Requirement Specification Document

Label Refinement by Behavioral Similarity

Document owners:
Bianka Bakullari
Christopher Beine
Nicole Ventsch
Juan

Last edited: April 27, 2019

Contents

1 Project Drivers

1.1 The Purpose of the Project

The purpose of the project is providing a platform that allows refinements of event logs, where actions that originally had the same label are relabelled if they have different structural behaviours. In order to do so, an interactive web service should be implemented that allows users to upload event logs in the standard XES or CSV format and set thresholds for the refinements. This event log should contain actions that are carried out multiple times and are named identically. The web service should then apply an algorithm to it that refines the log in a way that these actions are relabelled based on structural similarity. Finally, the user should be able to download the refined log in XES.
The users can then use the refined log to gain insight by applying process discovery algorithms to the data. By using the