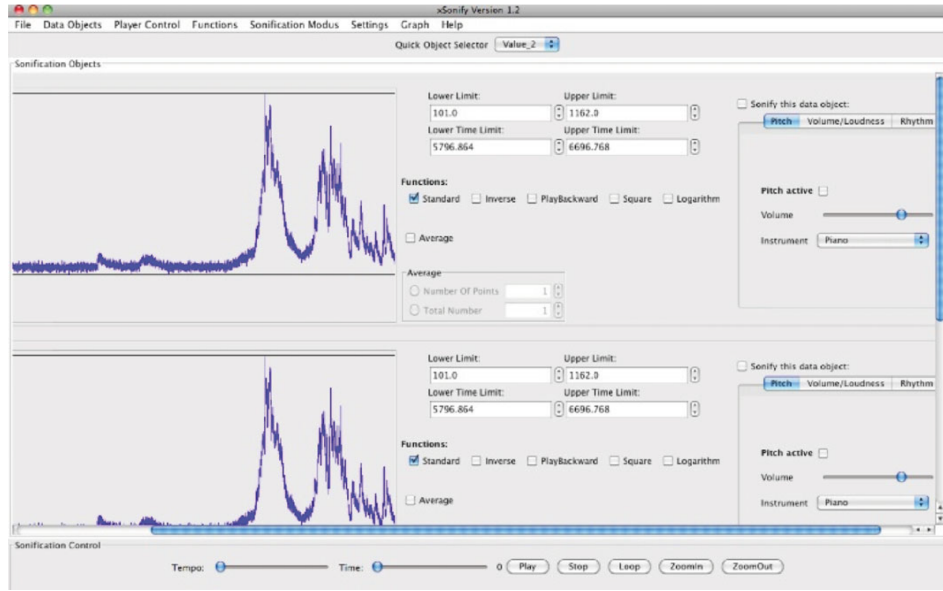


Fig. 6: xSonify UI

To sum up the presented tools and environments the main weaknesses are either the focus on numeric data for the Sonification Sandbox and xSonify or being a too general solution like SuperCollider. Both approaches can't be used directly for process logs for example in the XES format. Limiting the tool input to numeric data neglects the rich nominal data available in event logs and a general solution requires the sonification user to define its own sonification mapping concepts using the capabilities of the environment. SuperCollider can be therefore seen as possible implementation technology and not as ready made solution or tool. Its implementation technology capabilities are discussed in Section 3.2.2.

2.4 Visualization

The evaluation of visualization in the context of this thesis is limited to popular visualization approaches mentioned in current books covering BPM [11], and more focused books covering only business process mining [36]. Existing visualizations like cockpit views are not directly bound to event logs or process models. Process mining specific visualizations support Petri nets, social networks or event-driven process chains. These visualization types share the property that they visualize a model from the event logs. The differences between them are caused by the different perspectives.

Song et. al. [34] categorize the approaches mentioned above as either a *data driven approach* when aggregated data is shown using graphs or graphical metaphors

like a speedometer. Petri nets and other process model visualization are following a *process driven approach* and require a well-formed process model. To show the process execution without additional abstraction they propose another approach that is best suited for the discovery aspect of the three basic types of process mining. It is called dotted chart and plots dot according to some time information available in the event log. The concrete implementation as ProM plugin allows to vary both the aspect of the event log on which the user focuses and to choose between different ways in which the time values available in the event log are influencing the positioning of the dots in the visualization. A zoom functionality allows the user to limit the visualized information and helps to identify patterns in the visualization.

2.5 Multimodal Approaches

A multimodal system needs to make use of at least two modalities for input and/or output [44, p.7]. [6] sees the term modality in a one to one relationship to the human senses vision, hearing and feeling while [35] already counts systems containing text and graphic output as multimodal systems even though text and graphics are just two information representations in the vision modality. In the context of this thesis multimodal emphasizes the support of vision and hearing. One motivation for supporting both modalities is that a sonification tool may work without a visualization part but in many cases visualization is supported. For example the Sonification Sandbox and xSonify discussed above both visualize the data along with the sonification. SuperCollider even though mostly known for its elaborate sound synthesis capabilities has support for visualization as described in [27] where visualization aspects like vector graphics and animations are described.

Reasons for adding visualization to sonification in the context of business processes could be.

- To make use of all the well-established visualization approaches that are used for event logs or mined process models. This allows the user to have a better understanding of the data and allows the user to make more appropriate choices for the sonification mapping.
- The creation of visualizations that reflect the effects of the sonification mappings applied to the event logs. So if a user of the tool makes a change to some mapping he/she can estimate for example the influence range of this without being required to listen to the whole sonification.
- When listening to the sonification a progress indicator in the visualization helps the user to map the things heard to the visualization. If the user now wants to inspect the sonification of the last few seconds he/she can start using the visualization with the current state of the progress indicator to get more information about the currently sonified data.

2.6 Related Work

2.6.1 ARKola

ARKola [15] simulates a soft drink factory and uses auditory icons for a sonification that allows users to manage the production process. One interesting aspect of ARKola for business process sonification is how these auditory icons were chosen. Everyday sounds should allow the user an intuitive understanding of the sonification. For example the bottle dispenser sounds like clanking bottles. An increase in the production rate leads to an increase in the number of sound events. The different sounds should have properties according to their intended meaning. For example urgency may be reflected in the number of the transients of the sound. Other aspects of the ARKola simulation like the collaboration of two persons or the real-time aspect don't match the requirements for business process sonification based on recorded event logs.

2.6.2 Grooving Factory

Grooving Factory [46] is an attempt to identify bottlenecks in production processes during the planning and control phase. Production process data similar to business process data has an inherent time ordering and is therefore well suited for sonification. To create an auditory display (sonification) a mapping has to be defined which data gets displayed and how this data is mapped to sound. Limitations of the human hearing like the limited frequency range have to be considered and it should be avoided to overwhelm the user with too much concurrent audio events as this would prohibit identifying relevant behavior. This overwhelming of the user might also be an interesting aspect when thinking about business process event data where the concurrent sonification of too many traces might prevent the user from identifying interesting aspects of the data.

2.6.3 Business Process Sonification

A related series of papers [21,22,23,19] correspond closest to the subject of this thesis. All four papers discuss the possibilities of applying sonification to business process event logs. [21,22] give a general introduction to the topic.

- The limitations of visualization as listed in Section 1.
- The fit of the temporal medium sonification for the time based business process event logs.
- Proposing a multimodal approach to combine the strengths of sonification and visualization.
- The different sonification techniques shown in Section 2.2 are evaluated for their suitability for business process sonification in the different BPM phases. Audification has the disadvantage of requiring huge amounts of quantitative data that is not available in many logs. Auditory icons and earcons are both suitable with earcons having the advantage of being easier

to create when no auditory icon can be found to cover a specific information. A further information increase can be achieved by using parameterized auditory icons or earcons to include quantitative aspects of the log data. This required quantitative data is also a reason why parameterization is more suitable for the process operation and analysis than in the process design phase. When monitoring a process or analyzing a finished process quantitative data gets accumulated.

The two remaining papers [23,19] focus more on the process monitoring aspects of business process sonification but some ideas developed for this BPM phase may also apply to process mining and analysis.

- Multimodal approaches show better results than sonification or visualization only approaches.
- The sonification of events when they occur as opposed to sonifying derived data like key performance indicators (KPIs).
- Identifying the problem that the sonification may get overwhelming when too many events happen in a short time period.
- A prototype was developed that uses melodies and instruments to convey information about event data.
- The need to filter the event data and therefore influence the resulting sonification as it is appropriate for specific users of the sonification.

After showing the closest related work it can be seen that ARKola and the later business process sonification papers focus on process monitoring and not process mining or process analysis. Grooving Factory is focused on the process analysis of a specific domain and uses therefore a customized approach to generate its sonification which is a mixture from audification and parameter based sonification. All these concrete discussions of applied sonifications don't match the requirement of dealing with random business process event logs for mining or analysis. The earlier business process sonification papers match the subject but don't provide enough information to design a concrete tool. Building the required knowledge to design such a tool is therefore one focus of this paper.

2.7 ProM

ProM¹¹ is a cross-platform Open Source framework for process mining. It can be used out of the box on all major desktop platforms but also aims at developers that want to extend its functionality. It is widely used in the academic process mining community. For example when looking at its core team¹² Eric Verbeek¹³ is the person responsible for the framework but also contributes to research papers like the process mining manifesto [38]. So there is a significant overlap between topics discussed in the research community and the actual implementations found in the ProM tool.

¹¹ <http://www.processmining.org/prom/start>

¹² <https://svn.win.tue.nl/trac/prom/wiki/Team>

¹³ <http://www.win.tue.nl/~hverbeek/>

An event log may be the starting point for working with ProM. If the log is in XES format ProM can directly parse and import it using the XES reference implementation OpenXES¹⁴. Also the older Mining eXtensible Markup Language (MXML)¹⁵ format is directly supported by the framework but all interactions with the log after it is imported is handled in a way along the XES standard definition. If the event data is not available as an event log XESame¹⁶ [3] allows to define a mapping from data stored in a relational database to an XES event log. XESame is provided as part of the ProM download package.

The ProM program itself handles the three aspects of process mining: model discovery, conformance checking and model extension¹⁷. The models are discovered in event logs and conformance checking allows to compare event logs against an existing model. ProM also allows to analyze and convert event logs and models.

The architecture of ProM reflects the diverse functionality and the fact that most functionality is not provided by the core framework and its developers but developed independently. Therefore, the core of ProM is designed for extensibility with plugins. For example the previous version 5.2 features over 280 plugins [40, p.34]. Each of these plugins has a well defined input and output with this information the framework can combine the plugins in useful ways so that the output of one plugin is used as input for the next plugin. A typical use case involving a plugin for each step could be.

1. Parsing of a XES event log file.
2. Enriching the parsed event log data.
3. Discovery of a model.
4. Transformation of the model.

All the intermediary results created by the single plugins are stored in an object pool like workspace [46, p.35]. So that intermediary results can be reused and need not be recalculated each time they are used for further processing. The user can even decide to run resource intensive plugin executions on other nodes and still keep the results of these executions available in one workspace.

2.7.1 ProM integration

Before deciding how to achieve the required functionality, it needs to be discussed which functionality of existing plugins can be reused or extended. As the principal idea is to support a multimodal business process sonification the support for visualization and sonification has to be evaluated. A complete listing of the investigated plugins can be found in Section C.

¹⁴ <http://www.xes-standard.org/openxes/start>

¹⁵ <http://www.processmining.org/logs/mxml>

¹⁶ <http://www.processmining.org/xesame/start>

¹⁷ http://www.processmining.org/_media/flyers/prom_flyer_main.pdf

2.7.2 Visualization Support

ProM as process mining tool handles process discovery, conformance checking and model extension. All these types of process mining techniques deal with process models and therefore, many plugins support the visualization of these models. Petri nets are widely used and Figure 7 is an example for a simple Petri net visualization. Other visualization options are targeting BPMN [24] or social networks [41, p.22].

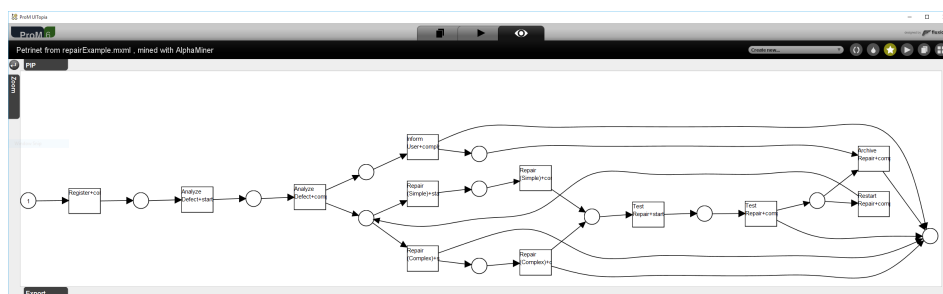


Fig. 7: ProM AlphaMiner visualization using the *repairExample.xes* from <http://www.promtools.org/prom6/downloads/example-logs.zip>

All these approaches have in common that they are visualizing a mined network representation and not the single traces and events as they are found in an event log. As already discussed in Section 2.4, visualization of the unprocessed event log is available in the dotted chart visualization¹⁸ shown in Figure 8. It is a 2-dimensional diagram showing each trace in a horizontal line with each line consisting of the event found in the trace. The positioning of the events is according to an per event time stamp value with older events to the left and newer events to the right. The coloring and the shape of the events encode information about event data.

¹⁸ <http://www.processmining.org/online/dottedchartanalysis>

3 Design Options

After showing the basic ideas behind BPM, sonification, visualization and the implementation platform ProM this chapter gives an overview about the design options for a tool supporting process mining and process analysis using sonification on the ProM platform. In detail these are options for the input data, the visualization and sonification aspect and, as a concrete GUI should be built, the interaction design.

3.1 Event Logs

The prototype operates on business process process event logs. For the prototype only XES is relevant. The older MXML [16, p. 3f] was superseded by XES as described in [39, p.1] and the ProM website¹⁹. The general structure of XES is described in Section 2.1.1, for the prototype a decision has to be taken which type of data in log files should be supported. These data types can be found either in the standard or custom extensions.

3.1.1 XES Standard Extensions

There are seven standard extensions [17] where each extension defines one or more keys. These standard extensions have a fixed semantic. This includes also a fixed data type for the value of the key which allows for an easier interpretation.

Time Extension

This extension allows to store date and time for each event. The millisecond precision for the time value covers typical business process demands.

Listing 2: XES standard extensions: time

```
<event>
  <date key="time:timestamp" value="2015-06-19T12:55:22+02:00.000" />
</event>
```

Concept Extension

The concept stores two possible values for an event. *Name* stores the name of the event and *instance* stores an identifier of the activity instance that generated the event. *Name* is therefore interesting for the sonification as it might occur more than once within a trace or across a trace. For example the *Name* buy might occur with both lifecycle transition *start* and *complete* within one trace and this tuple might occur in each trace of the log.

¹⁹ <http://www.processmining.org/logs/mxml>

Listing 3: XES standard extensions: concept

```
<event>
  <date key="time:timestamp" value="2015-06-19T12:55:22+02:00" />
  <string key="concept:name" value="buy" />
  <string key="concept:instance" value="ActivityInstanceID123" />
</event>
```

Lifecycle Extension

Defines for each event the current lifecycle state. The extension allows to use a predefined lifecycle consisting of 13 transitions when the *model* attribute is set to *standard* on the log level. When this attribute is not set the concrete lifecycle transitions can be freely chosen. In any case the concrete values for transitions are likely a manageable set of values and their meaning is well-defined.

Listing 4: XES standard extensions: lifecycle

```
<event>
  <date key="time:timestamp" value="2015-06-19T12:55:24+02:00" />
  <string key="lifecycle:transition" value="start"/>
</event>
```

Organizational Extension

Allows to link a person (*resource*), its role (*role*) and its organizational group (*group*) to an event.

Listing 5: XES standard extensions: organization

```
<event xmlns="http://www.xes-standard.org/">
  <date key="time:timestamp" value="2015-06-19T12:55:24+02:00" />
  <string key="org:resource" value="John Doe" />
  <string key="org:role" value="case handler" />
  <string key="org:group" value="Accounting" />
</event>
```

Semantic Extension

This extension allows to link to an external ontology model on each level of an event log.

Listing 6: XES standard extensions: semantic

```
<event xmlns="http://www.xes-standard.org/">
  <date key="time:timestamp" value="2015-06-19T12:55:24+02:00" />
  <string key="semantic:modelReference" value="http://www.example.org/ontology/123" />
</event>
```

ID Extension

The ID extension provides unique identifiers in the form of UUID for elements of all levels of an event log.

Listing 7: XES standard extensions: ID

```
<event xmlns="http://www.xes-standard.org/">
  <date key="time:timestamp" value="2015-06-19T12:55:24+02:00" />
  <id key="identity:id" value="f81d4fae-7dec-11d0-a765-00a0c91e6bf6" />
</event>
```

Cost Extension

The cost extension allows to assign costs to an event. The amount is stored as floating-point number in the *total* attribute and the currency associated with this amount is stored in the *currency* attribute. There is also be to the possibility to further specify the costs of an event using one more meta attribute for *amount*, *driver* and *type*.

Listing 8: XES standard extensions: cost

```
<event xmlns="http://www.xes-standard.org/">
  <date key="time:timestamp" value="2015-06-19T12:55:24+02:00" />
  <string key="cost:currency" value="EUR" />
  <float key="cost:total" value="19.90" />
</event>
```

3.1.2 XES Custom Extensions

The process to define custom extensions is the same as for standard extensions where an example for the semantic extension is available in the specification [17, p.17]. Also as for the standard extensions the custom extension may contain data on the log, trace, event or meta level.

3.2 Sonification

3.2.1 Techniques

Section 2.2 shows the five basic types of sonification. The description there allows to derive requirements to identify which types may fit for the purpose of business process sonification.

- The amount of data required for the sonification.
- The parameterization options provided by the sonification technique.
- The number of input data dimensions that can be reflected by the sonification technique.
- The traceability for the sonification user how changes in the input data or the mapping influence the resulting sonification.

- The familiarity for the sonification user with the construction principles of the sonification technique.
- The feasibility for the sonification user to not only parameterize the sonification but also to create completely new mappings.
- The interoperability options for the sonification techniques.

Other aspects are context and reproducibility.

3.2.2 Technologies

The techniques shown above have to be applied using concrete technologies. Section 2.3 shows Sonification Sandbox, SuperCollider and xSonify as three different approaches for sonification tools. Sonification Sandbox and xSonify don't work with event logs and are therefore no options for an implementation technique. SuperCollider on the other hand would be flexible enough to generate sonifications based on event data but the integration of a stand-alone audio server in a ProM plugin would be a big hurdle as all other plugins don't depend on external processes. A possible implementation technique should be a library using the same Java Virtual Machine (JVM) that also executes the ProM main program itself and can be able to generate at least one of the sonification techniques mentioned above with a manageable effort.

Low Level JVM Sonification

ProM runs on the JVM so an obvious candidate may be using the sound capabilities of this platform directly. For the JVM this is called Java Sound API, with API standing for application programming interface. Information about this API is available in a programming guide²⁰ and in the JavaDoc for the packages *javax.sound.sampled* and *javax.sound.midi*²¹. The Java Sound API can handle sampled audio and MIDI input, output, generation and transformation. The two aspects sampled sound and MIDI are treated separately and both can deal with unbuffered and streamed playback. The abstraction level for both aspects is low. This may propose a limitation or additional effort on behalf of the client software especially in the MIDI context. For example the Java Sound API MIDI classes deal with single notes but don't provide an abstraction for musical concepts like chords so the client software is responsible for providing this abstractions. The Sonification Sandbox relies directly on the capabilities of the Java Sound API for generating MIDI output [43, p.1].

JFugue

JFugue²² is a music library running on the JVM. The JVM support makes it a good fit for a ProM plugin as it can be integrated into a plugin without requiring further configuration on the computer running ProM. JFugue is an abstraction over the MIDI capabilities of the JVM.

²⁰ https://docs.oracle.com/javase/8/docs/technotes/guides/sound/programmer_guide/contents.html

²¹ <https://docs.oracle.com/javase/8/docs/api/>

²² <http://www.jfugue.org/>

JFugue requires basic musical knowledge like to be able to name the notes in a scale and allows the user to build earcons. For constructing earcons the auditory attributes mentioned in [12] serve as a design guideline.

- *Timbre* is supported by using the different timbres of one of the 128 instruments as described in the MIDI specification.
- *Register* in general can be applied to voices and instruments. As MIDI, and therefore JFugue, is focused on instruments, the pitch of tones played on specific instrument should be chosen in a register that is typical for the instrument as otherwise the sound synthesis may deliver unsatisfactory results to the listener.
- *Pitch* can be set either by using the usual twelve semitones or by setting the frequency of a tone directly.
- *Rhythm* can be achieved by using a different tone duration and rests.
- *Intensity* is adjustable by setting a numeric volume value.
- *Spatial Location* is called panning in the context of MIDI and is mapped to a numeric value.
- *Timing* is supported by defining the start time of an earcon.

Beside these auditory attributes other features of JFugue can be useful for a earcon based sonification. *Voices* and *layers* allow to handle cases when multiple earcons are overlapping or to have a constant metronome like sound playing along with the event-related earcons. Another aspect is the playback speed of an earcon. JFugue supports to change the playback speed of successive notes (e.g. an earcon) without the need to change the definition of the earcon itself. Details to the older JFugue version 4 can be found in [25] and the documentation for version 5 (the version used for the tool) is available in [26].

After showing the fundamental capabilities of JFugue for an earcon based sonification an open aspect is how the tool user can specify an earcon without using the library APIs directly. For this purpose JFugue offers an external domain specific language (DSL) called *Staccato*. It allows to set all the auditory attributes defined above by having special commands that are easily accessible using an English or similar keyboard layout.

Listing 9: JFugue Staccato

```
// simple scale
c d e f g a b
// using rests and note durations
cq rq eq rq gq
// using voices and setting volume and panning by calling midi controller functions to
  influence the playback of a chord
V2 :Controller(7,127) :Controller(8,61) cmaj
```

Other JVM libraries

The Java Sound Resources²³ project provides resources for sound related Java programming and also a list of available Java sound libraries²⁴.

3.2.3 Mapping

Applying one or more techniques, for concrete technologies, for a concrete event log, is done in a mapping. A mapping has various aspects. First it defines which parts of the input data are considered for the sonification. So filtering out specific traces or events can also be seen as mapping. The selection of the input data requires an exact definition which aspects get mapped. Possible candidates are values available directly in the log like attribute values or derived values like a trace index. The input data can then be seen as input data units. The second aspect defines in which ways sonification rules are applied to input data units. It may be that input data units have independent sonification values or that sonification rules are applied to groups of input data units. There may be a function with input data unit(s) as input and a sonification value as output or it may be just an assignment. For both types the user of the software may have to define the function or assigned value while in another mapping scenario a user-mutable or user-immutable function or assigned value is proposed or dictated by the software. The sonification value may be in the form of a selectable option or offer stronger differentiation abilities when the user is allowed to make use of a DSL to declare the sonification value.

A requirement for the chosen mapping options is the practical feasibility in the GUI.

Some aspects of the mapping are not influenced by the input data. The playback speed and context information like a metronome influence the perception of the sonification but are not influenced by the input data.

3.3 Visualization

As shown in Section 2.4 for both the visualization and the sonification it has to be decided whether a data driven approach and / or a process driven approach should be followed. For the process driven approach an understanding of the underlying process is required. This requirement prevents it from being used in the process mining phase where no information about the underlying process is available.

3.3.1 Options

A concrete visualization can make use of various basic properties to visualize information.

²³ <http://www.jsresources.org/>

²⁴ <http://www.jsresources.org/links.html>

- Use different colors optionally with further distinguishing properties like a gradient.
- The size and shape of the visualization items.
- The principal ordering of visualization item in two or three dimensions and the scaling of input data for these dimensions.
- The overlapping of visualization items may result in covering underlying elements or the blending of multiple elements.
- The usage of text in the visualization.

When the basic properties of the visualization are defined and the visualization for a given input data and mapping can be drawn the simplest way is to show the whole visualization to the user. This may not always be desired depending on the information density in the visualization and the available screen size. The selection of parts of the visualization is done by zooming into the visualization. A change of the visualization over time requires an animation.

3.3.2 Technologies

The concrete visualization implementation technology may influence the practicality for supporting the previously defined visualization options. While support for low level drawing routines like lines and rectangles is sufficient to achieve all the options providing abstractions for complex operations like animations, drag and drop or zoom support may allow to support more of the more complex visualization options assuming a given amount of implementation time.

Another aspect not covered so far is the fact that the visualization technology is not only used for building the visualization itself but should also be able to display other UI elements like the sonification and visualization mapping controls. Examples for these controls are buttons, select boxes and sliders. A recent challenge is the support for HiDPI displays where these control elements need to be laid out using more pixels compared to a display with lower resolution.

ProM is built on the Java graphic library Swing which itself is based on the initial Java graphic library AWT. Swing provides many low to medium abstraction level 2d drawing elements and control widgets. There is a special ProM library²⁵ adapting common Swing widgets to the ProM look & feel. The support for HiDPI displays is poor²⁶ and often requires changes to the code of the running application.

SWT is a long time concurrent for Swing and also provides a roughly comparable feature set for drawing operations and control widgets. The main challenge is the integration²⁷ of SWT in a Swing application like ProM.

²⁵ <https://westergaard.eu/2011/02/slicker-widgets-for-prom/>

²⁶ <http://stackoverflow.com/questions/15659044/how-to-set-the-dpi-of-java-swing-apps-on-windows-linux>

²⁷ <https://gist.github.com/caprica/6890618>

JavaFX is the successor to Swing and provides more higher level drawing routines along with many control widgets. It supports HiDPI screens. As for SWT the main challenge is the integration²⁸ in a Swing centric application like ProM.

3.3.3 Mapping

The visualization mapping has to deal with similar question as the sonification mapping shown in Section 3.2.3. Visualization technologies are applied to event logs in the form of a mapping. As the main focus of the thesis is to explore the role of sonification in a multimodal approach it can be questioned if the visualization mapping needs to have the same complexity and flexibility as the sonification mapping. For example should the visualization mapping be completely independent of the sonification mapping. This would mean that changes made to the sonification mapping are not reflected in the visualization. An even stricter restriction would prohibit the user from choosing from visualization mapping values so that these values are chosen for the user by the tool itself. These kind of question leads to the next section where the interaction between sonification, visualization and their mapping is discussed.

3.4 Interaction Design

Interaction design has the purpose to define the behavior of the software to fulfill the need of the user interacting with this software [8, p.xxvii f]. The top level goal of the user is to perform process mining and process analysis. This goal can be split in many activities where for many activities multiple ways for goal achieving are imaginable and technically feasible. Specific to business process sonification is the fact that there exists no other tool for this purpose. The Sonification Sandbox and xSonify are the closest comparable sonification tools so some interaction and UI design decisions may be reused. As deciding on a concrete mapping for a log file is a non-trivial task the user may need several approaches with bigger changes in the beginning until some fine tuning for the last mapping modifications before the result is satisfying. This iterative steps need to be easily accessible in the UI. As the resulting mapping and the sonification require a considerable amount of effort the user may be interested in preserving the mapping and the resulting sonification and may also wish to do and present her work on more than one computer. This requires reproducible results independent of the operating system, screen resolution or sound hardware available on the different machines. Visual information seeking helps to deal with large event logs. Shneiderman [32, p.3] provides a list of seven tasks that are here now applied more concretely to the domain of business process sonification and may be applied to sonification and/or visualization.

1. Overview: Show the whole log.

²⁸ <https://docs.oracle.com/javase/8/javafx/interoperability-tutorial/swing-fx-interoperability.htm>

2. Zoom: Show parts of the log in a close-up.
3. Filter: Filter on trace or event level.
4. Details-on-demand: Show additional details for a log, trace or event on user demand.
5. Relate: Make relationships visible, for example events belonging to the same trace or events happening at the same time.
6. History: Undo and redo changes to the mapping and filtering.
7. Extract: Extract traces and events that satisfy a requirement.

All these tasks may be performed by the tool, by ProM or other external programs. This switching between programs may break the workflow of the user as external programs may make use of other interaction design patterns compared to the tool. But as the tool can not foresee all possible usage scenarios a convenient way to exchange data with other tools is required.

One aspect not considered by this information visualization discussions is the time aspect immanent in an event log and also a requirement for any kind of non-trivial sonification. How can the user be aware of the currently relevant time in the sonification and how can she switch and rewind or fast forward while playing the sonification.

4 Tool

While Section 3 describes requirements and design options, the concrete implementation can only support a subset of these options. This chapter has the main focus again on the research questions each of them targeted specifically in Sections 4.2, 4.3 and 4.5. Beside the description of the supported input data format in Section 4.1 the remaining sections cover interesting technical aspects of the tool implementation and the concrete design of the UI.

4.1 Event Logs

The tool doesn't support custom extensions in its current implementation. The main reason is that the standard extensions already cover many different data types. Another reason is the well understood semantic of these extensions. The practical evaluation of the sonification with use-cases, makes use of log files provided on the ProM homepage, that also require only the standard extensions.

The next decision has to be which standard extensions are supported. Here the *time* extension has special importance as sonification is a temporal medium and a temporal information in the input data allows for a straightforward mapping. The standard extensions *concept:name*, *lifecycle:transition*, *org:resource*, *org:role* and *org:group* are nominal values where the concrete values are easily understood and distinguishable by the tool user and therefore, are a good input for the sonification mapping. The other nominal values *concept:instance*, *semantic:modelReference* and *identity:id* are nominal values but not easily understood

without the context of e.g. the ontology and are neglected for the sonification. *cost:currency* would be a valid nominal value but the *cost:total* value of the cost extension is the only continuous value in the standard extensions and is therefore preferred.

All data discussed so far is mostly used from event properties. The other possible categories are traces where each trace is a single execution of the business process, the whole log containing all process executions, and the meta level for nested attributes. The meta level can be neglected as it is barely described in the XES standard definition [17] and not used by any of the logs evaluated for the use-cases. Sonifying properties on the log level would be possible but as sonification is a temporal medium and properties on the log level never change as long as only one log is considered the property values never change over time. On the trace level the *concept:name* property is used as nominal value to identify the business process instance. As on the event level, this can be used for the sonification if the listener is interested to distinguish different process instances.

Table 1 shows the data used for the sonification.

Level	Typ	Used
Log	-	-
Trace	Nominal	concept:name
Event	Time	time:timestamp
	Nominal	concept:name, lifecycle:transition, org:resource, org:role, org:group
	Continuous	cost:total
Meta	-	-

Table 1: Event log data used for sonification

4.2 Sonification

Based on the XES data described in the previous chapter the sonification is built by identifying matching techniques, technologies supporting these techniques and then specifying all the details how the input data influences the resulting sonification.

4.2.1 Techniques

The five sonification technologies introduced in Section 2.2 are reduced by identifying exclusion criteria. These criteria are based on the assumption that the sonification tool should be able to deal with the relevant XES event log data available in event logs ranging from around ten to event logs with hundred thousands of events.

- Audification requires too much data so it can not be applied to small to medium sized logs.

- Auditory Icons, in their simplest form, draw an analogy between a real world sound and a specific property of the log. As event logs may contain data from many domains, the tool can not provide all required sonifications. This would require the tool user to provide her own sound files. Problematic is also the lack of a general understanding how auditory icons can be parameterized. To sum it up a simple 1:1 mapping of event activity name to well-known auditory icon would be possible but supporting more input data dimensions or a richer collection of auditory icons would require a complex UI or require the tool user to bring along all the required sonifications herself.
- PMSon is inherently multidimensional [18, p.363] but as for auditory icons there is no general agreement how the the different dimensions affect the sound, a complex mapping for each parameter mapping would be required. This could lead to sonifications that are not intuitive and would require a complex UI for the two main tasks of a PMSon, data preparation and the mapping function. It is further complicated by the fact that the UI needs to be generic enough to handle a variety of event logs and support process mining and analysis. The input data for PMSon [18, p.365f] may contain multiple dimensions as found in event logs but quantitative data like temperature or price are expected while event logs consist mainly of nominal data that would require a more complex mapping. Event logs may not contain continuous data due to their statistic nature this would require a complex granular synthesis [18, p.365].
- For Model-Based Sonification it is unclear what the user actions creating the acoustic response could be in the context of business process event logs.

This leaves earcons as the preferred technique with support for multidimensional data. Especially the musical variant is familiar to many tool users and well supported by sonification technologies. While earcons may provide a good starting point for the tool, future expansion may bring in other sonification techniques, either together with earcons or as alternative working mode.

4.2.2 Technologies

SuperCollider can be integrated into the Scala ProM package with ScalaCollider²⁹, but it still requires a separate SuperCollider server, that must even run on the same host if stand-alone usage is required.

From the libraries running on the JVM using the low level Java Runtime Environment (JRE), MIDI oriented classes would require additional effort, to perform the transformation, from a high level user DSL describing the earcons to the API calls for building the sonification for single events and the whole log.

JFugue on the other hand focuses on high level musical concept that can be reused for building earcons. With Staccato it provides an DSL that can be used from describing a single note property like pitch to a complete log sonification. JFugue contains a player that allows to play and pause a running sonification

²⁹ <https://github.com/Sciss/ScalaCollider>

and jump to any point in the complete sonification. Events can be omitted by this player that indicate the current playback progress, a valuable help when the visualization should indicate the current sonification progress. As disadvantage JFugue supports only earcons out of the box but outputs the resulting sonification also as Java MIDI Sequence³⁰ object. This object could be played back with other low level Java sound objects allowing to integrate other sonification techniques.

4.2.3 Sonification Structure

The sonification is mainly determined by the events in the log. On the one hand the relative *time:timestamp* of an event is always mapped to its relative position in the complete sonification. The relative position in the log is determined by the time passed since the *time:timestamp* of the first event of the whole log. On the other hand all except one input data dimension shown in Table 1 are derived from event properties with the only exception being the *concept:name* found as trace property.

Checking again the auditory attributes mentioned in [12] this leaves timbre (instrument), register, pitch, rhythm, intensity (volume), and spatial location (panning) as possible output dimensions. Register and pitch are not explicitly distinguished in Staccato, and they are also not independent from each other, so they are grouped together here.

Volume and panning are a good output dimension fit for continuous input data as both can be described as numerical values. This allows for an intuitive mapping function from *cost:total* to panning. The lowest *cost:total* value is mapped to a playback only on the left output channel. While the *cost:total* increases the playback ratio shifts more and more to the right channel with the highest *cost:total* being played only on the right channel. For the in between input values a linear function is used. The concrete function used for the volume mapping is slightly more complex. A minimal output volume value of 0 when using the Staccato notation format would mean complete silence which is likely difficult to identify in the sonification. So the function for the volume mapping produces at least a value of 10, on the range from 0 to 127, so that the sonification is still identifiable.

Nominal values don't allow for an intuitive automatic mapping, for example it can be difficult to argue that a fictive checkout activity should sound like a drum (which one?) while a cancel activity is clearly a grand piano. The output dimensions for nominal values are grouped into units that easily comprehensible and allow for a manual mapping. When thinking in musical terms, a instrumental musical work is mainly characterized by the different instruments playing notes at specific points in time. So the instrument setting the timbre of the sonification is a good first candidate for nominal values. In this case pitch and rhythm remain and could serve as two output dimensions. But for a intuitive sonification, pitch and rhythm can be grouped together as melody, as this

³⁰ <http://docs.oracle.com/javase/7/docs/api/javax/sound/midi/Sequence.html>

is more easily identified by the sonification user. An alternative to instrument and melody originate in the special treatment of drum instruments in the MIDI channel 10 that is also reflected in the Staccato DSL. Instrument becomes drum instrument and also set the timbre. The pitch has no meaning in the context of drums but the rhythm alone serves as counterpart for the drum instrument for the sonification.

After identifying the four output dimensions panning, volume, instrument / melody and drum instrument / rhythm the next question is for their concurrent usage in the sonification of a single event. Volume and panning can be used together with the other categories, but as with *cost:total* there is only one continuous input data dimension available, both of them can't be used in the same log, at least if not a more complex mapping defines when volume and when panning should be applied in parallel. Instrument / melody and drum instrument / rhythm can be seen as exclusive alternatives. Sonifying a single event with either a single instrument and a single melody or a single drum instrument and a single rhythm allows for an understandable sonification. All other two, three and four way combinations like combining a drum instrument with a melody are either technically impossible or not helpful. A single category alone doesn't make a clearly defined sonification, for example a sonified event with only an instrument assigned is incomplete without a melody.

In total the sonification output dimensions consist of the mandatory exclusive alternatives instrument/melody and drum instrument/rhythm. Volume and panning are de facto exclusive alternatives but are not mandatory as the absence of an explicit volume or panning mapping simply results in the playback of all events at the same volume level or equally amplified on both channels.

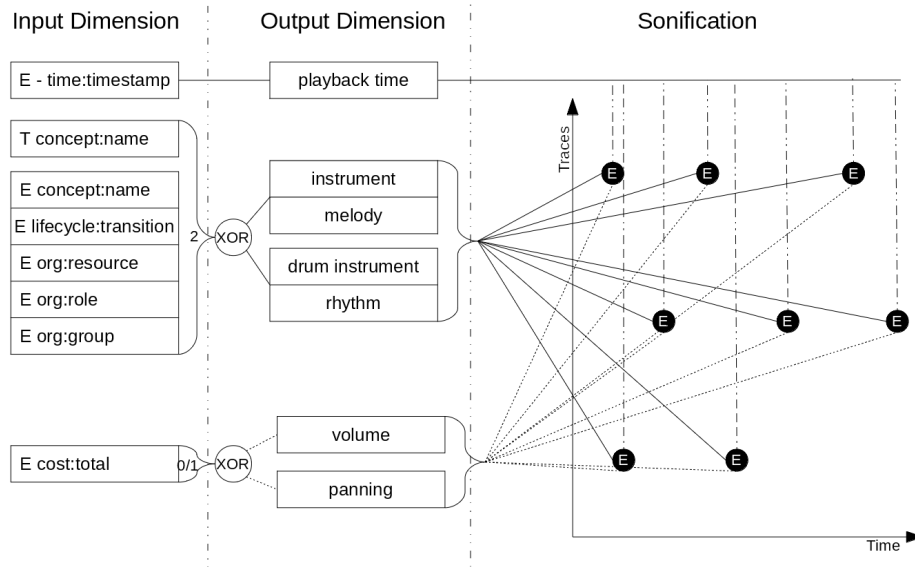


Fig. 9: Sonification Structure

To raise again the topic of context defined as “Context refers to the purposeful addition of non-signal information to a display” [33, p.1067]. In practice this is implemented by supporting a metronome. A metronome allows the listener to have a absolute reference point for the passing time even when the number of sonified events is slowly increasing or decreasing. These slow changes would have been otherwise difficult to detect as shown by [18, p.26]. A metronome or click track can take over the role of tick marks used in visualizations.

Another aspect is the absolute playback time of events. The relative playback among events is derived from their *time:timestamp* compared to other events. If the first, second and third event are evenly spread the break between the playback start of the first and second event is the same as the break between the playback start of the second and third event. In the example above the first event is played immediately, and the second event is played after two and a half hours. In practice this is not useful as the sonification of the first event takes only a few seconds at most. A naive approach is to just speed up the whole sonification by the factor 1000 resulting in a break of only 9 seconds between the first and second event. But this would also lead to a much too fast and therefore unidentifiable playback of the event sonification. A separate way to control the event sonification speed is required. The tool supports setting the playback speed defined as the speedup of the time between two event independently from the event speed, the speedup of a single event sonification. So the pauses between events can be reduced while the sonification of single events can be done in a way that different sonifications can be distinguished by the listener.

As shown in Figure 9 traces may overlap, leading to a mix up of event sonifications of different events. For specific use cases the listener is interested only in a disentangled playback of the trace sonifications. This is achieved in the tool by changing the relative playback time of the traces in a way that the trace ordering found in the log is used to change the playback time of the traces so that a later trace starts after an earlier trace finished, without any overlapping.

4.2.4 Default and Custom Mappings

The default instruments, melodies, drum instruments and rhythms help getting a useful sonification when the instrument and melody mapping are applied to the values of a specific XES standard extension. This may happen when a new XES file is loaded or when the mapping switches between XES standard extensions. This chapter describes also in detail the possible mapping values when the mapping is limited to a fixed set of options as it is the case for instruments and drum instruments (examples available in Sections A.2, A.3, A.4, A.5).

Instruments

The 128 General MIDI³¹ instruments are grouped into 16 categories. These categories are mostly related to the way an instrument produces its characteristic sound. For example the first group *Piano* produces its sound by vibrating strings while the third group *Organ* uses vibrating air. Other groups share a common way of sound production but for example *Guitar* and *Bass* have a different playing range, meaning the distance between the lowest and the highest pitch it can produce. Beside traditional instruments, including percussion instruments, ensembles, voice simulations, synthetic sounds and sound effects are available.

The 10 default instruments have been chosen so that they are easily distinguishable and can be combined with arbitrary melodies. For this reason some MIDI instrument types have been omitted.

- Voices have a specific role in popular music genres like pop or rock as they transport the lyrics of a song. Using them for the sonification might mislead listener to focus specifically on the trace(s) using a voice instrument.
- The sound effects are auditory icons as they are already familiar to the listener. For example a telephone ring or a gunshot are easily recognizable. Modifying these sounds by applying a different tone duration or pitch might lead to unfamiliar or unrecognizable results (example see Section A.1).
- The woodblock (MIDI-ID 116) as percussive instrument is used for the metronome. Therefore other percussive instruments are not used for the instrument part of the sonification. This allows the listener to easier distinguish between sonified events and context added to this sonification.

³¹ https://en.wikipedia.org/wiki/General_MIDI

The default instruments are shown below in Table 2, and they are also applied in these order to the alphabetically sorted values found in the currently chosen standard extension for the mapping. For example instruments with the numbers 1, 6, 11 and 16 are all taken from the MIDI group Piano. But as the first four instruments of this group (Acoustic Grand Piano, Bright Acoustic Piano, Electric Grand Piano, Honky-tonk Piano) sound similar another selection of four instruments of this group has been selected.

Number	MIDI-ID	MIDI-Group	Name
1	1	Piano	Acoustic Grand Piano
2	9	Chromatic Percussion	Celesta
3	41	Strings	Violin
4	57	Brass	Trumpet
5	30	Guitar	Overdriven Guitar
6	5	Piano	Electric Piano 2
7	10	Chromatic Percussion	Glockenspiel
8	45	Strings	Tremolo Strings
9	60	Reed	Oboe
10	32	Guitar	Guitar Harmonics
11	7	Piano	Harpsichord
12	14	Chromatic Percussion	Xylophone
13	46	Strings	Pizzicato Strings
14	61	Brass	French Horn
15	82	Synth Lead	Lead 2 (sawtooth)
16	8	Piano	Clavinet
17	15	Chromatic Percussion	Tubular Bells
18	47	Strings	Orchestral Harp
19	67	Reed	Tenor Sax
20	85	Synth Lead	Lead 5 (charang)

Table 2: Default instruments

Melodies

For the default melodies the following options would be available.

- Single notes, preferably from one scale.
- Chords played at once or as arpeggio.
- Short motifs as recognizable patterns played as single or polyphonic notes.
- More complex sonifications including different instruments, changes in volume or panning.

As the purpose is to identify different melodies, each possible melody should be recognizable when played alone or in short succession with other melodies. The pitch of single notes is difficult to identify for a listener without absolute hearing. Chords played at once contain the same information as chords played

as arpeggio but when played as arpeggio the overlapping probability is higher and this decreases the chance to identify the melody correctly. Using different instruments, volume and panning values may interfere with other mapping options. For example a melody mapping may include a note played exclusively on the right channel and other notes played using both channel. If the ordinal value mapping would now require the event to be sonified completely on the right channel the distinction between left and right channel in the melody mapping would be lost. Short motifs on the other hand can be identified easily and don't require features from other mapping categories. So for many scenarios short motifs would be good fit as default sonification. But for sonifications with much overlapping, or a high density of sonifications, short motifs become difficult to distinguish. For these reasons chords played as arpeggio are used as default values. The chords from the major scale are used with the characteristic succession of major, minor and diminished chords. Using the chords from the most popular scale has also the benefit that listeners familiar to western music don't need to learn to recognize new chords which should decrease the time needed to make use of a sonification using these chords.

Number	Melody	Number	Melody	Number	Melody
1	c5maj	8	c6maj	15	c6maj
2	d5dim	9	d6dim	16	d6dim
3	e5min	10	e6min	17	e6min
4	f5maj	11	f6maj	18	f6maj
5	g5maj	12	g6maj	19	g6maj
6	a5min	13	a6min	20	a6min
7	b5dim	14	b6dim	21	b6dim

Table 3: Default melodies

Drum Instruments

The drum instruments are the instruments available on MIDI channel 10 and are referred to by their JFugue name in the tool. All instruments in the way they are applied as default mapping, together with their better known name used in the MIDI specification³², are shown in Table 4.

³² <https://www.midi.org/specifications/item/gm-level-1-sound-set>

Number	JFugue-Name	MIDI-Name
1	ACOUSTIC_BASS_DRUM	Acoustic Bass Drum
2	BASS_DRUM	Bass Drum 1
3	SIDE_STICK	Side Stick
4	ACOUSTIC_SNARE	Acoustic Snare
5	HAND_CLAP	Hand Clap
6	ELECTRIC_SNARE	Electric Snare
7	LO_FLOOR_TOM	Low Floor Tom
8	CLOSED_HI_HAT	Closed Hi Hat
9	HIGH_FLOOR_TOM	High Floor Tom
10	PEDAL_HI_HAT	Pedal Hi-Hat
11	LO_TOM	Low Tom
12	OPEN_HI_HAT	Open Hi-Hat
13	LO_MID_TOM	Low-Mid Tom
14	HI_MID_TOM	Hi-Mid Tom
15	CRASH_CYMBAL_1	Crash Cymbal 1
16	HI_TOM	High Tom
17	RIDE_CYMBAL_1	Ride Cymbal 1
18	CHINESE_CYMBAL	Chinese Cymbal
19	RIDE_BELL	Ride Bell
20	TAMBOURINE	Tambourine
21	SPLASH_CYMBAL	Splash Cymbal
22	COWBELL	Cowbell
23	CRASH_CYMBAL_2	Crash Cymbal 2
24	VIBRASLAP	Vibraslap
25	RIDE_CYMBAL_2	Ride Cymbal 2
26	HI_BONGO	Hi Bongo
27	LO_BONGO	Low Bongo
28	MUTE_HI_CONGA	Mute Hi Conga
29	OPEN_HI_CONGA	Open Hi Conga
30	LO_CONGA	Low Conga
31	HI_TIMBALE	High Timbale
32	LO_TIMBALE	Low Timbale
33	HI_AGOGO	High Agogo
34	LO_AGOGO	Low Agogo
35	CABASA	Cabasa
36	MARACAS	Maracas
37	SHORT_WHISTLE	Short Whistle
38	LONG_WHISTLE	Long Whistle
39	SHORT_GUIRO	Short Guiro
40	LONG_GUIRO	Long Guiro
41	CLAVES	Claves
42	HI_WOOD_BLOCK	Hi Wood Block
43	LO_WOOD_BLOCK	Low Wood Block
44	MUTE_CUICA	Mute Cuica
45	OPEN_CUICA	Open Cuica
46	MUTE_TRIANGLE	Mute Triangle
47	OPEN_TRIANGLE	Open Triangle

Table 4: Default drum instruments

Rhythms

The default rhythms have all the same length which allows for more uniform sonification result, when differences in input data frequency are combined with differences in sonification duration. So all default rhythms have a total duration of a half note. For drum instruments the duration of a single note is more difficult to identify by the listener compared to the length of note in melody using a instrument like a piano. As a consequence the default rhythms in Table 5 don't require the listener to identify the length of a single note. For example the rhythm two quarter notes, and the rhythm quarter note followed by a eighth note, would make this single note length based identification necessary.

Number	Rhythm	Number	Rhythm	Number	Rhythm	Number	Rhythm
1	q q	4	q i i	7	q s s i	11	s s i s s i
2	i i i i	5	i i q	8	s q s i	12	i s s i s s
3	s s s s s s s s	6	i q i	9	s s q i	13	s i s s i s
				10	s s i q	14	i i s s s s
						15	i s s s s i

Table 5: Default rhythms

4.2.5 Complete Sonification Example

After explaining the sonification principles they are applied in a complete example. There are two executions of the business process resulting in two traces. In each case the two roles *merchant* and *customer* are performing the activities *offer* and *handover* for the *merchant* and *select* and *pay* for *customer*. The sell activity is the only one carrying the *cost:total* information.

Listing 10: Sonification: complete example XES

```
<?xml version="1.0" encoding="UTF-8" ?>
<log xes:version="1.0" xes:features="nested-attributes" xmlns="http://www.xes-
standard.org/">
  <extension name="Organization" prefix="org" uri="http://www.xes-standard.
    org/lifecycle.xesext"/>
  <extension name="Time" prefix="time" uri="http://www.xes-standard.org/time.
    xesext"/>
  <extension name="Concept" prefix="concept" uri="http://www.xes-standard.org
    /concept.xesext"/>
  <extension name="Cost" prefix="cost" uri="http://www.xes-standard.org/cost.
    xesext"/>
  <classifier name="Event Name" keys="concept:name"/>
  <trace>
    <string key="concept:name" value="Case 1"/>
    <event>
      <date key="time:timestamp" value="2015-05-01T08:00:00.000+01:00"/>
      <string key="concept:name" value="offer"/>
```

```

        <string key="org:role" value="merchant"/>
    </event>
    <event>
        <date key="time:timestamp" value="2015-05-01T08:00:02.000+01:00"/>
        <string key="concept:name" value="select"/>
        <string key="org:role" value="customer"/>
    </event>
    <event>
        <date key="time:timestamp" value="2015-05-01T08:00:04.000+01:00"/>
        <string key="concept:name" value="handover"/>
        <string key="org:role" value="merchant"/>
    </event>
    <event>
        <date key="time:timestamp" value="2015-05-01T08:00:06.000+01:00"/>
        <string key="concept:name" value="sell"/>
        <string key="org:role" value="customer"/>
        <float key="cost:total" value="9.90"/>
    </event>
</trace>
<trace>
    <string key="concept:name" value="Case 2"/>
    <event>
        <date key="time:timestamp" value="2015-05-01T08:00:01.000+01:00"/>
        <string key="concept:name" value="offer"/>
        <string key="org:role" value="merchant"/>
    </event>
    <event>
        <date key="time:timestamp" value="2015-05-01T08:00:03.500+01:00"/>
        <string key="concept:name" value="select"/>
        <string key="org:role" value="customer"/>
    </event>
    <event>
        <date key="time:timestamp" value="2015-05-01T08:00:05.000+01:00"/>
        <string key="concept:name" value="handover"/>
        <string key="org:role" value="merchant"/>
    </event>
    <event>
        <date key="time:timestamp" value="2015-05-01T08:00:08.000+01:00"/>
        <string key="concept:name" value="sell"/>
        <string key="org:role" value="customer"/>
        <float key="cost:total" value="14.90"/>
    </event>
</trace>
</log>

```

The first step is to define which of the four information types are mapped to which sonification types. *time:timestamp* is always mapped to the timestamp when an event is played but the other options can be set by the user of the tool. Therefore the sonification shown here is only one of many possible options how

the log could be sonified depending on the information the listener is interested in.

- the trace name is mapped to instrument and shape
- *concept:name* as literal value is mapped to melody and color
- *cost:total* as continuous value is mapped to volume

This mapping settings result in the following staccato music string.

Listing 11: Sonification: staccato example

```
T120
V0 :Controller(7,127) @0.0 I[Piano] d5min
V1 :Controller(7,127) @0.5 I[Celesta] d5min
V2 :Controller(7,127) @1.0 I[Piano] e5min
V3 :Controller(7,127) @1.75 I[Celesta] e5min
V4 :Controller(7,127) @2.0 I[Piano] c5maj
V5 :Controller(7,127) @2.5 I[Celesta] c5maj
V6 :Controller(7,10) @3.0 I[Piano] f5maj
V7 :Controller(7,127) @4.0 I[Celesta] f5maj
```

Besides the direct result of the mapping volume (e.g. *:Controller(7,69)*), instrument (e.g. *I[Piano]*) and melody (e.g. *d5min*) other information is contained in the Staccato music string. T120 sets the tempo in which a melody is played in beats per minute. In this example *d5min* uses the default duration of 1 quarter and at 120 bpm each note is played for 0.25 seconds resulting in a total length of 0.75s for the complete melody. V0 to V7 are the voices used to allow for polyphonic event sonification. *@0.0*, *@0.5* etc. define when an event sonification is played. The first event is played at time 0 while the next event happens 1.0 seconds after the first event. These 1.0 seconds need to be adjusted for the current playback speed resulting in a event playback start time of 0.5. The event sonifications are sorted chronologically over all traces. The complete sonification settings are accessible by pressing the *c* button in the tool and are shown below.

Listing 12: Sonification: example mapping details

```
CostTotal->
Volume
    9.9=10
    14.9=127
-
TraceName->
Instrument
    Case 1=Piano x
    Case 2=Celesta x
Shape
    Case 1=Square x
    Case 2=Circle x
ConceptName->
Melody
    handover=c5maj x
```

```

offer=d5min x
select=e5min x
sell=f5maj x
Color
    handover=red x
    offer=green x
    select=blue x
    sell=pink x
traces=all
speed=1.0
eventSpeed=1.0
duration=-
eventDurationSpeed=-
metronome=false

```

The resulting sonification is shown as sheet music in Figure 10 but is missing the volume adjustments for the last two notes.

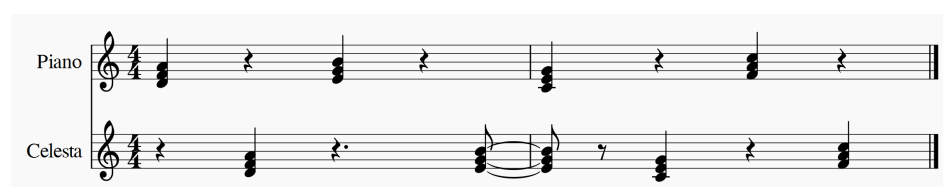


Fig. 10: Score of sonification example

The resources available for this simple example are

- sonification_example.xes (the input log)
- sonification_example.son (a project containing the log and the mapping, exported by the tool)
- sonification_example.midi (sonification as MIDI, exported by the tool)
- sonification_example.wav (sonification as audio stream, exported by the tool)
- sonification_example.flac (sonification as lossless compressed audio stream, converted from sonification_example.wav)
- sonification_example.mscz (MuseScore³³ 2.03 project used for Figure 10)

4.3 Visualization

The visualization part of the multimodal tool has two main purposes. For one thing it allows to get an overview about the data and for another thing it should visualize the chosen sonification mapping settings. The basic principles of the

³³ <https://musescore.org/>

visualization are taken from the ProM dotted chart visualization described above (2.7.2). The user gets an overview how many events happen in which trace and when they occur. Color and shape are used to encode information about the event standard extension attribute value found in an event. The basic representation of an event is a small black dot. This black dot changes to a colored shape only when the mandatory sonification categories are mapped for attribute values contained in the event. The reasoning behind this approach is that a melody without an instrument and vice versa is not a valid sonification and the focus for users of the visualization are the events that are actually sonified. All shapes including the black dot for not sonified events and some example colors are shown in Figure 11.

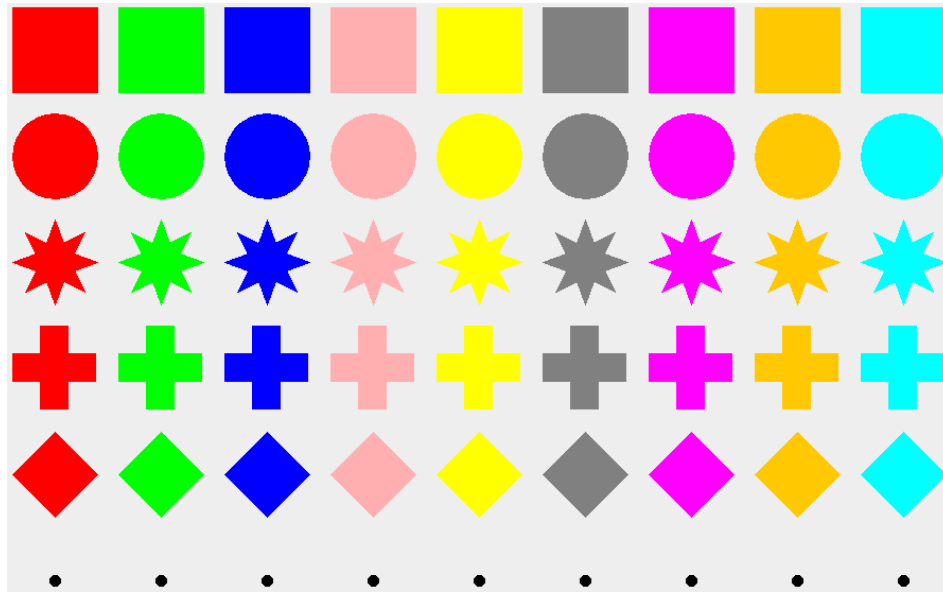


Fig. 11: Visualization colors and shapes

The five different shapes (square, circle, star, plus and rhombus) have been chosen to be easily distinguishable and are horizontally and vertically symmetric. These symmetries allow the viewer to identify an obvious and precise center of the shape. This is important especially in the horizontal dimension as the horizontal center of the shape indicates the time when the underlying event is happening. The vertical center indicates the trace to which the underlying event belongs. The colors have been taken from predefined constants in the *java.awt.Color* class. As the background is gray (RGB 238/238/238) and the dots for not sonified events are black some lighter greys, white and black have been omitted from the used constants to keep the colors easily distinguishable and

avoid the usage of the same color for multiple purposes. The available shapes are limited to the shapes shown in Figure 11 but the colors can be set for each mapping value to a custom value with a color picker.

Another aspect of the visualization is a way to give feedback about an ongoing sonification. A progress indicator shows the actual playback time so that the user can know which event is sonified while the sonification is running. On the other hand many configuration options of the dotted chart visualization are missing as the visualization has only the role to help understanding the sonification and changing the visualization without changing the sonification is not intended.

The continuous value sonification types volume and panning have no influence on the visualization. This is caused by the problem of encoding more than two dimensions of information in the event visualization. As volume and panning are optional it is not so important to reflect their values in the visualization.

4.4 Sonification and Visualization Overview

As a reference, Table 6 and Table 7 show the supported input data and the supported mapping options with their value ranges and their degree of customizability.

Toool Support	Mapping	XES	Data/Mapping Type
yes	Trace Name	concept:name	Literal
	Event Time	time:timestamp	Timestamp
	Event Name	concept:name	Literal
	Event Lifecycle State	lifecycle:transition	
	Event organizational Resource	org:resource	
	Event organizational Role	org:role	
	Event organizational Group	org:group	
	Event Cost	cost:total	Continuous
no	Log/Trace/Event/Meta Ontology Concept	semantic:modelReference	Literal
	Log/Trace/Event/Meta Identity	identity:id	ID
	Trace Cost	cost:total	Continuous
	Trace/Event Currency	cost:currency	Literal
	Meta Cost Amount	cost:amount	Continuous
	Meta Cost Driver ID	cost:total	Literal
	Meta Cost Type	cost:type	Literal

Table 6: Supported XES concepts

Types	Sonification		Visualization		Assignment required
Timestamp	playback time	-	position x-axis		implicit
Nominal	instrument	fixed set of 128 MIDI instruments	shape	fixed set of shapes	yes
	melody	free text JFugue Staccato	color	24 bit RGB	
Continuous	volume	auto-mapped coarse MIDI volume	-		no
	panning	auto-mapped coarse MIDI pan	-		

Table 7: Sonification and visualization mapping

4.5 Interaction Design

Six out of seven tasks for information seeking tasks are supported by the tool

1. Overview: The whole log is accessible in the visualization by using the *fit to page* zoom option in the main visualization and always showing the whole log in the smaller visualization overview. For the sonification part the whole sonification can be played back. An overview, showing the whole sonification in its Staccato form as text, allows the user to look for interesting patterns in the text.
2. Zoom: Parts of the log can be inspected by zooming in the main view of the visualization. The zoom operates only on the y-axis the time axis but by filtering the traces the data shown on the x-axis can also be reduced for the visualization. For the sonification zooming in requires reducing the playback speed so that the listener has more time to process the sonification. The jump to functionality together with the progress are required to (re)listen to specific parts of the sonification.
3. Filter: Filtering on the trace level removes the traces from the visualization and sonification. Filtering on the event level is supported for the mandatory mapping types and results in omitting filtered events from the sonification and showing them only as black dots in the visualization.
4. Details-on-demand: The visualization shows the trace belonging, the event time and the mandatory mapping types while the sonification is also influenced by optional mapping types with the trace belonging only shown when one of the two mandatory mapping spots is assigned to it. The details on demand are available only for the the visualization as it is easier to select an event in the visualization than to do the same in a running sonification. The details show the corresponding event sonification together with the complete XES as XML.
5. Relate: Relationships can be made by assigning mandatory and optional mapping types. The used mandatory mapping types are reflected in the

sonification and visualization with the sonification also considering the optional mapping types.

6. History: Undo and redo is currently not explicitly supported but the user can un- and redo all changes to the mapping as long as she remembers what she has done. The mapping, together with the log it operates on, can be saved at any time so that saving and loading projects can be used for keeping snapshots.
7. Extract: Is not supported.

Another type of interaction design supported by the tool helps the user to become familiar with the sonification parts. A prelisten functionality for the mandatory sonification type complements the currently edited type with its required counterpart to make a short prelisten example possible. For example when choosing the instrument a fixed melody is played for the chosen instrument for prelisten sonification. When listening to the complete sonification jumping back and forth in the sonification allows to listen repeatedly to relevant parts of the sonification.

As the sonification mapping requires entering Staccato, validation support provides quick feedback for invalid sonifications.

4.6 Software Architecture and Implementation

Implementing the standalone and plugin version of the tool faces a few challenges and restrictions described here.

As every ProM package the prototype plugin has to define the input(s) and output(s) it generates. Starting point for each possible usage of the prototype plugin is the import of the event log. The input for the prototype plugin is a parsed event log, an instance of the class `XLog`³⁴. Therefore all previous plugin executions must have a `XLog` as output of the last plugin used prior to the multimodal plugin, so that this output can be used as input for the prototype plugin. The prototype plugin itself offers no output that can be reused by other ProM plugins or itself, its only a visualization plugin as far as ProM is concerned. For example changing the input `XLog` by deselecting specific traces does not result in a new trace that is made accessible as `XLog` for later usage as input for the prototype or other plugins. Also the sonification and visualization mapping for a specific log are not defined as output of the plugin. Reopening a `XLog` in the prototype always starts with the default settings for sonification and visualization.

The sonification is technically a MIDI playback, as described in more detail in the JFugue subsection 4.2.2. It can be exported as audio either as MIDI or the MIDI converted to an audio bitstream. The later one should be used to get final, repetitive results as MIDI defines only music events and their properties and not the concrete sound synthesis that is applied to get audio bitstream. The concrete results of the sound synthesis then depend on the Java MIDI environment used on the computer. To get reproducible results when using the same

³⁴ `org.deckfour.xes.model.XLog`

mapping for an event log the synthesizer has to use fixed sound synthesis format. For MIDI a sample-based synthesis is preferred where instrument audio samples are used to produce a rich sound. There are many technical implementations even when focusing only on the JVM. With Java 7³⁵ support for the widely used SF2 (SoundFont³⁶) format was added allowing to choose between a wide selection from free to paid implementations. For the evaluation in the use cases a freely available general purpose SoundFont is used³⁷³⁸.

The visualization is based on basic Java Swing and AWT classes. As there are potentially thousands of events that need to be visualized, and the progress indicator is constantly moving, recalculating and redrawing all the events and the progress indicator would require too long to keep a constant frame rate of 30 frames per second. Chet et al. [7, p.134ff] propose to render complex graphics only on demand. In this context the events are only rendered once and a *java.awt.image.BufferedImage* buffers this image in memory so only the finished rendering and the progress indicator at the current position have to be drawn with each new frame. The buffered image contains all events even if zoomed in and it has to be recalculated on some sonification and visualization mapping changes, trace filtering and resizing the ProM window.

The plugin UI contains many panels, buttons, tabs, sliders etc. To give them the typical ProM look & feel the Slickerbox³⁹ library is used that provides specialized UI elements and allows to customize the look & feel of existing Swing elements.

Performance is a critical point, not only for smooth animations of the running visualization. Applying timers⁴⁰ to measure both the rate and the duration at which specific code blocks are called allows to identify the following expensive operations.

1. Parsing of the XES file from a file to a in memory representation of the whole log using OpenXES.
2. Based on the OpenXES log object the default visualization and sonification settings are applied.
3. The buffered image is calculated for the visualization.
4. Generating the Staccato DSL from the log and the default mappings.
5. Generate the Java MIDI sequence object⁴¹ using the Staccato DSL instance as input.
6. Generate actual MIDI file from the sequence object.

To test the actual performance of these calculations test XES files using random data are generated⁴². These files use all relevant XES extensions, have

³⁵ https://docs.oracle.com/javase/8/docs/technotes/guides/sound/enhancements_7.html

³⁶ <https://en.wikipedia.org/wiki/SoundFont>

³⁷ <http://www.schriancollins.com/generaluser.php>

³⁸ https://dl.dropboxusercontent.com/u/8126161/GeneralUser_GS_1.47.zip

³⁹ <http://code.deckfour.org/slickerbox/>

⁴⁰ <http://metrics.dropwizard.io/3.1.0/getting-started/#timers>

⁴¹ <http://docs.oracle.com/javase/7/docs/api/javax/sound/midi/Sequence.html>

⁴² performance_1k.xes, performance_10k.xes, performance_100k.xes

50 events per trace and the last event starts one hour after the first event. To test the scalability of each of the calculations the total number of traces varies. For example 1.000 total events requires 20 traces at 50 events and 10.000 total events require 500 traces at 50 events. Table 8 shows the execution times for three different XES file sizes. These concrete numbers have been calculated on a specific configuration⁴³ and especially the shorter (<100ms) execution times may vary greatly due to JVM hotspot optimizations and other typical problems when measuring the performance of desktop software. But as for example the first line *XES parsing* shows the results are mostly plausible, parsing 100.000 instead of 10.000 requires around 6 times the calculation duration which is below a linear increase. But the sequence calculation as fifth task slows down by the factor 1000 for 10 times the events. More than three minutes for 100.000 events makes going beyond this event count impracticable. A closer look in the code reveals the JFugue method *org.jfugue.player.Player#getSequence()* as being responsible for the execution time. Therefore the scalability of the prototype in regard to huge event logs is limited at the moment by the sonification library used.

#	Task	1.000 events	10.000 events	100.000 events
1	XES parsing	19	70	1588
2	calculating default mappings	13	207	1579
3	calculate buffered image	9	116	1548
4	generate Staccato	3	33	659
5	sequence	43	3411	194551

Table 8: Performance measurements

As the input of the prototype is only a XLog instance and the output is not specific to ProM a standalone version of the prototype can be created without much additional effort. As shown in the screenshots showing the ProM plugin version in Figure 12 and standalone version in Figure 13 the main difference are the missing ProM controls. An additional button next to the volume control was added to allow opening a XES file from the file system. The other functionality is identical to the ProM plugin version.

⁴³ Core i6600k, Windows 10, JDK 1.8.72

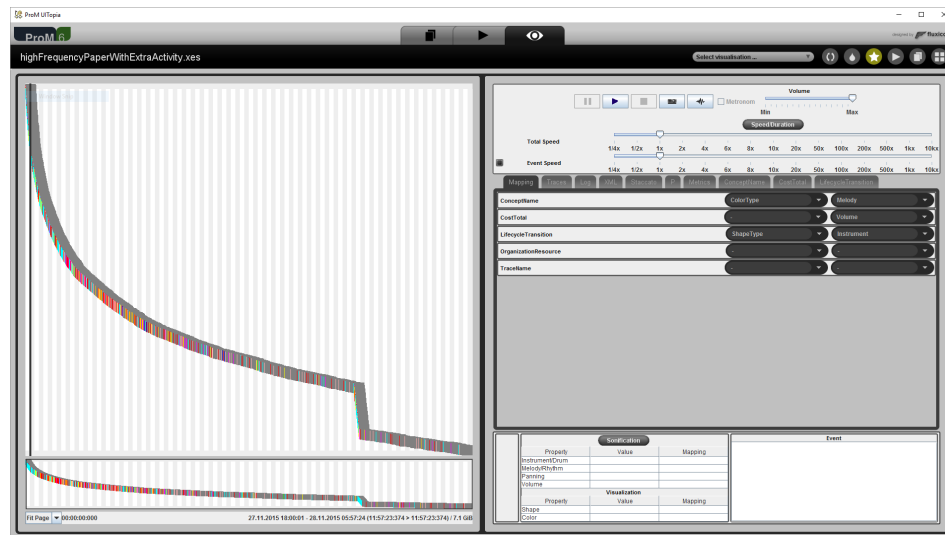


Fig. 12: ProM version

4.7 User Interface

The UI description in this chapter shows the concrete implementation of the sonification mapping, the visualization and all other interaction related aspects like filtering log files. The sonification itself is also part of the UI. Concrete examples are shown in the Section 5. All screenshots show the standalone variant of the tool as this variant has a few additional options compared to the ProM plugin.

4.7.1 Overview

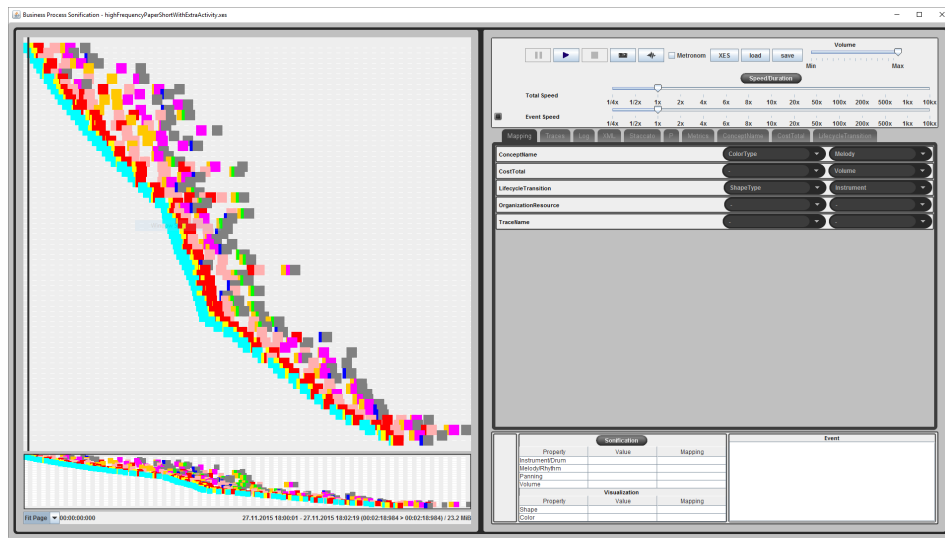


Fig. 13: UI overview

The UI shown in Figure 13 is visually split into four main areas. The visualization to the left, the player on the top right, the sonification mapping on the middle right and the properties on the bottom right. As the properties serve only as an additional information for the visualization they are described along with the visualization.

4.7.2 Visualization

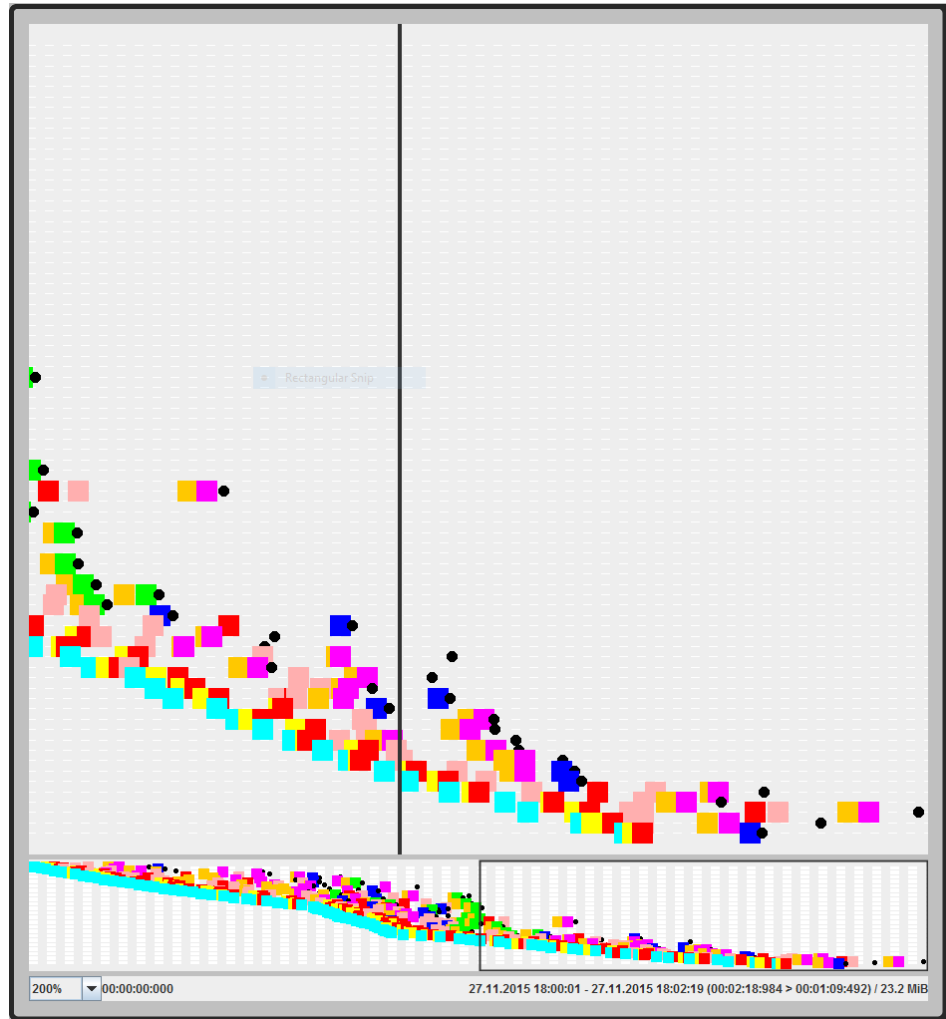


Fig. 14: Visualization

The visualization shown in Figure 14 is split in a detail view on the top, an overview below the details and additional settings and information at the bottom. On the bottom left, the zoom level for the detail view can be set. Figure 13 shows the 100% zoom setting resulting in identical visualization in the detail and overview. When zooming in as shown in Figure 14 the overview shows which part of the total visualization is currently visible in the detail view. The last row shows the current zoom factor for the visualization. The other numbers

are related to the sonification playback. From left to right: the current playback time relative to the first event, start and end of the log, the log duration and the actual playback duration that may be shorter or longer depending on the speedup factor. At the bottom right the file size when saving the sonification as audio bitstream.

	Sonification			Event
	Property	Value	Mapping	<event> <string value="system" key="org.resource"/> <string value="complete" key="lifecycle.transition"/> <string value="Process Availability" key="conceptname"/> <date value="2015-11-27T18:01:47.590+01:00" key="time.timestamp"> </date> </event>
	Drum Instrument	complete	ACOUSTIC_BAS...	
	Rhythm	Process Availabi...	s q s i	
	Panning			
	Volume			
	Visualization			
	Property	Value	Mapping	
	Shape	complete	Square	
	Color (RGB)	Process Availabi...	255/200/0	

Fig. 15: Properties

The properties in Figure 15 show the sonification and visualization details for an event on the left side and the event as XES on the right side. The event visualization is shown and the event sonification can be played when clicking on the *Sonification* button.

4.7.3 Sonification Mapping

Core Mapping

Mapping	Traces	Log	XML	Staccato	P	Metrics	ConceptName	CostTotal	LifecycleTransition
ConceptName							ColorType		Melody
CostTotal									Volume
LifecycleTransition							ShapeType		Instrument
OrganizationResource									
TraceName									

Fig. 16: Global mapping

The main mapping is shown in Figure 16. It allows to set mapping for the sonification in the right column and takes care that only valid sonification mapping combinations can be selected. The visualization in the middle column can't be manually selected as it is selected along with the mandatory sonification mapping types.

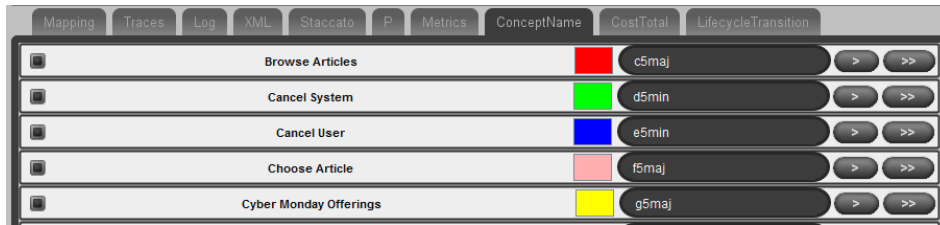


Fig. 17: Melody mapping

Figure 17 shows an example for mapping nominal input data to a melody and color. The checkbox to the left allows to filter events for the given value shown in the next column. The color can be chosen with a color picker. The melody can be entered as Staccato string that gets automatically validated. The playback buttons on the right allow to prelisten to the melody with a fixed instrument either in a standard tempo (120 bpm) or at the currently chosen event speedup.

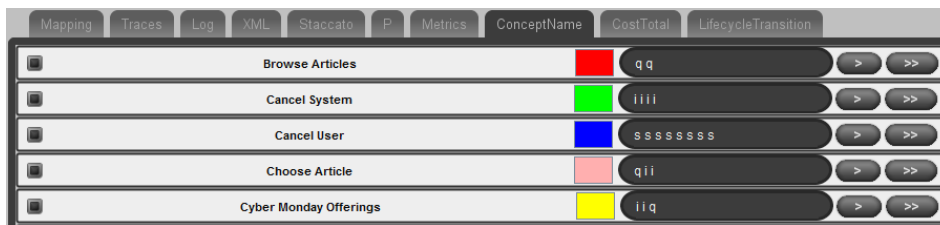


Fig. 18: Rhythm mapping

Figure 18 works in the same way as the melody mapping shown before. The only difference is that the text field expects a Staccato string containing a rhythm.

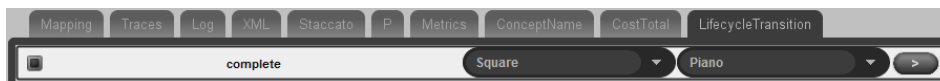


Fig. 19: Instrument mapping

Figure 19, again, shows a mapping for nominal input data. For instruments the user can select an instrument from a list of available instruments. The visualization shape can be chosen among a list of options. Filtering works the same

way as for drum and rhythm. Prelisten is only supported for a standard tempo (120 bpm) where a major scale is played for the chosen instrument.

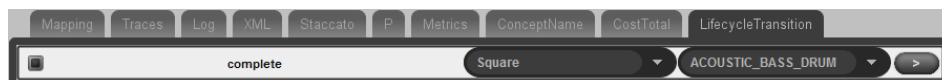


Fig. 20: Drum instrument mapping

Figure 20 shows the drum instrument mapping working the same way as the instrument. Clicking on the prelisten button plays a short rhythm for the chosen drum instrument.

Mapping	Traces	Log	XML	Staccato	P	Metrics	ConceptName	CostTotal	LifecycleTransition
Cost					Volume				
33.52925876150222					10				
58.14684215027063					35				
58.920774656407914					35				
61.15194350187837					38				
65.093200383953					42				
66.11959321096128					43				
75.46536101139802					52				
77.50040151374216					54				
79.64494614377182					56				
80.73717867868903					57				
81.09023064963716					58				

Fig. 21: Volume mapping

Mapping	Traces	Log	XML	Staccato	P	Metrics	ConceptName	CostTotal	LifecycleTransition
Cost					Panning				
33.52925876150222					0				
58.14684215027063					27				
58.920774656407914					28				
61.15194350187837					30				
65.093200383953					34				
66.11959321096128					35				
75.46536101139802					45				
77.50040151374216					48				
79.64494614377182					50				
80.73717867868903					51				
81.09023064963716					52				

Fig. 22: Panning mapping

Figure 21 and 22 show volume and panning mapping. In both cases it is just a list of sorted input data in the left column and the corresponding volume or panning MIDI value in the right column.

Mapping Helpers

Mapping

Traces

Log

XML

Staccato

P

Metrics

ConceptName

CostTotal

LifecycleTransition

Select All

Deselect All

☐	Trace 1 (0001) from 27.11.2015 18:00:01 to 27.11.2015 18:00:18 (00:00:17:959) with 7 events
☐	Trace 2 (0002) from 27.11.2015 18:00:02 to 27.11.2015 18:00:15 (00:00:13:357) with 7 events
☐	Trace 3 (0003) from 27.11.2015 18:00:03 to 27.11.2015 18:00:21 (00:00:18:336) with 8 events
☐	Trace 4 (0004) from 27.11.2015 18:00:04 to 27.11.2015 18:00:23 (00:00:19:104) with 7 events
☐	Trace 5 (0005) from 27.11.2015 18:00:05 to 27.11.2015 18:00:36 (00:00:31:532) with 7 events

Fig. 23: Trace selection

The trace list in Figure 23 informs about the trace index, the trace name, the first and last event time stamp along with the resulting trace duration and the number of events available in the trace.

```

<log xmlns:xes="http://www.openxes.org/xes" xes:features="nested-attributes" xes:version="1.0">
  <string value="highFrequencyPaperShortWithExtraActivity.xes" key="conceptname"/>
  <trace>
    <string value="0001" key="conceptname"/>
    <event>
      <string value="customer" key="org.resource"/>
      <string value="complete" key="lifecycle.transition"/>
      <string value="Start Page" key="conceptname"/>
      <date value="2015-11-27T18:00:01.000+01:00" key="time.timestamp"/>
    
```

Fig. 24: Log file

The XML tab shown in Figure 24 shows the complete log after it was parsed by the OpenXES parser. The main difference to a possible XML file input is the omitting of comments and the formatting of the XML with fixed values for indentation.

```

T120
V0 :Controller(7,127) @0.0 I[Piano] d6min
V1 :Controller(7,127) @0.514 I[Piano] d6min
V2 :Controller(7,127) @0.9625 I[Piano] g5maj
V3 :Controller(7,127) @1.025 I[Piano] d6min
V4 :Controller(7,127) @1.2385 I[Piano] g5maj
V5 :Controller(7,127) @1.538 I[Piano] d6min
V6 :Controller(7,127) @1.585 I[Piano] c5maj

```

Fig. 25: Staccato

The Staccato for the current log and mapping is available on the Staccato tab shown in Figure 25. The actual Staccato used for the sonification is formatted for better readability as each sonification event is put on a single line.

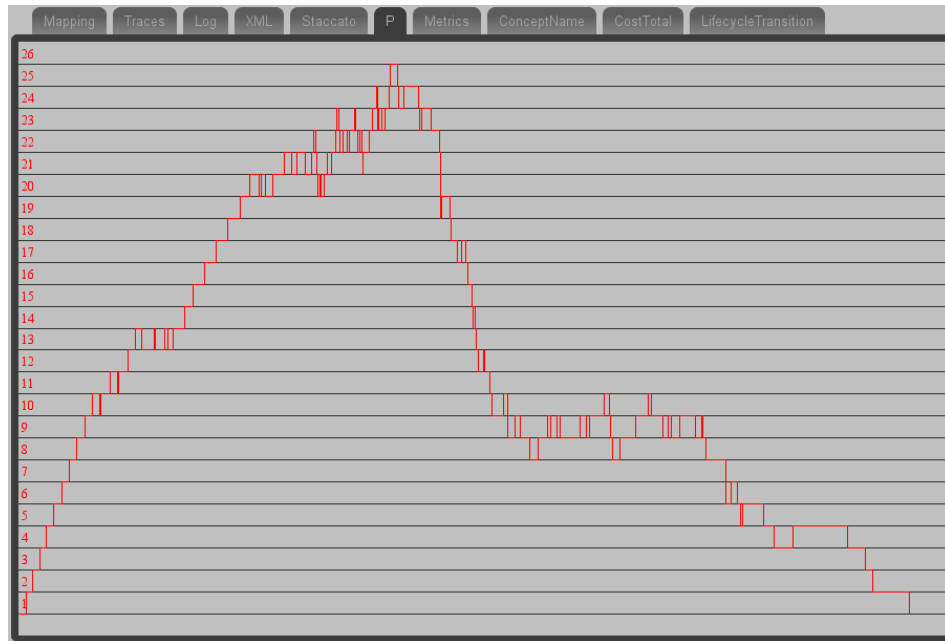


Fig. 26: Parallelism

The parallelism count in Figure 26 shows the time on the x-axis and the overlapping trace parallelism on the y-axis.

Other

Mapping	Traces	Log	XML	Staccato	P	Metrics	ConceptName	CostTotal	LifecycleTransition
Refresh Reset									
Function	#	Min	Max	Mean	StdDev	Median			
V-calculateBuffered...	6	4	164	74	69	24			
SU-defaultSettings	3	9	341	141	151	95			
parseXES	3	14	193	97	83	144			
V-paintComponent	8	2	131	26	44	8			

Fig. 27: Metrics

Figure 27 is only used for debugging when the application doesn't respond in a timely manner so that the user can inspect what is causing the delay.

4.7.4 Player

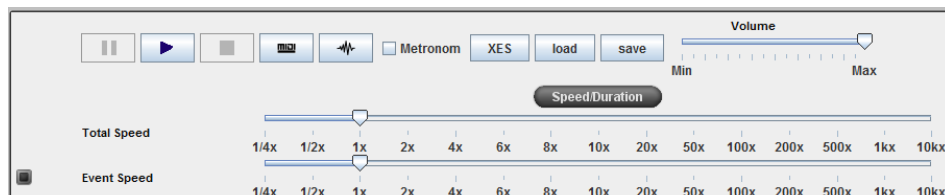


Fig. 28: Player

Figure 28 shows the playback controls and the player operating in *speed* mode. The controls in the top row from left to right start with pause, play, stop, MIDI export, audio bitstream export. The metronome can be enabled with the next checkbox but it gets disabled for longer logs as the resulting sonification Staccato would be too big. The next button allow to load a XES file and load and save a project containing the XES file and a mapping. The volume button controls the global volume of the sonification.

Below are the controls for the speed setting the speedup for the time passing between event sonifications. The event speed can be controlled independently from the total speed when the checkbox in the last row is selected.

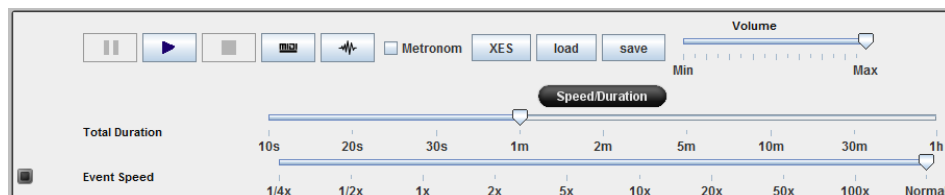


Fig. 29: Player duration mode

Figure 29 shows similar controls as the previous Figure but the speedup controls work now in *duration* mode. This allows to set the total duration for the complete sonification. The event speedup in the last row is relative to the speedup derived from the chosen duration.

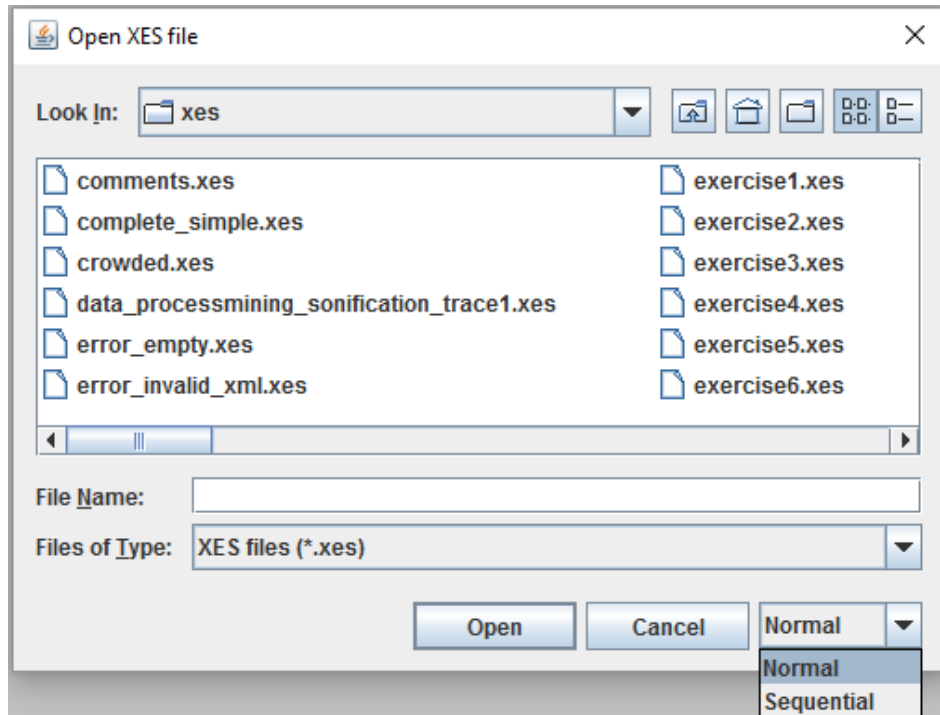


Fig. 30: XES loading modes

Figure 30 shows the loading dialog for a XES file. It is default loading dialog with the extra ability to disentangle the traces in the log when loading it in the *sequential* mode. If loading the XES file fails the warning dialog shown in Figure 31 is displayed.

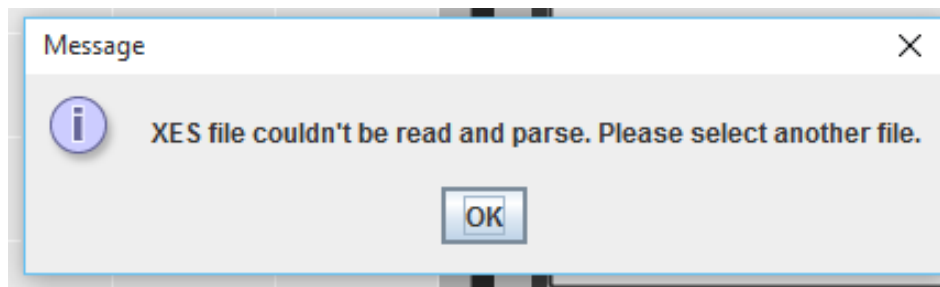


Fig. 31: XES loading error

5 Use-Cases

Following the discussion of the sonification and visualization concepts and how they are implemented as a ProM plugin in Sections 3 and 4 the applicability is tested with concrete use-cases. The chosen use-cases are tackling process mining in Section 5.1 and process analysis in Section 5.2. The overview in Table 9 shows that the repair example is used for both process mining and process analysis while an additional log file is used for process analysis. This focus on process analysis is caused by the better suitability of sonification to identify deviations from a well known behavior than to identify this behavior in the first place.

Type	Log	Topics
Process mining	repair example	control-flow perspective, case perspective, organizational perspective
Process analysis	repair example with added anomalies	increased throughput
		changed task executor
	webshop	longer execution time
		changed probabilities
		activity burst

Table 9: Use-cases overview

5.1 Process Mining

A single use case deals with the discovery of processes for process mining. The business process log file is taken from the ProM 6 tutorial [41] with further information about the log file and how to access it in Section B.1.3. The concrete steps to perform process mining are also mostly overlapping with the steps taken by the tutorial. The tutorial applies process mining using the same major ProM version that is used by the prototype so it can be evaluated in which ways the prototype with its sonification capabilities can extend the process mining process using ProM without sonification capabilities.

Log Description

A detailed description of the underlying process is available in [41, p.5f]. In sum, the process describes the repairing of telephones with customer registration, sending in the phone, categorizing the defect, a limited number of repair attempts resulting in either, sending the repaired phone or a new phone back to the customer. For process mining a complete process model is usually not available until the discovery step of the process mining has been finished so it is provided here only for convenience reasons.

Log Preparation

Before a first process model can be mined the quality of the log has to be checked, and it has to be determined which parts of the log are used for the model extraction.

Insufficient quality of the log can prevent the process mining. There may be various reasons why a event log has insufficient quality. If it is not be a well-formed XML file (Section B.1.1) ProM already fails to load the log and therefore no plugin can be applied to the log. The prototype plugin has even stricter requirements for the quality of the log. For example the prototype plugin requires that a time stamp as defined by the XES *time extension* for each event (Section B.1.2) of the log is available. If for any reasons the log does not fulfill the requirements it can not be used with the prototype.

The repair example log has sufficient quality to be loaded by ProM and can also used by the prototype. But not all parts of the log are relevant for the process. The concrete steps required to remove the unwanted parts of the log are described in [41, p.10ff] and have the purpose to filter out ongoing cases, where the final task sending a new or repaired phone to the customer, has not been reached yet. This requirement filters out 104 out of 1104 traces. The resulting log is described in Section B.1.4 and the unfiltered log in Section B.1.3.

Log Inspection

[41, p.8f] asks for basic information about the log file and how this information can be accessed with the log summary available in ProM and shown in Figure 32 for the filtered repair example. It shows absolute and relative occurrence numbers and rates for various event properties and gives information about special properties for the first and last event in a trace.

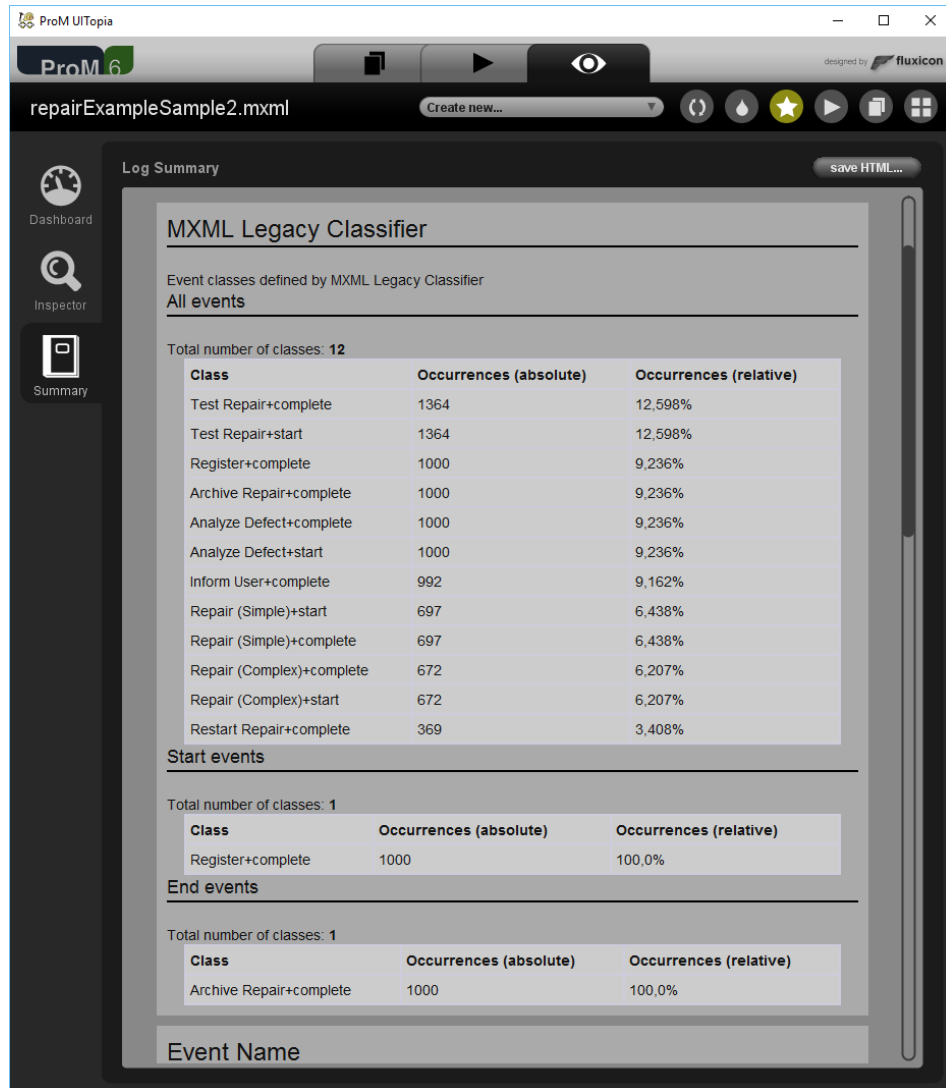


Fig. 32: Log summary

The prototype plugin allows to view similar information about the log but here the information is more focused on the supported sonification mapping capabilities of the tool. It shows for all occurring and supported basic extensions like *concept:name* the sorted values and the values sorted by number of occurrences.

Listing 13: Use case process mining: log summary

Traces: 1000

Events: 10827

time:timestamp

Usage 10827/10827

Values: 01.01.1970 06:36:00.000, 01.01.1970 06:44:00.000, 01.01.1970 06:56:00.000,
01.01.1970 07:03:00.000, 01.01.1970 07:13:00.000, 01.01.1970 07:18:00.000, 01.01.1970
07:24:00.000, 01.01.1970 07:35:00.000, 01.01.1970 07:41:00.000, 01.01.1970
07:51:00.000, 01.01.1970 10:16:00.000, 01.01.1970 10:22:00.000, 01.01.1970
10:29:00.000, 01.01.1970 10:43:00.000, 01.01.1970 11:09:00.000, 01.01.1970
11:15:00.000, 01.01.1970

...

22.01.1970 19:18:00.000, 22.01.1970 19:22:00.000, 22.01.1970 19:28:00.000, 22.01.1970
19:38:00.000, 22.01.1970 19:46:00.000, 22.01.1970 19:48:00.000, 22.01.1970
19:53:00.000, 22.01.1970 19:55:00.000, 22.01.1970 19:56:00.000, 22.01.1970
19:59:00.000, 22.01.1970 20:00:00.000, 22.01.1970 20:03:00.000, 22.01.1970
20:12:00.000, 22.01.1970 20:13:00.000, 22.01.1970 20:18:00.000, 22.01.1970
20:22:00.000, 22.01.1970 20:23:00.000, 22.01.1970 20:30:00.000, 22.01.1970
20:37:00.000, 22.01.1970 20:41:00.000, 22.01.1970 20:46:00.000, 22.01.1970
20:51:00.000, 22.01.1970 20:57:00.000, 22.01.1970 21:04:00.000, 22.01.1970
21:13:00.000

Frequency: 20.01.1970 22:21:00.000 (7), 20.01.1970 02:51:00.000 (7), 12.01.1970
03:32:00.000 (6), 09.01.1970 16:12:00.000 (6), 04.01.1970 12:13:00.000 (6), 19.01.1970
04:33:00.000 (5), 18.01.1970 04:08:00.000 (5), 17.01.1970 17:08:00.000 (5), 16.01.1970
06:39:00.000 (5), 13.01.1970 11:13:00.000 (5), 13.01.1970 09:50:00.000 (5), 13.01.1970
10:20:00.000 (5), 12.01.1970 07:13:00.000 (5), 12.01.1970 04:02:00.000 (5), 12.01.1970
03:17:00.000 (5), 11.01.1970 09:28:00.000 (5), 10.01.1970 16:39:00.000 (5), 09.01.1970
10:24:00.000 (5), 08.01.1970 13:09:00.000 (5), 02.01.1970 12:00:00.000 (5), 07.01.1970
18:17:00.000 (5), 07.01.1970 21:46:00.000 (5), 02.01.1970 22:22:00.000 (5), 06.01.1970
23:09:00.000 (5), 05.01.1970

...

11:15:00.000 (1), 02.01.1970 13:10:00.000 (1), 02.01.1970 12:31:00.000 (1), 02.01.1970
12:30:00.000 (1)

concept:name

Usage 10827/10827

Values: Analyze Defect, Archive Repair, Inform User, Register, Repair (Complex),
Repair (Simple), Restart Repair, Test Repair

Frequency: Test Repair (2728), Analyze Defect (2000), Repair (Simple) (1394), Repair (Complex) (1344), Archive Repair (1000), Register (1000), Inform User (992),
Restart Repair (369)

lifecycle:transition

Usage 10827/10827

Values: complete, start

Frequency: complete (7094), start (3733)

org:resource

Usage 10827/10827

Values: SolverC1, SolverC2, SolverC3, SolverS1, SolverS2, SolverS3, System, Tester1, Tester2, Tester3, Tester4, Tester5, Tester6
Frequency: System (3361), Tester2 (812), Tester1 (810), Tester3 (806), Tester6 (788), Tester5 (778), Tester4 (734), SolverS1 (534), SolverC1 (498), SolverC2 (480), SolverS3 (430), SolverS2 (430), SolverC3 (366)

Parallelism 01.01.1970 06:36:00.000 → 1, 01.01.1970 11:09:00.000 → 2, 01.01.1970 12:25:00.000 → 1, 01.01.1970 20:55:00.000 → 2, 01.01.1970 20:58:00.000 → 3, 01.01.1970 21:52:00.000 → 2, 01.01.1970 21:57:00.000 → 1, 02.01.1970 00:46:00.000 → 2, 02.01.1970 00:55:00.000 → 3, 02.01.1970 01:05:00.000 → 4, 02.01.1970 01:26:00.000 → 3, 02.01.1970 01:54:00.000 → 2, 02.01.1970 02:08:00.000 → 3, 02.01.1970 02:19:00.000 → 2, 02.01.1970 02:35:00.000 → 1, 02.01.1970 02:41:00.000 → 2, 02.01.1970 03:30:00.000 → 1, 02.01.1970 03:33:00.000 → 2, 02.01.1970 03:39:00.000 → 1, 02.01.1970 03:46:00.000 → 2, 02.01.1970 04:52:00.000 → 1, 02.01.1970 08:02:00.000 → 2 ... 22.01.1970 06:53:00.000 → 3, 22.01.1970 07:14:00.000 → 2, 22.01.1970 07:42:00.000 → 1, 22.01.1970 10:40:00.000 → 2, 22.01.1970 11:26:00.000 → 3, 22.01.1970 11:39:00.000 → 2, 22.01.1970 11:50:00.000 → 3, 22.01.1970 12:27:00.000 → 2, 22.01.1970 14:41:00.000 → 1, 22.01.1970 15:02:00.000 → 2, 22.01.1970 15:35:00.000 → 3, 22.01.1970 15:46:00.000 → 2, 22.01.1970 15:55:00.000 → 1, 22.01.1970 19:10:00.000 → 2, 22.01.1970 19:53:00.000 → 3, 22.01.1970 20:03:00.000 → 2, 22.01.1970 20:23:00.000 → 1

Process Discovery

[41, p.13ff] analyzes various process mining perspectives of the filtered repair example log file. A short listing of this perspective, and a summary which plugins helped to get the result for the perspective, give an overview about concrete process mining tasks. Afterwards it can be discussed in detail which role visualization and sonification can play.

- The control-flow perspective examines the order in which tasks are executed and considers aspects like parallel tasks and loops where a sequence of tasks is executed repeatedly. It can be mined with plugins relying on simple algorithms like the α -algorithm or more complex plugins like the Genetic Miner.
- Case-related information like the most frequent path or loops can be determined with Pattern Abstraction visualization.
- Organizational-related information like the number of involved people can be determined with the Social Network plugin.
- The (non-)compliance to rules and regulations can be evaluated with the LTL checker plugin.

ProM Visualization-centric Plugins

Before comparing a multimodal approach to pure visualization two visualization plugins are presented.

The dotted chart visualization can be configured with the two main configuration options being the *Component type* and the *Time option*. The component defines the grouping criteria for the x-axis and is set to trace id (trace *concept:name*)

by default. Other options for the default extensions available in the repair example are to group by *Originator* (*org:resource*), *Task ID* (*event concept:name*) or *Event* (*lifecycle:transition*). The time defines the position on the y-axis and can be set to reflect the actual or multiple variants of the relative position. Figure 33 shows an example for an alternative configuration using the *Task ID* and the *Relative(Time)* options. This allows to gain insight when different event types occur over all traces by setting the same start time for the first event of each trace. For example it can be seen that all traces start with the *Register* event.



Fig. 33: Dotted Chart for filtered repair example - alternative settings

Another visualization-centric plugin is the Basic Performance Analysis⁴⁴ plugin. Unfortunately the author was not able use this plugin with a current version of ProM and so only a short summary of the information available on the plugins homepage is given. The plugin allows to calculate different performance measures like execution time and waiting time. The values can be shown either as plain text in a table or with eight different diagram types ranging from simple one like a bar chart to more specific diagrams like a Gantt chart. Additional options allow to filter outliers and special treatment of a Monday to Friday workweek is possible.

⁴⁴ <http://www.processmining.org/online/basicperformanceanalysis>

Sonification Control-Flow Perspective

Defining the sonification requires a mapping for some or all of the data available in the default extensions. For the repair example log this means that *concept:name*, *lifecycle:transition* and *organizational:resource* could be mapped to two available sonification options either *Melody* and *Instrument* or *Drum* and *Drum Instrument*. *Volume* and *Panning* are not available for this log as no *cost:total* values are defined. To avoid making a wrongheaded mapping decision a closer look at the output of the Log tab and the therein contained Listing 13 is recommended.

- All three default extensions are available for all events.
- For both *concept:name* and *org:resource* various values are defined while *lifecycle:transition* is only defined with start and complete.
- *time:timestamp* is defined for all events but there are many events starting at the same time.

To get a first overview about the control flow of the process the *concept:name* and the *lifecycle:transition* values are more helpful than the *org:resource* values as these two extensions contain a describing name and the status of each event. After identifying which data to use for the sonification the next step is to define which extension should be mapped to which sonification option. As it is easier to distinguish between different melodies than between different instruments *concept:name* having more concrete values gets mapped to *Melody* and *lifecycle:transition* gets mapped to *Instrument*. Alternatively a mapping to *Drum* and *Drum Instrument* would be possible but is not evaluated for this case. The actual mapping is done on two separate tabs that are only shown when the respective extension is mapped. For a start the default melodies are applied. Taking a closer look at the *concept:name*, values four are related to *repair* so these four values are mapped to related sonifications, here by putting them all in their own octave range from c6 to f6 with all other values in a lower octave.

- Analyze Defect c5maj
- Archive Repair g6maj
- Inform User e5min
- Register f5maj
- Repair (Complex) c6maj
- Repair (Simple) d6min
- Restart Repair e6dim
- Test Repair f6maj

The instrument mapping for the lifecycle transition is easier done as only two values have to be mapped. The default instruments are sufficient for this purpose.

The current settings result in an audio available in Section B.1.5. But listening to the whole audio in this way is not useful as the audio would play for more than 21 hours and with long pauses between event sonification as there is an event, in average, only every 3 to 4 minutes. So the next step when dealing

with a low frequency logs is to speed up the the sonification so that the whole log file or at least a single trace can be listened to in a suitable time frame. The tool supports the speed up with different approaches. The first approach is to just increase the playback speed. Here, assuming the log duration is one hour, a speed up of factor 10 would decrease the time required to listen to the log from one hour to 6 minutes. As it may be difficult to determine the speed up factor for a given log duration another way to speed up the playback is to set the the total playback duration.

Regardless of whether the playback speed or the playback duration is set to speed up the sonification it can be difficult to recognize instrument and melody of single event if it is played at a much higher speed. The tool allows to work around this limitation by setting the playback speed of an event sonification independently from the playback speed of the whole log. For example when listening to a single trace with one event every 60 seconds and a total duration of 1 hour setting the total playback duration to 1 minute results in a speed up factor 60. To set this playback speed it can either be set as event playback speed $1x$ or by setting the event speed to *Normal*.

For this concrete example with 10827 events in 31117 minutes a speed up factor 100 results in total playback duration of ~18670 (~5 hours 11 minutes) seconds or on average one event every two seconds. The resulting MIDI file is available in Section B.1.5. But the resulting sonification is not easily understood.

- Many events start at the same time.
- There are many different traces playing at the same time makes it difficult to know to which trace an event sonification belongs.

The problem of too many parallel traces can be fixed by limiting the number of traces, by mapping the trace index to a sonification property or by serializing the playback of traces. The serializing variant is used for the case-related information use case. An example MIDI file and the XES file with only the first trace selected is available in Section B.1.5. By looking at the visualization in the prototype or at the XES log, the reason for overlapping events can be identified as the *complete* lifecycle event of one event overlapping with *start* lifecycle event of the following event. A workaround can be applied by filtering out all events in lifecycle state *start*. This can be done by deselecting the checkbox in the *start* row, the resulting sonification is available in Section B.1.5. Now the listener can identify the task execution order of this single trace. This step could in principle be repeated for many traces but is not efficient as it takes too much time and the listener has to manually somehow keep track of all the findings for each trace. The sonification part of the multimodal approach provides here also no benefits compared to a pure visualization where *concept:name* is mapped to a visualization property.

Sonification Case-Related Information

The previous sonification shows the prototype to determine the sonification mapping for the control-flow perspective. For mining case-related information

Verbeek [41, p.16] asks five questions, from which the question asking for the existence of loops in the log is evaluated here using the multimodal tool. The problem here is to identify which events in a trace get repeated as listening to all traces at once, even at a slow speed, the trace belonging of one event is not deducible solely from the sonification. The first improvement could be an option to listen to all traces one after another with a short signal indicating the jump from one trace to the next. Another help to identify the order of events solely by sonification could be to play one event after the other with a small fixed size gap between each event to make the most of the working memory capabilities of the listener. The option to serialize the traces is available in the standalone variant of the prototype. For the sonification a similar approach is chosen as for the control-flow perspective, the *concept:name* as tasked name is the only required information and only the events in their *complete* lifecycle state are considered. The example in Section B.1.6 shows three selected traces where only the middle one shows a loop where three repair attempts are required before the phone is fixed. The problem with discovering these loops, using the multimodal approach, is that the loops are nested within many traces that don't have any loop. And for the control-flow perspective identifying these traces with a multimodal approach is ineffective compared to algorithms optimized for this purpose. Similar findings can be made when checking the other four questions. Path frequencies and case distributions over the whole log require extensive counting and calculation that aren't suited for visualization or sonification. The remaining two questions tackle operations where a more complex tool than the current prototype is required.

Sonification Organizational-Related Information

For organizational aspects Verbeek [40, p.19] asks six questions from which two of them will be evaluated here with the help of sonification.

- The number of people involved in a specific case.
- Transfers from one role to the next.

The suitability of sonification to identify the number of people depends on the complexity of the case under review. A sonification approach could be to use melodies for the different persons involved in the log. This might be difficult if too many persons are used in the log as it is hard to remember and identify so many melodies. This is aggravated by the fact there is no clear understanding of the underlying process in the process mining phase so it's hard to make sophisticated assumptions which people might or might not be used in the same trace. The sonification can help here with mapping the trace information to an instrument so that the listener can listen to more than one trace at the same time. To get the current person *org:resources* like *SolverC1*, *SolverC2* and *SolverC3* are mapped to the melody property. To have a chance to distinguish overlapping traces the trace name is mapped to instrument.

Finding a sonification to identify the role transfer is simpler compared to the identification of the number of involved people as there are fewer roles than

people in the whole log. For example there are traces where multiple testers are involved that can all be mapped to a single melody as it not necessary to distinguish between the different persons with the role tester in the sonification.

A shortcoming of the prototype prohibits answering both questions at once. If *org:resource* could be mapped to instrument and melody at the same time the melody could be used to identify people and the instrument could be used to identify roles without needing to listen to the sonification twice.

The sonifications for the people and role transfer identification and their descriptions are available in Section B.1.6. In the sonifications only the first ten traces are selected as this number of traces is sufficient to show how the sonification mapping works for identifying the number of people and the role transfer. To simplify the selection of overlapping traces a version of the log is used where the traces are sorted according to their start time. Using the first ten traces still results in no overlapping traces most of the time and periods with two and three parallel traces.

Using sonification for the two questions mentioned above works for bigger logs only for a sample consisting of few traces and the same results can be achieved with visualization alone, probably with less effort. Answering any of these six questions for the whole log is better done with algorithms.

5.2 Process Analysis

Characteristic for the process analysis is that the as-is processes are known after the process identification and process discovery phase have passed.

5.2.1 Repair example

After using the prototype tool for process mining in the previous chapter the same log is used for the first process analysis use case. As the log has no pronounced anomalies two fictive anomalies are introduced. In both cases the human resources department can gain insights by using the sonification for a process analysis.

- The number of cases is increased during a specified time frame. The HR department needs to react to these changes as employees may get overworked which may lead to a quality decrease how the employees can handle their tasks. In this case the increase lasts only for a short period but a similar sonification approach could be used for detecting longer increases that may require a resource replanning.
- The *org:resource* describing the person who is executing a specific activity changes for a specific task during a specified time frame. Noticing such changes may be of interest to the HR department as it allows them investigate the reasons why the workload changed among employees fulfilling the same role.

As noted before the original repair example log has 1000 completed traces. To make the visualization screenshots more overseeable and allow to listen to all

sonified events the traces are sorted according to the start time of first the event and only the first 100 traces are used to introduce the anomalies.

Visualization

For the uninterrupted state an overview is available in Figure 34 and the log and its two variations required for this use-case are available in Section B.2.1.



Fig. 34: Uninterrupted state for first 100 traces

Case 1 - Increased throughput

The task for the anomaly in this case is to identify the increased throughput and the period when it happens. This increase is reflected in the log by reducing the duration between the start of two subsequent traces to one fifth of its original value for the traces with the index 41 to 60. The traces with the index 61 to 100 have an unchanged relative delay compared to their respective predecessors. This results in an increase in parallelism defined as the number of active traces at a given point in time. A trace is active from the time stamp of the first event to the time stamp of the last event. Figure 35 shows the parallelism for the uninterrupted state and the increased throughput case. The maximum parallelism increases from six to 14. Note that the effective parallelism for the sonification may be higher as the sonification of the last event of a trace needs a specific amount of time starting with the time stamp of the last event.

Recognizing the increase may be of interest for the Human Resources department as it performs a capacity planning based on results from the past.

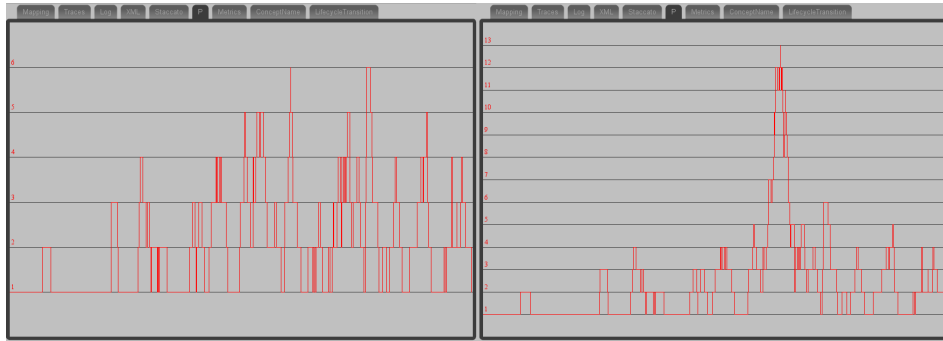


Fig. 35: Parallelism increase for increased throughput on the right side compared to uninterrupted state on the left side

Visualization

The increased throughput case is visualized in Figure 36. The modified traces with the indexes 40 to 59 are located in the vertical middle of the visualization. While the traces before and after these traces are starting one after each other with a distinct gap between two subsequent traces, the modified traces are happening with short gaps between each event. The visualization allows also to look for changes by simply comparing the visualization of the increased throughput state with the visualization of the uninterrupted state. But of course this parallel world comparison works only for very regular event sources where a certain sequence of events occurs periodically.



Fig. 36: Low frequency case 1 dotted chart

Sonification

Listening to the sonification with the default settings requires too much time, as even after increasing the throughput, the log still lasts for almost two days and nine hours. When increasing the playback speed by setting the playback duration to one minute while keeping the event speed normal the increase can be heard but it's not clear whether this increase has its cause in more events per trace or in a reduced gap between subsequent traces. Filter all but the first event type (*concept:name Register*) allows to identify the time when the increase happens more precisely. This first process analysis use case shows that sonification is a good fit when deviations from a normal state should be recognized. In this case a change in the temporal density of the traces is easily identifiable by the listener. And as opposed to many of the process mining use cases there is no need keep a sonification of the whole log in memory. It is sufficient to have a sliding window like perception where the listener can focus only on this current sonification window. Together with the automatically generated summary of the sonification a deviation can be recognized. All sonifications with the detailed applied settings to generate the sonifications for the longer execution time case of this use case are available in Section B.2.2.

Case 2 - Changed task executor

The task for the anomaly in this case is to identify the period when the task executors change and which persons are affected by this change. For this purpose the traces with the index 41 to 60 the *org:resource* are set from *SolverC2* and

SolverC3 to *SolverC1* wherever necessary. This anomaly may be of interest to a department manager in charge of solving activities in the company.

Visualization

The changed task executor is visible in Figure 37. The anomaly is visible as a gap for *SolverC2* and *SolverC3* around the white vertical line right next to the middle. This gap indicates that consequent tasks are solved by only one solver. For this case it is useful to disable the rendering option that the dot size should increase with the number of event as one dot per event allows a better estimation for the number of events happening.



Fig. 37: Low frequency case 2 dotted chart

Sonification

As for the increased throughput case the sonification is speed up to a play-back duration of one minute. As the focus is on analyzing the complex solvers all other tasks get filtered. The listener needs to distinguish between the three solvers. This could be done by either using different melodies or instruments. Using easily distinguishable instruments for the different solvers is best combined with a short melody or tone. Here it is important not to use too short notes as it is difficult to identify an instrument when listening to only a short note. For both visualization and sonification identifying the deviation is easy but requires an understanding what to look for. In this use case when the department manager is suspicious that role specific work is not done by all responsible persons in equal shares a concentration of work can only occur within the groups simple solvers, complex solvers and testers, each having between three and six members. This was arguably the simplest case, to recognize where the work is done roughly done in equal shares by all three persons in the uninterrupted state, and only by one person in the interrupted state. But even more complex cases like a period where all the work is done by only half of the testers team should be recognizable. As for the earlier use cases identifying a 10% under-performance over the whole log by one person is more suited to an algorithmic analysis. For the simple case above comparing the multimodal approach fo-

cusing on the sonification with the pure visualization shows a similar required effort to configure the data display. It still might be quicker to analyze in the pure visualization as a possible work concentration for multiple teams can be detected using the same visualization.

The sonification with the detailed settings for the changed task executor case of this use case is available in Section B.2.3

5.2.2 High Frequency Logs

Process Description

A high frequency log is characterized by many events happening in a short time. As a concrete example, a fictive order process in a web shop is analyzed. The process is shown in Figure 38 ([20]) and shows the steps of a customer, from surfing the web shop homepage until an order is placed or the order process is canceled. Each activity belongs either to the *customer* or the *system*, with only the *Process Availability* activity being assigned to the *system*. The numbers on the exclusive gateways describe the probabilities of the different possible paths. The execution of each activity varies within a given range that is shown together with other information in Table 10. With this information the process is simulated for twelve hours where the delays between the first activities (*Start Page*) of subsequent traces varies between 1 and 5 minutes. The average delay between two subsequent traces is increased over the 12 hour time frame to simulate the slowdown of demand over the night. At beginning of the evening the interval between two traces are very close to one second while at the end of the log the interval can vary between 1 second and 5 minutes.

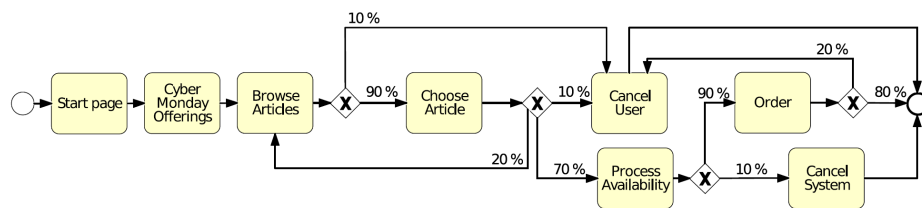


Fig. 38: BPMN Diagram High Frequency Log - Web shop ([20])

Name	org:resource	Min	Max	Follower 1	Follower 2	Follower 3
Start Page	customer	500ms	5s	Cyber Monday Offerings	-	-
Cyber Monday Offerings	customer	100ms	5s	Browse Articles	-	-
Browse Articles	customer	100ms	10s	90% Choose Article	10% Cancel User	-
Choose Article	customer	1s	10s	70% Process Availability	20% Browse Articles	10% Cancel User
Cancel User	customer	1s	1s	End	-	-
Process Availability	system	100ms	2s	90% Order	10% Cancel System	-
Order	customer	100ms	10s	80% End	20% Cancel User	-
Cancel System	system	1s	1s	End	-	-

Table 10: Execution times high frequency log

The log contains three anomalies for which the web shop owner uses sonification to identify the deviations.

1. Two hours after the log starts the activity *Process Availability* takes ten times longer as usual, resulting in execution times between one second and 20 seconds. This should simulate the slowing down of the software component responsible for the activity. The anomaly continues for two hours. The web shop owner is interested in a constant service and therefore wants to notice the deviation and estimate how long it takes.
2. Six hours after the log start a system malfunction is simulated by changing the probabilities of the XOR decision following the *Process Availability* activity. All traces continue with the *Cancel System* task for the two hour duration of this anomaly. Again the web shop owner is interested in a constant service quality that may be disturbed if customers can't proceed past the *Process Availability* activity.
3. Nine hours after the log start a burst of user demand is taking place for 5 minutes. During this period the interval between two traces is shortened to a random value between 200 ms and five seconds. Here the web shop owner is interested in identifying the time, duration and strength of the change in demand so that she can think about possible causes for the rise.

Sonification

As shown in Section 5.2.1 playing back the events at the same speed as they have been recorded is not always a viable option. When dealing with high frequency logs the problem is similar as the actual event occurrence rate doesn't fit to the desired playback frequency. The playback has to be slowed down to allow the listener to distinguish the different events. This can be done by using

the same steps as in Section 5.1 but this time the playback speed of the event sonifications may have to be decreased compared to the total playback speed.

Another difficulty lies in handling the huge amount of events available in a high frequency log file. A process manager may only be interested in a few seconds of the log where all required information is available. One way to find the interesting part is to use the zoom functionality of the visualization component as shown in Figure 39. This allows to distinguish the individual events that are indistinguishable when the whole log is shown on one screen. If there are too many parallel traces limiting the number of active traces might be helpful. For this concrete log, both zooming in and limiting the number of traces has been applied to get a manageable amount of events.

Figure 39 shows also a black bar in the zoomed in part of the visualization. This bar indicates the current position of the sonification and can be set to any position so that the sonification does not need to start with the first event of the currently selected traces. When the interesting part of the log is identified other ProM plugins can be used to trim the log to contain only this part.

For the concrete sonification in Section B.2.5 the first five traces have been selected, the playback speed slowed down. The activity type is mapped via *concept:name* to melody and trace id is mapped to instrument. Even with this limited number of traces it requires some traces to distinguish between the different traces in the sonification and concentrate on the activities in one or two traces.

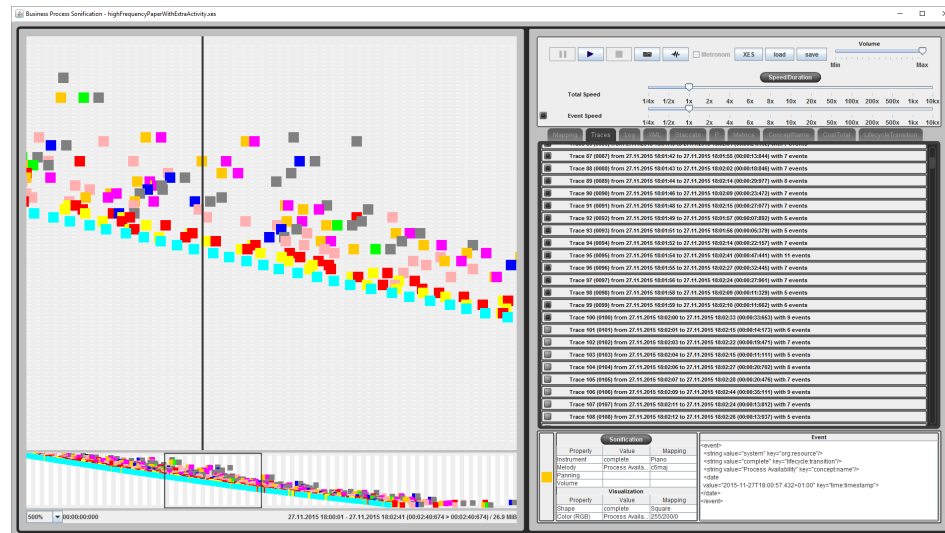


Fig. 39: Zoom into a high frequency log file

Anomaly 1 - Longer execution time

Visualization

The longer execution times are easy to notice in the visualization part of the prototype shown in Figure 40. First this requires to identify the trace which starts the increasing execution time of the *Process Availability* activity. The visualization options of the prototype are too limited for this purpose so that the screenshot shows only 20 traces, 10 before and 10 after the change where the increased distance between the *Process Availability* activity in orange and the follow up activities *Cancel System* (green) and *Order* (pink) shows the longer execution time.

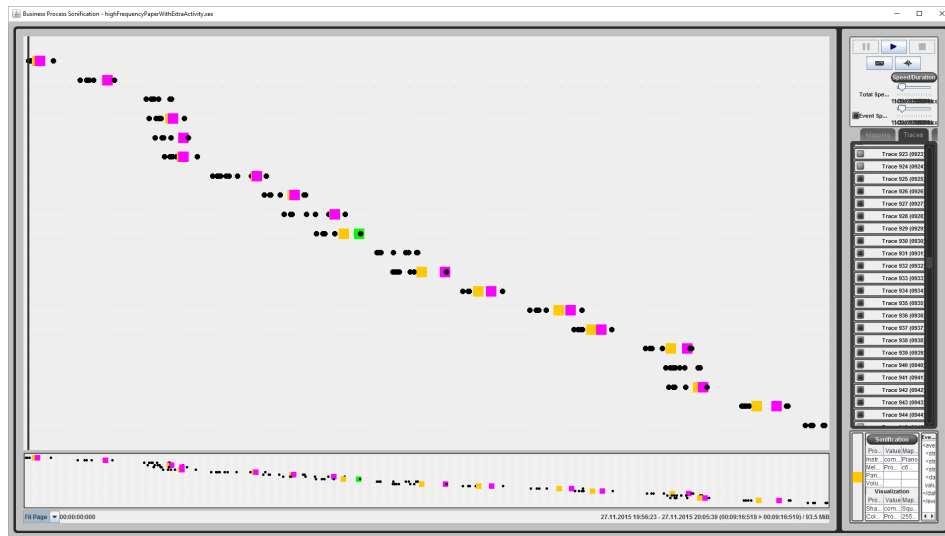


Fig. 40: Anomaly 1 prototype visualization

Sonification

To identify the longer processing time the *concept:name* values are filtered to *Process Availability* and its two successors *Order* and *Cancel System*. A first idea for the sonification could be to map the activity names to melody and the executor of the activity (*org:resource*) to instrument. But this would make it impossible to make use of the activity types for overlapping traces. The listener could not identify which *Order/Cancel System* activity belongs to which *Process Availability* activity. To allow this identification the trace index is mapped to the instrument. For the melody three short single notes are used with the *Process Availability* having a medium pitch, the *Order* a higher pitch and the *Cancel System* a lower pitch. The main difficulty is now to choose an appropriate sonification speedup. Many overlapping traces at the beginning causes also overlaps for the limited activity selection chosen for this case. So if the sonifica-

tion speedup is too high, anomalies at the beginning of the log would remain unnoticed by the listener. A slower sonification speed would allow to notice anomalies at the beginning of the log but listening to the whole sonification would take too much time and the sonification would be mostly empty at the end. Using the untangle functionality of the tool fits also not for this case as the breaks before and after the last selected activity of a trace until the beginning or end of the trace still remain part of the total sonification. This could be improved by letting the untangle feature consider only the actively sonified events but this is currently not support. The chosen sonification speed of 20 is chosen to allow identifying the anomaly after two hours but subtle anomalies at the beginning would remain unnoticed at this speedup. Also the total duration takes around 35 minutes which is too much time to identify this anomaly compared to using advanced visualization tools.

All sonifications with the detailed settings applied settings to generate the sonifications for the longer execution time case of this use case are available in Section B.2.6.

Anomaly 2 - Changed probabilities

Visualization

Figure 41 shows a new perspective of the dotted chart visualization with each activity type in a separate lane, the time on the y axis with the beginning of the log on the top. Of special interest are here the *Order* and the *Cancel System* lanes, the two possible outcomes of the *Process Availability* activity, which is shown as a constant line, meaning no longer interrupts on its own lane. For the *Order* activity the gap beginning in the second half of its lane is easily noticeable. Knowledge about the the process model and the the uninterrupted *Process Availability* activities leads the interpreter of the visualization to check for the only other possible outcome the *Cancel Activity* activities. During the gap in one lane a increased level in system cancellations allows to conclude that the process executions behave in an atypical way during this period.

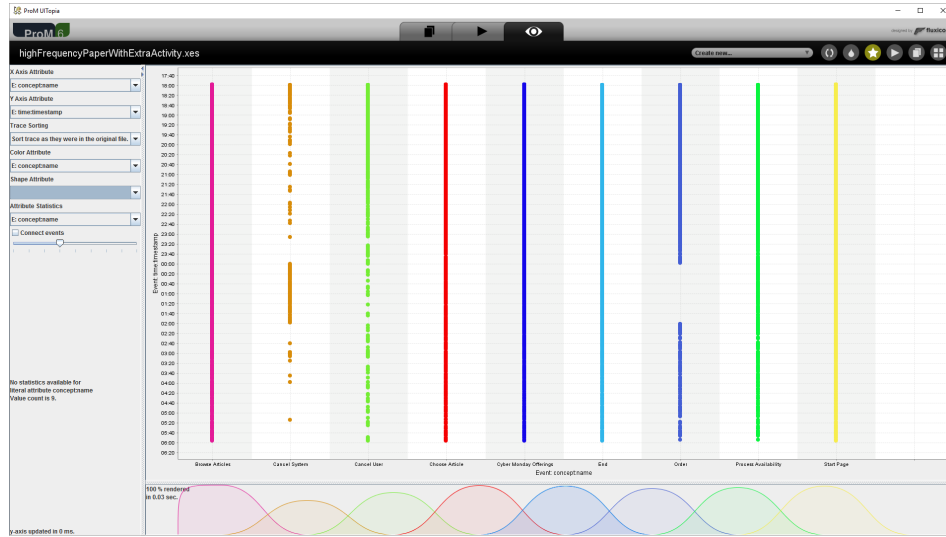


Fig. 41: Anomaly 2 dotted chart visualization

Sonification

For the sonification we assume that the webshop owner is interested if the system is reliable for customers that get to the *Process Availability* activity. Here the *Cancel System* activity represents errors that should be clearly fewer in number than successful *Order* activity sonifications. Another aspect of interest could be the price that comes along with each order. So the mapping has to map at least the three activity types and the *cost:value* of the *Order* activities. The current linear volume mapping allows to identify positive outliers as they would be much louder than the remaining activities. A problem of the currently chosen volume mapping approach is that only activities where the volume mapping data is available get assigned a volume from almost inaudible to full volume while all other activities are mapped at full volume. This applies to this case's sonification where only the *Order* activities have the *cost:total* value. For the activity types three single notes are chosen with *Process Availability* as base and from there going a third down to indicate a *Cancel System* and a third up to indicate a *Order*. The remaining sonification option is used to map trace names to instruments to help the listener to easier keep track which events originate from the same trace.

All sonifications with the detailed applied settings to generate the sonifications for the changed probabilities case of this use case are available in Section B.2.7.

Anomaly 3 - Activity Burst

Visualization

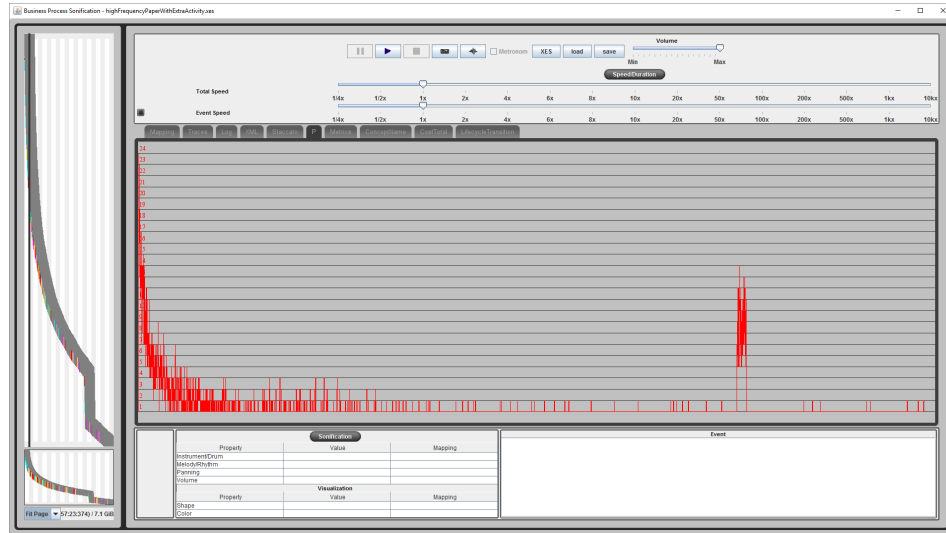


Fig. 43: Anomaly 3 prototype visualization

Sonification

For the sonification we assume a typical check of last nights activities performed by the webshop owner on the following morning. She may be especially interested in the users arriving at the shop and the users performing a buying order. These interests map to the two activities *Start Page* and *Order*. This time the drum instrument and rhythm sonification option is used, mapping the different activities to two easily distinguishable instruments, the bass drum and the wood block. The *org:resource* is mapped to the rhythm but since both *Start Page* and *Order* belong to the same *org:resource customer* this sonification property is not used for any differentiation. For the rhythm just a very short single note is chosen that needs just to be long enough to distinguish the different instruments. To fulfill the purpose of the sonification, as summary of the preceding night, a playback duration of two minutes for the whole duration is chosen. This settings result in a sonification that resembles a Geiger counter in the beginning when the interval between subsequent traces is very short. With the decrease in trace starts the listener is more and more able to distinguish between *Start Page* and *Order* activities. If the listener is interested in a more exact differentiation between between *Start Page* and *Order* activities the speedup factor needs to be reduced. Arriving at the activity the change in the sonification is easily recognizable.

All sonifications with the detailed applied settings to generate the sonifications for the changed probabilities variant of this use case are available in Section B.2.8.

6 Conclusion

Section 1 laid out the motivation and defined the research question. Section 2 covered the topics required to tackle these questions and identified the lack of research and tool support a multimodal sonification. The design options and their pros and cons have been shown in Sections 3 while Section 4 arguments why the earcon based sonification has been chosen and how it works in detail. The suitability of this tool was shown in Section 5 for process mining and process analysis use cases. With all these result the research question can now be answered.

1) Are sonification techniques and concrete mappings useful to perform typical process mining and process analysis tasks? If so which insights can be gained by these techniques and mappings? Which properties of the process data are beneficial for sonification?

Musical earcons, with the ability to parameterize them, prove useful to sonify individual events. Melodies together with instruments are easier to distinguish than rhythms applied to drum instruments. In the use-cases only the volume mapping is used but it can be assumed, that also for panning, smaller changes in the mapped values are difficult to identify. So in its current state the sonification tool is better suited to work with logs making use of nominal data than logs relying heavily on continuous data. The input data needs to contain time stamps, otherwise a sonification is not possible. As a custom sonification with earcons is not intuitive and requires the user to perform some training before results can be achieved the log should contain at least hundreds of events otherwise the reduced training effort of a pure visualization is to be preferred. The use cases also show that specific preprocessing of the input data, e.g. in the form of untangling the traces, can complement mapping concepts. Comparing the suitability of sonification for process mining and process analysis shows that identifying deviations for a process analysis is more promising than performing log exploration for process mining. Taking a closer look at process analysis, a qualitative process analysis e.g. in the form of a root-cause analysis is feasible with a multimodal tool while a quantitative process analysis is better done applying algorithms to the log or relying on more sophisticated visualization tools. When listening to concrete sonifications the identifiable deviations may be caused by one of the following changes to one or more input data dimensions: increase or decrease in its frequency, change of order or its complete omission.

2) Which kind of visualization fits to a multimodal approach where the focus is on sonification?

The dotted chart like visualization proves adequate for an event-centric sonification as it allows the user to estimate where a mapping is used in the log. The additional aspect of sonification being a temporal medium is reflected by a progress indicator. The supported concrete color and shape mappings help the listener to get a visual preview of some aspects of the sonification while other important aspects like the duration of the sonified event are not visible. The use cases repeatedly show that overlapping traces resulting in overlapping

sonifications can overwhelm the listener. For this purpose the parallelism visualization is helpful to identify this problems upfront before listening to the sonification.

3) Both sonification and visualization have to be configured and applied to concrete logs. Log data may have to be loaded and filtered. The sonification needs to be played and the visualization shown. All these activities require user interactions leading to the question how the tool can best support the user in these interactions.

A trial and error approach is supported by allowing the user to change a sonification mapping without losing more than necessary of the existing mapping. The most frequently used filtering and sorting is best directly supported by the tool so that the user does not need to switch to other packages or tools. As not all possible filtering and sorting algorithms should be reimplemented the integration in an existing plugin ecosystem like with ProM is recommended. This also helps when preceding tasks like process mining are performed without using sonification and sonification is used for the process analysis phase. All visualizations require little to no configuration effort as they serve only to show the changes to the sonification mapping. This design decision is helpful to let the user focus on sonification aspects.

The multi-modal sonification as shown in this paper showed its usefulness to cover some process analysis tasks that could have otherwise been solved by a pure visualization. There were no insights exclusive to the multi-modal approach, and there exists a long list of restrictions as the developed tool is still very limited in supporting different sonification and mapping techniques. Requiring time stamps in the log follows from the motivation to match temporal data with the temporal medium sonification. Other restrictions like to avoid too large log files are caused by the simplistic implementation. The multi-modal sonification can not replace algorithmic process mining and analysis algorithms but this is not to be expected as pure visualizations can not achieve this either.

Before showing some concrete ideas how the current state of the multimodal sonification tool can be improved, a few lessons learned are mentioned.

On a conceptual level there are a few lessons. For the pure sonification it is difficult to find the right information density that is neither overloading the user nor results in a sonification that takes too long to listen to. While a visualization containing too much information may allow the user to recognize the dominating aspects of the visualized data a overloaded sonification may appear to the user as noise without any identifiable or assignable elements. This is reinforced by most users lack of experience with sonification compared to visualization. This leads also to the importance of input data filtering and transformation for the sonification. Unwanted elements in the sonification need to be filtered out to keep the sonification understandable. Acknowledging the importance of data transformation as used by the trace disentanglement feature when loading a process log file was gained by experimenting with the use cases shown above. The first sonification mapping concepts did not contain this data transformation aspect as it is not consistent with the idea of a natural match between sonification as temporal medium and process logs with time stamps

for each event. Regarding the complete sonification, one should not aim for a perfect sonification but apply different sonification approaches to different use-cases or use concrete use-cases as inspirations for new sonification approaches.

Regarding the practical implementation the foremost difficulty is finding a matching sonification technology for the intended sonification techniques. The options are limited by the implementation platform and interoperability between different technologies is difficult to achieve. Even when the technology works in principle, a naive approach to scale it to large sonifications may not be possible. This is shown by the non-linear increase in computation time when using JFugue. When using these sonification technologies there are fewer answered questions in the web compared to visualization technologies that have more users. When experimenting with different sonification approaches a multimodal tool may require changes in many different parts of the application code. This may lead to spending too much time with low-level implementation details especially if the change in the approach is not reasoned out. By listening to two different sonifications for the same log it may be difficult to identify the differences in the mapping. A functionality to export the mapping as text as shown e.g. in Section B.1.5 is helpful for this purpose and allows the user to use her preferred text comparison tool to identify changes if the delta is not obvious.

6.1 Future work

Business process sonification has not been applied practically on concrete use cases so using the prototype to tackle concrete process mining and process analysis questions revealed many shortcomings that would make a further theoretical and practical effort necessary. To have a starting point for future work, these shortcomings and other ideas are collected here.

6.1.1 Sonification

- At the moment only single events can be sonified and the input for the events sonification is solely the event data. Other possible input are imaginable, for example a global status like the parallelism count at a given time stamp.
- Currently only earcons are supported. Other sonification types like auditory icons could allow for richer or more intuitive sonifications. This may require the use of other other sonification techniques beside JFugue.
- The current earcon mapping concept doesn't support compound earcons.
- The mapping allows only to use the concrete event values of the standard extensions. Data saved in custom elements or attributes is not accessible as input for the sonification. A more flexible mapping with XPath expressions as data selector could broaden the data input.
- Traces can be selected and deselected for the sonification but the temporal placement of the single events depends solely on their relative position in the log. Its not possible to make the playback order depending on the trace

an event belongs to. This could be useful when a sonification is intended where traces are sonified sequentially.

- Make the filtering working on standard extension independent from the usage for sonification. At the moment the tool allows only to filter when the standard extension is influencing the sonification.
- The duration of an activity measured as time from lifecycle start to end can not be used as input parameter for the the sonification.
- The same log property can not be used for multiple sonification properties. E.g. *org:resource* could be interpreted as person and as role for the organizational information.
- KPIs (e.g. number of events per period) could be used as input for the sonification.
- The standard lifecycle values could be supported by predefined intuitive sonifications where for example a ascending melody signals the *start* state while a descending melody stands for the *complete* state.
- Adding intelligent playback speed, limiting the playback speed to an acceptable cognitive load depending on the listeners sonification proficiency.
- Support multiple default mappings so that the best fit for the current log and analysis target can be chosen.

6.1.2 Visualization

- Multiple events at same time and trace can not be distinguished. This could be improved by supporting blending of event visualizations.
- A huge number of traces are difficult to distinguish without vertical zooming support.
- The parallelism visualization could be improved by considering the actual event sonification time for the parallelism calculation. Another improvement could be to show the parallelism not only on trace level but also on event level as how many events happen at the same time.
- The visualization for melody, rhythm, instruments and drum instruments doesn't convey any information about the concrete mapping but only which mappings are assigned to an event. The display of other aspects like the duration of the concrete mapping would allow the user to get visual hints about these aspects.
- Volume and panning are not reflected in the visualization and would require to make use of gradient or similar visual concepts.
- A simple vertical line as progress indicator makes it difficult for overlapping traces to identify which event is sonified when a detailed inspection is intended.

6.1.3 Interaction Design

- Using the same sonification for different input data is not supported by the tool and not reflected by default by the visualization.
- Undo/Redo would allow for more robust trial and error to find a suitable sonification.

6.1.4 Software Architecture / ProM Integration

- Make the visualization and sonification output for applying a concrete mapping to a log accessible to other plugins.
- Make the sonification functionality available as a library, so it can be used in other programs e.g. as for a web based solution or when using the command line.
- Allow to sonify bigger logs with more events by splitting the resulting music strings into smaller units that are sonified individually and where the individual sonifications are combined for the playback or export.

References

1. Meera M Blattner, Denise A Sumikawa, and Robert M Greenberg. Earcons and icons: Their structure and common design principles. *Human-Computer Interaction*, 4(1):11–44, 1989.
2. Stephen Anthony Brewster. *Providing a structured method for integrating non-speech audio into human-computer interfaces*. PhD thesis, University of York York, UK, 1994.
3. JCAM Buijs. Mapping data sources to xes in a generic way. *Masters Thesis*, 2010. Available at http://www.processmining.org/_media/xesame/xesma_thesis_final.pdf.
4. William Buxton, William Gaver, and Sara Bly. Auditory interfaces: The use of non-speech audio at the interface. *Unfinished book*, 1994.
5. Robert M Candey, Anton M Schertenleib, and WL Diaz Merced. Xsonify sonification tool for space physics. *Proceedings of the 12th International Conference on Auditory Display*, 2006.
6. H.J. Charwat. *Lexikon der Mensch-Maschine-Kommunikation*. Oldenbourg, 1994.
7. Haase Chet et al. *Filthy Rich Clients: Developing Animated And Graphical Effects For Desktop JavaApplications*. Pearson Education India, 2008.
8. Alan Cooper, Robert Reimann, and Dave Cronin. *About face 3: the essentials of interaction design*. John Wiley & Sons, Inc., 2007.
9. Alberto de Campo. A data sonification design space map. In *Proc. of the 2nd International Workshop on Interactive Sonification*, York, UK, 2007.
10. Wanda L Diaz-Merced, Robert M Candey, Nancy Brickhouse, Matthew Schneps, John C Mannone, Stephen Brewster, and Katrien Kolenberg. Sonification of astronomical data. *Proceedings of the International Astronomical Union*, 7(S285):133–136, 2011.
11. Marlon Dumas, Marcello La Rosa, Jan Mendling, and Hajo A Reijers. *Fundamentals of business process management*. Springer, 2013.
12. ADN Edwards, SA Brewster, and PC Wright. Experimentally derived guidelines for the creation of earcons. *Proceedings of Human Computer Interaction (HCI'95)*, 1995.
13. Stavros Garzonis, Simon Jones, Tim Jay, and Eamonn O'Neill. Auditory icon and earcon mobile service notifications: intuitiveness, learnability, memorability and preference. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1513–1522. ACM, 2009.
14. William W Gaver. The sonicfinder: An interface that uses auditory icons. *Human-Computer Interaction*, 4(1):67–94, 1989.
15. William W Gaver, Randall B Smith, and Tim O'Shea. Effective sounds in complex systems: The arkola simulation. In *Proceedings of the SIGCHI Conference on Human factors in Computing Systems*, pages 85–90. ACM, 1991.
16. Christian W Günther and Wil MP van der Aalst. A generic import framework for process event logs. In *International Conference on Business Process Management*, pages 81–92. Springer, 2006.
17. Christian W Günther and HMW Verbeek. Xes-standard definition. *Fluxicon Process Laboratories*, 13:14, 2014.
18. Thomas Hermann, Andy Hunt, and John G Neuhoff. *The sonification handbook*. Logos Verlag Berlin, 2011.
19. Tobias Hildebrandt. Towards enhancing business process monitoring with sonification. In *Intl. Workshop on Theory and Applications of Process Visualization (TAProViz 2013)*, Business Process Management Workshops, pages 529–536, Berlin, Germany, August 2013. Springer.

20. Tobias Hildebrandt, Felix Amerbauer, and Stefanie Rinderle-Ma. Combining sonification and visualization for the analysis of process execution data. In *18th IEEE Conference on Business Informatics (CBI) 2016*, Business Informatics (CBI), 2016 IEEE 18th Conference on, pages 32–37, December 2016.
21. Tobias Hildebrandt, Simone Kriglstein, and Stefanie Rinderle-Ma. Beyond visualization: On using sonification methods to make business processes more accessible to users. In *18th International Conference on Auditory Display (ICAD 2012)*, pages 248–249, Atlanta, June 2012. Georgia Institute of Technology.
22. Tobias Hildebrandt, Simone Kriglstein, and Stefanie Rinderle-Ma. On applying sonification methods to convey business process data. In *CAiSE 2012 Forum*, pages 74–81. CEUR, 2012.
23. Tobias Hildebrandt and Stefanie Rinderle-Ma. Toward a sonification concept for business process monitoring. In *19th International Conference on Auditory Display (ICAD 2013)*, pages 323–330, Lodz, Poland, July 2013. Lodz University of Technology Press.
24. Anna A Kalenkova, Massimiliano de Leoni, and Wil MP van der Aalst. Discovering, analyzing and enhancing bpmn models using prom. In *Business Process Management-12th International Conference, BPM*, pages 7–11, 2014.
25. David Koelle. The complete guide to jfugue: Programming music in java. 2008. Purchasable at <http://www.jfugue.org/4/book.html>.
26. David Koelle. The complete guide to jfugue: Programming music in java 2nd edition. 2015. Purchasable at <http://www.jfugue.org/guide.html>.
27. Marinos Koutsomichalis. *Mapping and Visualization with SuperCollider*. Packt Publishing Ltd, 2013.
28. Gregory Kramer. An introduction to auditory display. In *Auditory Display: Sonification, Audification and Auditory Interfaces, SFI Studies in the Sciences of Complexity, Proceedings*, pages 1–77. Addison-Wesley, 1994.
29. James Manyika, Michael Chui, Brad Brown, Jacques Bughin, Richard Dobbs, Charles Roxburgh, and Angela H Byers. Big data: The next frontier for innovation, competition, and productivity, may 2011. Available at <http://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/big-data-the-next-frontier-for-innovation>.
30. Elizabeth D Mynatt. Transforming graphical interfaces into auditory interfaces. In *Conference Companion on Human Factors in Computing Systems*, pages 67–68. ACM, 1995.
31. SB Rinderle, Ralph Bobrik, Manfred Reichert, and Thomas Bauer. Business process visualization-use cases, challenges, solutions. *Proceedings of the Eighth International Conference on Enterprise Information Systems (ICEIS'06): Information System Analysis and Specification*, pages 204–211, 2006.
32. Ben Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Visual Languages, 1996. Proceedings., IEEE Symposium on*, pages 336–343. IEEE, 1996.
33. Daniel R Smith and Bruce N Walker. Effects of auditory context cues and training on performance of a point estimation sonification task. *Applied Cognitive Psychology*, 19(8):1065–1087, 2005.
34. Minseok Song and Wil MP van der Aalst. Supporting process mining by showing events at a glance. In *Proceedings of the 17th Annual Workshop on Information Technologies and Systems (WITS)*, pages 139–145, 2007.
35. Dimitros Tzovaras. *Multimodal User Interfaces: From Signals to Interaction*. Springer Science & Business Media, 2008.

36. Wil Van Der Aalst. *Process mining: discovery, conformance and enhancement of business processes*. Springer Science & Business Media, 2011.
37. Wil Van Der Aalst. Process mining: X-ray your business processes. 2011. Available at <http://www.wis.win.tue.nl/~wvdaalst/publications/p678.pdf>.
38. Wil Van Der Aalst, Arya Adriansyah, Ana Karla Alves de Medeiros, Franco Arcieri, Thomas Baier, Tobias Blickle, Jagadeesh Chandra Bose, Peter van den Brand, Ronald Brandtjen, Joos Buijs, et al. Process mining manifesto. In *Business process management workshops*, pages 169–194. Springer, 2012.
39. BF van Dongen and Sh Shabani. Relational xes: data management for process mining. In *CAiSE Forum*, pages 169–176, 2015.
40. HMW Verbeek, JCAM Buijs, BF Van Dongen, and Wil MP van der Aalst. Prom 6: The process mining toolkit. *Proc. of BPM Demonstration Track*, 615:34–39, 2010.
41. HMW Eric Verbeek and RP Jagadeesh Chandra Bose. Prom 6 tutorial. *ProM.(nd). Extraído el*, 27, 2010.
42. Bruce N Walker. Magnitude estimation of conceptual data dimensions for use in sonification. *Journal of Experimental Psychology: Applied*, 8(4):211, 2002.
43. Bruce N Walker and Joshua T Cothran. Sonification sandbox: A graphical toolkit for auditory graphs. *Proceedings of the Ninth International Conference on Auditory Display ICAD2003*, 2003.
44. Ina Wechsung. *An evaluation framework for multimodal interaction*. PhD thesis, Springer, 2014.
45. Mathias Weske. *Business process management: concepts, languages, architectures*. Springer Science & Business Media, 2012.
46. Katja Windt, Michael Iber, and Julian Klein. Grooving factory-bottleneck control in production logistics through auditory display. *Proceedings of the 16th International Conference on Auditory Display (ICAD2010)*, June 2010.

A Resources

A.1 Sonification Sound Effect

Listing 14: Sonification sound effect

```
I[Gunshot]
cw rs ch rs cq rs ci rs cs rs ct rs cx rs co rh
c5 rs c6 rs c7 rs c8 rh
RT120 c e g rh T240 c e g rh T480 c e g
```

sonification_gunshot.flac

A.2 Sonification Default Instruments

Listing 15: Sonification default instruments

```
V0 I[Piano] c d e f g a b
V1 I[Celesta] c d e f g a b
V2 I[Violin] c d e f g a b
V3 I[Trumpet] c d e f g a b
V4 I[Overdriven_Guitar] c d e f g a b
V0 I[Electric_Piano_2] c d e f g a b
V1 I[Glockenspiel] c d e f g a b
V2 I[Tremolo_Strings] c d e f g a b
V3 I[Oboe] c d e f g a b
V4 I[Guitar_Harmonics] c d e f g a b
V0 I[Harpsichord] c d e f g a b
V1 I[Xylophone] c d e f g a b
V2 I[Pizzicato_Strings] c d e f g a b
V3 I[French_Horn] c d e f g a b
V4 I[Sawtooth] c d e f g a b
V0 I[Clavinet] c d e f g a b
V1 I[Tubular_Bells] c d e f g a b
V2 I[Orchestral_Strings] c d e f g a b
V3 I[Tenor_Sax] c d e f g a b
V4 I[Charang] c d e f g a b
```

sonification_default_instruments.flac

A.3 Sonification Default Melodies

Listing 16: Sonification default melodies

```
V0 c5maj
V1 d5min
V2 e5min
V3 f5maj
V4 g5maj
```

V0 a5min
V1 b5dim
V2 c6maj
V3 d6min
V4 e6min
V0 f6maj
V1 g6maj
V2 a6min
V3 b6dim
V4 c7maj
V0 d7min
V1 e7min
V2 f7maj
V3 g7maj
V4 a7min
V0 b7dim

sonification_default_melodies.flac

A.4 Sonification Default Drum Instruments

Listing 17: Sonification default drum instruments

V9 [ACOUSTIC_BASS_DRUM]i [ACOUSTIC_BASS_DRUM]i
[BASS_DRUM]i [BASS_DRUM]i
[SIDE_STICK]i [SIDE_STICK]i
[ACOUSTIC_SNARE]i [ACOUSTIC_SNARE]i
[HAND_CLAP]i [HAND_CLAP]i
[ELECTRIC_SNARE]i [ELECTRIC_SNARE]i
[LO_FLOOR_TOM]i [LO_FLOOR_TOM]i
[CLOSED_HI_HAT]i [CLOSED_HI_HAT]i
[HIGH_FLOOR_TOM]i [HIGH_FLOOR_TOM]i
[PEDAL_HI_HAT]i [PEDAL_HI_HAT]i
[LO_TOM]i [LO_TOM]i
[OPEN_HI_HAT]i [OPEN_HI_HAT]i
[LO_MID_TOM]i [LO_MID_TOM]i
[HI_MID_TOM]i [HI_MID_TOM]i
[CRASH_CYMBAL_1]i [CRASH_CYMBAL_1]i
[HI_TOM]i [HI_TOM]i
[RIDE_CYMBAL_1]i [RIDE_CYMBAL_1]i
[CHINESE_CYMBAL]i [CHINESE_CYMBAL]i
[RIDE_BELL]i [RIDE_BELL]i
[TAMBOURINE]i [TAMBOURINE]i
[SPLASH_CYMBAL]i [SPLASH_CYMBAL]i
[COWBELL]i [COWBELL]i
[CRASH_CYMBAL_2]i [CRASH_CYMBAL_2]i
[VIBRASLAP]i [VIBRASLAP]i
[RIDE_CYMBAL_2]i [RIDE_CYMBAL_2]i
[HI_BONGO]i [HI_BONGO]i

```

[LO_BONGO]i [LO_BONGO]i
[MUTE_HI_CONGA]i [MUTE_HI_CONGA]i
[OPEN_HI_CONGA]i [OPEN_HI_CONGA]i
[LO_CONGA]i [LO_CONGA]i
[HI_TIMBALE]i [HI_TIMBALE]i
[LO_TIMBALE]i [LO_TIMBALE]i
[HI_AGOGO]i [HI_AGOGO]i
[LO_AGOGO]i [LO_AGOGO]i
[CABASA]i [CABASA]i
[MARACAS]i [MARACAS]i
[SHORT_WHISTLE]i [SHORT_WHISTLE]i
[LONG_WHISTLE]i [LONG_WHISTLE]i
[SHORT_GUIRO]i [SHORT_GUIRO]i
[LONG_GUIRO]i [LONG_GUIRO]i
[CLAVES]i [CLAVES]i
[HI_WOOD_BLOCK]i [HI_WOOD_BLOCK]i
[LO_WOOD_BLOCK]i [LO_WOOD_BLOCK]i
[MUTE_CUICA]i [MUTE_CUICA]i
[OPEN_CUICA]i [OPEN_CUICA]i
[MUTE_TRIANGLE]i [MUTE_TRIANGLE]i
[OPEN_TRIANGLE]i [OPEN_TRIANGLE]i

```

sonification_default_drum_instruments.flac

A.5 Sonification Default Rhythms

Listing 18: Sonification default rhythms

```

V9 [BASS_DRUM]q [BASS_DRUM] [BASS_DRUM]q Rh
[BASS_DRUM]i [BASS_DRUM] [BASS_DRUM]i [BASS_DRUM] [BASS_DRUM]i [
  BASS_DRUM] [BASS_DRUM]i Rh
[BASS_DRUM]s [BASS_DRUM] [BASS_DRUM]s [BASS_DRUM] [BASS_DRUM]s [
  BASS_DRUM] [BASS_DRUM]s [BASS_DRUM] [BASS_DRUM]s [BASS_DRUM] [
  BASS_DRUM]s [BASS_DRUM] [BASS_DRUM]s [BASS_DRUM] [BASS_DRUM]s
  Rh
[BASS_DRUM]q [BASS_DRUM] [BASS_DRUM]i [BASS_DRUM] [BASS_DRUM]i Rh
[BASS_DRUM]i [BASS_DRUM] [BASS_DRUM]i [BASS_DRUM] [BASS_DRUM]q Rh
[BASS_DRUM]i [BASS_DRUM] [BASS_DRUM]q [BASS_DRUM] [BASS_DRUM]i Rh
[BASS_DRUM]q [BASS_DRUM] [BASS_DRUM]s [BASS_DRUM] [BASS_DRUM]s [
  BASS_DRUM] [BASS_DRUM]i Rh
[BASS_DRUM]s [BASS_DRUM] [BASS_DRUM]q [BASS_DRUM] [BASS_DRUM]s [
  BASS_DRUM] [BASS_DRUM]i Rh
[BASS_DRUM]s [BASS_DRUM] [BASS_DRUM]s [BASS_DRUM] [BASS_DRUM]q [
  BASS_DRUM] [BASS_DRUM]i Rh
[BASS_DRUM]s [BASS_DRUM] [BASS_DRUM]s [BASS_DRUM] [BASS_DRUM]i [
  BASS_DRUM] [BASS_DRUM]q Rh
[BASS_DRUM]s [BASS_DRUM] [BASS_DRUM]s [BASS_DRUM] [BASS_DRUM]i [
  BASS_DRUM] [BASS_DRUM]s [BASS_DRUM] [BASS_DRUM]s [BASS_DRUM] [
  BASS_DRUM]i Rh

```

```
[BASS_DRUM]i [BASS_DRUM] [BASS_DRUM]s [BASS_DRUM] [BASS_DRUM]s [
  BASS_DRUM] [BASS_DRUM]i [BASS_DRUM] [BASS_DRUM]s [BASS_DRUM] [
  BASS_DRUM]s Rh
[BASS_DRUM]s [BASS_DRUM] [BASS_DRUM]i [BASS_DRUM] [BASS_DRUM]s [
  BASS_DRUM] [BASS_DRUM]s [BASS_DRUM] [BASS_DRUM]i [BASS_DRUM] [
  BASS_DRUM]s Rh
[BASS_DRUM]i [BASS_DRUM] [BASS_DRUM]i [BASS_DRUM] [BASS_DRUM]s [
  BASS_DRUM] [BASS_DRUM]s [BASS_DRUM] [BASS_DRUM]s [BASS_DRUM] [
  BASS_DRUM]s Rh
[BASS_DRUM]i [BASS_DRUM] [BASS_DRUM]s [BASS_DRUM] [BASS_DRUM]s [
  BASS_DRUM] [BASS_DRUM]s [BASS_DRUM] [BASS_DRUM]s [BASS_DRUM] [
  BASS_DRUM]i
```

sonification_default_rhythms.flac

B Use Cases: Log Files and Sonifications

B.1 Process Mining

B.1.1 Invalid Log - Not well formed

Listing 19: Use cases: overview invalid log not well formed XES

```
<?xml version="1.0" encoding="UTF-8" ?>
<log xes:version="1.0" xes:features="nested-attributes" xmlns="http://www.xes-
  standard.org/">
  <trace>
    <string key="concept:name" value="Case 1" />
    <!-- Missing closing trace -->
  </log>
```

usecase_processmining_invalid_notwellformed.xes

B.1.2 Invalid Log - No time stamp for all events

Listing 20: Use cases: overview invalid log no time stamps XES

```
<?xml version="1.0" encoding="UTF-8" ?>
<log xes:version="1.0" xes:features="nested-attributes" xmlns="http://www.xes-
  standard.org/">
  <extension name="Lifecycle" prefix="lifecycle" uri="http://www.xes-standard.org
    /lifecycle.xesext" />
  <extension name="Time" prefix="time" uri="http://www.xes-standard.org/time.
    xesext" />
  <extension name="Concept" prefix="concept" uri="http://www.xes-standard.org/
    concept.xesext" />
```

```

<extension name="Cost" prefix="cost" uri="http://www.xes-standard.org/cost.
xesext" />
<classifier name="Event Name" keys="concept:name" />
<string key="lifecycle:model" value="standard" />
<trace>
  <string key="concept:name" value="Case 1" />
  <event>
    <date key="time:timestamp" value="2015-05-01T08:00:00.000+01:00" />
    <string key="concept:name" value="offer" />
    <string key="lifecycle:transition" value="start" />
    <float key="cost:total" value="10.0" />
  </event>
  <event>
    <!-- Missing Timestamp -->
    <string key="concept:name" value="offer" />
    <string key="lifecycle:transition" value="complete" />
    <float key="cost:total" value="20.0" />
  </event>
</trace>
</log>

```

usecase_processmining_invalid_notimestamp.xes

B.1.3 Repair example log

Downloaded at 31st of January 2016 as part of
<http://www.promtools.org/prom6/downloads/example-logs.zip>. Unfiltered log
 with 1104 traces.

- repairExample.xes

B.1.4 Repair example cleaned log

Downloaded at 31st of January 2016 as part of
<http://www.promtools.org/prom6/downloads/example-logs.zip>. Filtered log
 with 1000 traces.

- repairExampleSample2.xes

B.1.5 Sonification Control-Flow Perspective

Repair example sonification: default total and event speed

Listing 21: Use cases: process mining control flow 1

```

ConceptName->
Melody
  Analyze Defect=c5maj x
  Archive Repair=g6maj x
  Inform User=d5min x

```

```

        Register=e5min x
        Repair (Complex)=c6maj x
        Repair (Simple)=d6min x
        Restart Repair=e6min x
        Test Repair=f6maj x
Color
        Analyze Defect=red x
        Archive Repair=green x
        Inform User=blue x
        Register=pink x
        Repair (Complex)=yellow x
        Repair (Simple)=gray x
        Restart Repair=magenta x
        Test Repair=orange x
LifecycleTransition->
Instrument
        complete=Piano x
        start=Celesta x
Shape
        complete=Square x
        start=Circle x
traces=all
speed=1.0
eventSpeed=1.0
duration=-
eventDurationSpeed=-
metronome=false

```

- data_prom_controlflow_son_defaulttotaleventspeed.son
- data_prom_controlflow_son_defaulttotaleventspeed.midi

Repair example sonification: speed up 100

Listing 22: Use cases: process mining control flow 2

```

ConceptName->
Melody
        Analyze Defect=c5maj x
        Archive Repair=g6maj x
        Inform User=d5min x
        Register=e6min x
        Repair (Complex)=c6maj x
        Repair (Simple)=d6min x
        Restart Repair=e6min x
        Test Repair=f6maj x
Color

```

```

        Analyze Defect=red x
        Archive Repair=green x
        Inform User=blue x
        Register=pink x
        Repair (Complex)=yellow x
        Repair (Simple)=gray x
        Restart Repair=magenta x
        Test Repair=orange x
LifecycleTransition->
Instrument
        complete=Piano x
        start=Celesta x
Shape
        complete=Square x
        start=Circle x
traces=all
speed=100.0
eventSpeed=1.0
duration=-
eventDurationSpeed=-
metronome=false

```

- data_prom_controlflow_son_speedup100.son
- data_prom_controlflow_son_speedup100.midi

Repair example sonification: single trace

Listing 23: Use cases: process mining control flow 3

```

ConceptName->
Melody
        Analyze Defect=c5maj x
        Archive Repair=g6maj x
        Inform User=d5min x
        Register=e5min x
        Repair (Complex)=c6maj x
        Repair (Simple)=d6min x
        Restart Repair=e6min x
        Test Repair=f6maj x
Color
        Analyze Defect=red x
        Archive Repair=green x
        Inform User=blue x
        Register=pink x
        Repair (Complex)=yellow x
        Repair (Simple)=gray x

```

```

        Restart Repair=magenta x
        Test Repair=orange x
LifecycleTransition->
Instrument
        complete=Piano x
        start=Celesta x
Shape
        complete=Square x
        start=Circle x
traces=1
speed=100.0
eventSpeed=1.0
duration=-
eventDurationSpeed=-
metronome=false

```

- subsec:data_prom_controlflow_son_singletrace.son
- subsec:data_prom_controlflow_son_singletrace.midi
- subsec:data_prom_controlflow_son_singletrace.flac

Repair example sonification: single trace only completed

Listing 24: Use cases: process mining control flow 4

```

ConceptName->
Melody
        Analyze Defect=c5maj x
        Archive Repair=g6maj x
        Inform User=d5min x
        Register=e5min x
        Repair (Complex)=c6maj x
        Repair (Simple)=d6min x
        Restart Repair=e6min x
        Test Repair=f6maj x
Color
        Analyze Defect=red x
        Archive Repair=green x
        Inform User=blue x
        Register=pink x
        Repair (Complex)=yellow x
        Repair (Simple)=gray x
        Restart Repair=magenta x
        Test Repair=orange x
LifecycleTransition->
Instrument
        complete=Piano x

```

```

        start=Celesta
Shape
        complete=Square x
        start=Circle
traces=1
speed=100.0
eventSpeed=1.0
duration=-
eventDurationSpeed=-
metronome=false

```

- data_prom_controlflow_son_singletracecompleted.son
- data_prom_controlflow_son_singletracecompleted.midi
- data_prom_controlflow_son_singletracecompleted.flac

B.1.6 Sonification Case-Related Information

Listing 25: Use cases: process mining case-related

```

ConceptName->
Melody
    Analyze Defect=c5maj x
    Archive Repair=g6maj x
    Inform User=d5min x
    Register=e5min x
    Repair (Complex)=c6maj x
    Repair (Simple)=d6min x
    Restart Repair=e6min x
    Test Repair=f6maj x
Color
    Analyze Defect=red x
    Archive Repair=green x
    Inform User=blue x
    Register=pink x
    Repair (Complex)=yellow x
    Repair (Simple)=gray x
    Restart Repair=magenta x
    Test Repair=orange x
LifecycleTransition->
Instrument
    complete=Piano x
    start=Celesta
Shape
    complete=Square x
    start=Circle

```

```
traces=27, 28, 29
speed=200.0
eventSpeed=1.0
duration=-
eventDurationSpeed=-
metronome=false
```

- data_prom_caserelated.son
- data_prom_caserelated.midi
- data_prom_caserelated.flac

B.1.7 Sonification Organizational-Related Information

Number of people involved in case

Listing 26: Use cases: process mining organizational-related 1

```
TraceName->
Instrument
    18=Electric_Jazz_Guitar x
    15=Bright_Acoustic x
    10=Celesta x
    13=Square x
    3=Steel_Drums x
    12=Electric_Bass_Finger x
    26=Tinkle_Bell x
    2=Contrabass x
    9=Reverse_Cymbal x
    37=0carina x
    ...
Shape
    18=Square x
    15=Star x
    10=Circle x
    3=Cross x
    12=Rhombus x
    26=Rhombus x
    2=Star x
    9=Rhombus x
    37=Circle x
    ...
OrganizationResource->
Melody
    SolverC1=c5maj x
    SolverC2=d5min x
```

```

SolverC3=e5min x
SolverS1=f5maj x
SolverS2=g5maj x
SolverS3=a5min x
System=c7maj x
Tester1=c6maj x
Tester2=d6min x
Tester3=e6min x
Tester4=f6maj x
Tester5=g6maj x
Tester6=a6min x
Color
SolverC1=null x
SolverC2=green x
SolverC3=blue x
SolverS1=pink x
SolverS2=yellow x
SolverS3=gray x
System=magenta x
Tester1=orange x
Tester2=cyan x
Tester3=red x
Tester4=green x
Tester5=blue x
Tester6=pink x
traces=1, 2, 3, 4, 5, 6, 7, 8, 9, 10
speed=100.0
eventSpeed=1.0
duration=-
eventDurationSpeed=-
metronome=false

```

- data_prom_organizationrelated_numberpeople.son
- data_prom_organizationrelated_numberpeople.midi
- data_prom_organizationrelated_numberpeople.flac

Role Transfer

Listing 27: Use cases: process mining organizational-related 2

```

TraceName->
Instrument
18=Electric_Jazz_Guitar x
15=Bright_Acoustic x
10=Celesta x

```

```

13=Square x
3=Steel_Drums x
12=Electric_Bass_Finger x
26=Tinkle_Bell x
2=Contrabass x
9=Reverse_Cymbal x
37=Ocarina x
...
Shape
18=Square x
15=Star x
10=Circle x
3=Cross x
12=Rhombus x
26=Rhombus x
2=Star x
9=Rhombus x
37=Circle x
...
OrganizationResource->
Melody
SolverC1=c5maj x
SolverC2=c5maj x
SolverC3=c5maj x
SolverS1=d5min x
SolverS2=d5min x
SolverS3=d5min x
System=c7maj x
Tester1=c6maj x
Tester2=c6maj x
Tester3=c6maj x
Tester4=c6maj x
Tester5=c6maj x
Tester6=c6maj x
Color
SolverC1=null x
SolverC2=green x
SolverC3=blue x
SolverS1=pink x
SolverS2=yellow x
SolverS3=gray x
System=magenta x
Tester1=orange x
Tester2=cyan x
Tester3=red x

```

```

        Tester4=green x
        Tester5=blue x
        Tester6=pink x
traces=1, 2, 3, 4, 5, 6, 7, 8, 9, 10
speed=100.0
eventSpeed=1.0
duration=-
eventDurationSpeed=-
metronome=false

```

- data_prom_organizationrelated_roles.son
- data_prom_organizationrelated_roles.midi
- data_prom_organizationrelated_roles.flac

B.2 Process Analysis

B.2.1 Low Frequency - Repair example Log

For the low frequency use case the same filtered repair example log file (see Section B.1.4) is used as for the process mining use case, but only the first 100 traces are considered as input. The trimming and sorting is done with a custom program.

- Trimmed and sorted repairExampleSample2.xes: repairExampleTrimmed-Sorted.xes
- Case 1 - increased throughput: repairExampleIncreasedThroughput.xes
- Case 2 - changed solver: repairExampleChangedSolver.xes

B.2.2 Low Frequency Sonification Case 1

Variant 1

All event types.

Listing 28: Use cases: process analysis low frequency
case 1 variant 1

```

ConceptName->
Melody
    Analyze Defect=c5maj x
    Archive Repair=d5min x
    Inform User=e5min x
    Register=f5maj x
    Repair (Complex)=g5maj x
    Repair (Simple)=a5min x
    Restart Repair=b5dim x
    Test Repair=c6maj x
Color

```

```

        Analyze Defect=red x
        Archive Repair=green x
        Inform User=blue x
        Register=pink x
        Repair (Complex)=yellow x
        Repair (Simple)=gray x
        Restart Repair=magenta x
        Test Repair=orange x
LifecycleTransition->
Instrument
        complete=Piano x
        start=Celesta x
Shape
        complete=Square x
        start=Circle x
traces=all
speed=3419.8
eventSpeed=-
duration=1 minute
eventDurationSpeed=0.0
metronome=false

```

- data_proa_lowfrequency_throughput_1.son
- data_proa_lowfrequency_throughput_1.midi
- data_proa_lowfrequency_throughput_1.flac

Variant 2

Only *Register* events.

Listing 29: Use cases: process analysis low frequency
case 1 variant 2

```

ConceptName->
Melody
        Analyze Defect=c5maj
        Archive Repair=d5min
        Inform User=e5min
        Register=f5maj x
        Repair (Complex)=g5maj
        Repair (Simple)=a5min
        Restart Repair=b5dim
        Test Repair=c6maj
Color
        Analyze Defect=red
        Archive Repair=green
        Inform User=blue

```

```

        Register=pink x
        Repair (Complex)=yellow
        Repair (Simple)=gray
        Restart Repair=magenta
        Test Repair=orange
LifecycleTransition->
Instrument
    complete=Piano x
    start=Celesta x
Shape
    complete=Square x
    start=Circle x
traces=all
speed=3419.8
eventSpeed=-
duration=1 minute
eventDurationSpeed=0.0
metronome=false

```

- data_proa_lowfrequency_throughput_2.son
- data_proa_lowfrequency_throughput_2.midi
- data_proa_lowfrequency_throughput_2.flac
- data_proa_lowfrequency_throughput_2.mkv
 (Subtitles data_proa_lowfrequency_throughput_2.srt)

B.2.3 Low Frequency Sonification Case 2

Listing 30: Use cases: process analysis low frequency case 2

```

OrganizationResource->
Instrument
    SolverC1=Piano x
    SolverC2=Celesta x
    SolverC3=Violin x
    SolverS1=Trumpet
    SolverS2=Overdriven_Guitar
    SolverS3=Electric_Piano_2
    System=Glockenspiel
    Tester1=Tremolo_Strings
    Tester2=Oboe
    Tester3=Guitar_Harmonics
    Tester4=Harpsichord
    Tester5=Xylophone
    Tester6=Pizzicato_Strings

```

Shape

```
SolverC1=Square x
SolverC2=Circle x
SolverC3=Star x
SolverS1=Cross
SolverS2=Rhombus
SolverS3=Square
System=Circle
Tester1=Star
Tester2=Cross
Tester3=Rhombus
Tester4=Square
Tester5=Circle
Tester6=Star
```

ConceptName->

Melody

```
Analyze Defect= x
Archive Repair= x
Inform User= x
Register= x
Repair (Complex)=c5 x
Repair (Simple)= x
Restart Repair= x
Test Repair= x
```

Color

```
Analyze Defect=red x
Archive Repair=green x
Inform User=blue x
Register=pink x
Repair (Complex)=yellow x
Repair (Simple)=gray x
Restart Repair=magenta x
Test Repair=orange x
```

traces=all

speed=3939.0

eventSpeed=-

duration=1 minute

eventDurationSpeed=0.0

metronome=false

- data_proa_lowfrequency_changed.son
- data_proa_lowfrequency_changed.midi
- data_proa_lowfrequency_changed.flac

B.2.4 High Frequency Log

The high frequency log is generated with a custom program.

- highFrequencyPaperWithExtraActivity.xes

B.2.5 High Frequency Sonification No Anomaly

Listing 31: Use cases: process analysis high frequency
no anomaly

```
TraceName->
Instrument
    0001=Piano x
    0002=Celesta x
    0003=Violin x
    0004=Trumpet x
    0005=Overdriven_Guitar x
    0006=Electric_Piano_2 x
    0007=Glockenspiel x
    0008=Tremolo_Strings x
    0009=Oboe x
    0010=Guitar_Harmonics x
    ...
Shape
    0001=Square x
    0002=Circle x
    0003=Star x
    0004=Cross x
    0005=Rhombus x
    0006=Square x
    0007=Circle x
    0008=Star x
    0009=Cross x
    0010=Rhombus x
    ...
ConceptName->
Melody
    Browse Articles=c5maj x
    Cancel System=d5min x
    Cancel User=e5min x
    Choose Article=f5maj x
    Cyber Monday Offerings=g5maj x
    End=a5min x
    Order=b5dim x
    Process Availability=c6maj x
    Start Page=d6min x
```

```

Color
    Browse Articles=red x
    Cancel System=green x
    Cancel User=blue x
    Choose Article=pink x
    Cyber Monday Offerings=yellow x
    End=gray x
    Order=magenta x
    Process Availability=orange x
    Start Page=cyan x
traces=1, 2, 3, 4, 5
speed=0.5
eventSpeed=0.5
duration=-
eventDurationSpeed=-
metronome=false

```

- data_proa_highfreq_noanomaly.son
- data_proa_highfreq_noanomaly.midi
- data_proa_highfreq_noanomaly.wav

B.2.6 High Frequency Sonification Case 1

Listing 32: Use cases: process analysis high frequency case 1

```

TraceName->
Instrument
    0001=Piano x
    0002=Celesta x
    0003=Violin x
    0004=Trumpet x
    0005=Overdriven_Guitar x
    0006=Electric_Piano_2 x
    0007=Glockenspiel x
    0008=Tremolo_Strings x
    0009=Oboe x
...
    1648=Star x
    1649=Cross x
    1650=Rhombus x
ConceptName->
Melody
    Browse Articles=c5maj
    Cancel System=a4s x

```

```

Cancel User=e5min
Choose Article=f5maj
Cyber Monday Offerings=g5maj
End=
Order=e5s x
Process Availability=c5s x
Start Page=
Color
Browse Articles=red
Cancel System=green x
Cancel User=blue
Choose Article=pink
Cyber Monday Offerings=yellow
End=gray
Order=magenta x
Process Availability=orange x
Start Page=cyan
traces=all
speed=20.0
eventSpeed=1.0
duration=-
eventDurationSpeed=-
metronome=false

```

- data_proa_highfrequency_case1.son
- data_proa_highfrequency_case1.flac
- data_proa_highfrequency_case1.midi

B.2.7 High Frequency Sonification Case 2

Listing 33: Use cases: process analysis high frequency case 2

```

TraceName->
Instrument
0001=Piano x
0002=Celesta x
0003=Violin x
0004=Trumpet x
0005=Overdriven_Guitar x
...
1645=Flute x
1646=Whistle x
1647=Atmosphere x
1648=Melodic_Tom x

```

```

1649=Contrabass x
1650=Sitar x
Shape
0001=Square x
0002=Circle x
0003=Star x
0004=Cross x
0005=Rhombus x
...
1648=Star x
1649=Cross x
1650=Rhombus x
ConceptName->
Melody
Browse Articles=c5s
Cancel System=a4s x
Cancel User=e5min
Choose Article=f5maj
Cyber Monday Offerings=g5maj
End=a5min
Order=e5s x
Process Availability=c5s x
Start Page=d6min
Color
Browse Articles=red
Cancel System=green x
Cancel User=blue
Choose Article=pink
Cyber Monday Offerings=yellow
End=gray
Order=magenta x
Process Availability=orange x
Start Page=cyan
CostTotal->
Volume
33.52925876150222=10
45.773355288027915=20
46.1097794217456=20
46.8414720396287=21
...
48.49673390965705=22
150.84397151095794=107
151.99408465518607=108
152.5112714963578=108
153.98292766125846=110

```

```

166.02128999524206=120
168.6353713691757=122
170.01501468973086=123
175.0379337180366=127
-
traces=all
speed=71.73895666666667
eventSpeed=-
duration=10 minutes
eventDurationSpeed=0.0
metronome=false

```

```

- data_proa_highfrequency_case2.son
- data_proa_highfrequency_case2.flac
- data_proa_highfrequency_case2.midi

```

B.2.8 High Frequency Sonification Case 3

Listing 34: Use cases: process analysis high frequency case 3

```

ConceptName->
Drum Instrument
  Browse Articles=ACOUSTIC_BASS_DRUM
  Cancel System=BASS_DRUM
  Cancel User=SIDE_STICK
  Choose Article=ACOUSTIC_SNARE
  Cyber Monday Offerings=HAND_CLAP
  End=ELECTRIC_SNARE
  Order=BASS_DRUM x
  Process Availability=CLOSED_HI_HAT
  Start Page=HI_WOOD_BLOCK x
Shape
  Browse Articles=Square
  Cancel System=Circle
  Cancel User=Star
  Choose Article=Cross
  Cyber Monday Offerings=Rhombus
  End=Square
  Order=Circle x
  Process Availability=Star
  Start Page=Cross x
OrganizationResource->
Rhythm
  customer=i x

```

```
        system=i i i i x
Color
        customer=red x
        system=green x
traces=all
speed=358.6947833333333
eventSpeed=-
duration=2 minutes
eventDurationSpeed=0.0
metronome=false
```

- data_proa_highfrequency_case3.son
- data_proa_highfrequency_case3.flac
- data_proa_highfrequency_case3.midi
- data_proa_highfrequency_case3.mkv (Subtitles data_proa_highfrequency_case3.srt)

C ProM Plugins

An evaluation of all documented packages found in <https://svn.win.tue.nl/repos/prom/Documentation/> (updated 3rd May 2015) showed the following results shown in Tables 11, 12 and 13. Descriptions are not available for all plugins. Therefore, the visualization info and a general description are not available for all plugins. But the main interest is the sonification support and searching for typical sonification related terms in the does not get any matches in the source code of the plugins.

Name	Visualization	Description
Compliance framework	yes, violation overview and details	compliance checking
Declare Miner	yes, declare model visualization	works with Declare Models
ETConformance	no	conformance between a log and a petri net
Fuzzy Miner	yes, graph representation fuzzy model	mine fuzzy model from log
ILP-Miner	yes, marked petri net	marked petri net from log
LogMerge	no	merges two logs with AIS algorithm
LTL checker	no	check log against LTL model
Object Versioning Compliance Analysis Plug-in	yes, versioning-annotated workflow net	“verification of versioning compliance between object versioning lifecycles and process models in PLM systems”
MoBuCon LTL	no	monitor cases against business constraints
Murata		simplify petri nets
Animation	yes	animation data structure and generic visualizer
BPNM	yes	
(BPNM visualization)	BPNM support	
COSAImportExport	no	COSA support
EPC	yes	
EPC	EPC support	
EPCCConversion		converters from/to EPC - petri nets
EPMLImportExport	no	Epc Markup Language support
Fuzzy	yes, fuzzy model, metrics...	support for fuzzy model
Log	yes	
log visualization	OpenXES backed log support	
LogAbstractions	yes	abstractions on logs
LogDialog		
LogInsert	yes, box plot	insert missing starts in log
Passage	yes	specific miners
Petrify		Petrify tool
PetriNet2EPC	no	convert PetriNet to EPC
PetriNetReplayer	yes	cost-based Petri-net replayer and a fitness plug
PetriNets	yes	basic support for petri nets
PNMLImportExport	no	PNML support
PomPom	yes	Petri-net/EPC visualizer
PomPom View		related to PomPom
SimpleLogFilter	no	basic simple log filter
TPNImportExport	no	support for TPN file format

Table 11: ProM Plugins 1/3

Name	Visualization	Description
TransitionSystems	no animation is displayed by generic animation visualizer	support for transition systems
TSAalyzer		support for the XPDl file format
TSMIner	yes	mine and visualize transition systems
TSMLImportExport	no	TSML support
Woflan	no	checks wheter a Petri net is a sound workflow net
Workload dependent speeds analysis	no	analyze dependency between workload and processing speed
XESLite	no	alternative implementations for XES standard
XPDLImportExport	no	support XPL file format
YAWLReplay	no	load a YAWL specification and replay log based on this specification
StochasticPetriNets	yes	enrich petri net with stochastic information from a log
GeneticMiner	no	genetic miner for process models from event logs
PNML export		export Reset/Inhibitor, Reset, Inhibitor or plain Petri net to PNML-format
PNML import	yes	import petri net from a PNML file
PPMChart Analysis	yes	“graphically represent the operations in the creation of one process model. The plugin takes as input an event log from the Cheetah Experimental Platform that was converted by the Convert Cheetah Log plugin and splitted by the Split Cheetah Log plugin”
ServiceDiscovery	no	“Given a service S and observed behavior Log of an unknown service interacting with S, the ServiceDiscovery package discovers a high-quality service model C such that S and C interact responsively. Thereby, quality is the ability of C to describe the observed behavior in Log well according to dierent quality dimensions. Responsive interaction of S and C, in turn, ensures that termination of their composition or communication between S and C is always possible.”
SimpleLogOperations	no	- add artificial start/end to log - rename events - make all traces start at the same time
TestBed	yes	generates nets and performs tests on tests on them

Table 12: ProM Plugins 2/3

Name	Visualization	Description
Transition System Miner	no	mine transition system from log
Divide and Conquer		
FeaturePrediction	yes	combine process mining with classification
InductiveMiner	no	mines a marked petri net or process tree from an event log
LTL Checker Plugin	yes	? (something with LTL language)
Pattern Abstractions Plugin	yes	discover abstraction and transform/preprocess the log
Petra	yes	toolset for the analysis of process families
Analyze Behavioral Property of Petri net Plugin	yes	calculate behavioral properties of petri net
Analyze Structural Property of Petri net Plugin	yes	calculate structural properties of petri net
Petrinet net to Coverability Graph Plugin	yes	construct coverability graph for petri net
Petrinet net to Reachability Graph Plugin	yes	construct reachability graph for petri net
Replay a Log on Petri Net for Conformance Analysis Plug-in	yes	
Replay a Log on Petri Net for Performance/- Conformance Plug-in	yes	
Process Tree Properties	yes	add information to formalism
PTMerge		merge process trees
Log Insert start plugin	yes box and whisker diagrams	insert missing start events
Mine SPD Model Plugin	yes	mine SPD (simple precedence diagram) from event log

Table 13: ProM Plugins 3/3

D Source Code

D.1 Standalone Variant

The resources to build and run the standalone variant are available in the *business_process_sonification* folder and can also be found on the public version control repository Github⁴⁵.

Build Preparation

- Copy the following JARs to the lib folder
 - Open XES 2.15
<https://svn.win.tue.nl/trac/prom/export/latest/Releases/OpenXES/OpenXES-20160212.jar>
 - JFugue 5.0.7 <http://www.jfugue.org/jfugue-5.0.7.jar>
 - slickerbox <http://code.deckfour.org/slickerbox/slickerbox-1.0rc1.tar.gz>
(extract TAR)
 - Spex (spex.jar Version 1.1 take from
<http://www.promtools.org/prom6/downloads/prom-6.6-all-platforms.tar.gz>)
 - Widgets (take from
<http://www.promtools.org/prom6/packages66/Widgets/Widgets-6.6.153-all.zip>)
- Other steps
 - Install JDK 1.8+ (Tested with Oracle JDK)
 - Install the build tool SBT <http://www.scala-sbt.org/download.html>
 - Optionally download the soundfont
 - * go to <http://www.schristiancollins.com/generaluser.php> and download the latest version of the soundfont in the Current section of the page (https://dl.dropboxusercontent.com/u/8126161/GeneralUser_GS_1.47.zip)
 - * extract zip and copy *GeneralUser GS v1.47.sf2* to the folder containing the fat JAR

Build

Run *sbt assembly*.

Usage

Go to *target/scala-2.11*, run *java -jar sonification_1.0.5.jar*.

D.2 ProM Package

The resources to build and run the standalone variant are available in the *business_process_sonification_package* folder and can also be found on the public version control repository Github⁴⁶.

⁴⁵ https://github.com/felixamerbauer/business_process_sonification

⁴⁶ https://github.com/felixamerbauer/business_process_sonification_package

Usage

- JRE 1.8+ is required
- download and extract <http://www.promtools.org/prom6/downloads/prom-6.6-all-platforms.tar.gz>
- edit *ProM.ini*
 - replace *PACKAGE_URL* = <http://www.promtools.org/prom6/packages66/packages.xml> with *PACKAGE_URL* = https://raw.githubusercontent.com/felixamerbauer/business_process_sonification_package/master/packages.xml
 - replace *PROM_USER_FOLDER* = with *PROM_USER_FOLDER* = *prom_bps* so that the new ProM installation doesn't interfere with other ProM 6.6 installations on the same machine
- run ProM package manager (*ProMPM66.bat* / *ProMPM66.sh*) and install package *BusinessProcessSonification*
- run *ProM66.bat* / *ProM66.sh* to use the package functionality in ProM
 - make a XES log file available by clicking the import... button on the main screen and select a XES file
 - start the package specific view by clicking the eye symbol button below the description of the imported log file

Build

The build is fairly complex and requires knowledge about the Scala build tool SBT and how a ProM package needs to be structured.

- In the standalone variant Github root dir run *sbt package* and take the resulting JAR in the target directory.
- The ProM package Github root dir contains a *package* subdirectory that contains two files that need to be updated whenever a new version with a new version number is released.
 - update the content of the zip file (e.g. *lib/BusinessProcessSonification-1.0.4-all.zip*) with the new JAR file from above by replacing the *business-process-sonification-core.jar*.
 - update the *packages.xml* to point to the new version of the zip file and update the version number, author and other fields as required.
- The ProM package Github root dir contains another *packages.xml* file. Update this *packages.xml* to point the URL of the *packages.xml* created in the step above. The *packages.xml* from the ProM package Github root dir then needs to be referenced correctly in *ProM.ini* as explained in the usage section above.

E Abstract

E.1 English

Process mining and process analysis can be performed on event logs generated by computer information systems. Visualization techniques help to understand

the log data but it may be difficult to work with large or complex logs or to find small irregularities in the log. Sonification is an alternative way to display data and sonification, being a temporal medium, seems to be a good fit for event logs containing time stamps. Different sonification techniques and technologies are discussed and combined with visualization to implement a multi-modal tool. The suitability of the tool to solve typical process analysis mining and analysis tasks is evaluated.

E.2 German

Prozess-Mining und Prozessanalyse operieren auf von Informationssystemen erzeugten Event-Logs. Visualisierungstechniken helfen diese Logs zu verstehen aber es kann schwierig sein damit größere oder komplexe Logs zu untersuchen oder kleine Unregelmäßigkeiten zu entdecken. Sonifikation ist eine alternativer Weg um Daten darzustellen und da Sonifikation ein zeitliches Medium ist scheint es gut zu Event-Logs mit Zeitstempeln zu passen. Verschiedene Techniken und Technologien werden diskutiert und mit Visualisierung zu einem multimodalem Werkzeug kombiniert. Die Eignung dieses Werkzeugs um typische Prozess-Mining- und Prozessanalyse-Fragestellungen zu lösen wird evaluiert.

F Summary - German

Diese Arbeit hat das Ziel die Eignung eines multimodalen Werkzeugs basierend auf Sonifikation und Visualisierung für das Prozess-Mining und die Prozessanalyse zu untersuchen. Dazu wird zuerst ein Überblick über den Dateninput Event-Logs und die für die Datendarstellung relevanten Sonifikations- und Visualisierungstechniken gegeben.

Die Visualisierung von Event-Logs ist bereits praxiserprobt und der Schwerpunkt dieser Arbeit liegt auf der Untersuchung möglicher Sonifikationstechnologien für das generische Event-Log-Format XES. Event-Logs beinhalten oft Zeitstempel und diese Information kann genutzt werden um den wichtigen zeitlichen Aspekt für die Sonifikation zu steuern. Andere verfügbare Informationen im Log wie Prozesszugehörigkeit und nominale und kontinuierliche Daten können ebenfalls genutzt werden um die Ausgabesonifikation zu erzeugen. Die Visualisierung zusammen mit einem geeigneten, den Analyse- und Mining-Workflow unterstützenden, User-Interface sollen beim Erstellen des Mapings und dem Arbeiten mit den Sonifikationen helfen.

Die Literaturrecherche brachte nur Ergebnisse für Teilbereiche dieser Aufgabenstellung und diese Erkenntnisse zusammen mit eigenen Ideen führten zu einer Earcon-basierten Sonifikation und einer zweidimensionalen Visualisierung basierend auf dem Event-Zeitstempel und der Prozesszugehörigkeit. Die Überprüfung und Verbesserung des konkreten Ansatzes erfolgte mit exemplarischer Anwendungsfällen für das Prozess-Mining und die Prozess-Analyse.

Die Implementierung basiert auf dem erweiterbaren, XES Logs, Prozess-Mining und Prozess-Analyse unterstützenden Tools ProM. Da aber für die Vi-

sualisierung und Sonifikation nicht auf konkreten ProM-Fähigkeiten aufgebaut wird ist die Ausführung des entstandenen Tools auch ohne ProM möglich.

Die Auswertung der Anwendungsfälle zeigt die prinzipielle Eignung des gewählten Ansatzes. Es können noch keine Ergebnisse erzielt werden die ohne Sonifikation nicht möglich wären aber für die konkreten Sonifikationsfähigkeiten können Aussagen über besser und weniger gut geeignete Auswertungsziele und Eigenschaften der Event-Logs getroffen werden. Konkrete Vorschläge für mögliche Erweiterungen und alternative Ansätze für Sonifikation, Visualisierung und UI werden kurz vorgestellt.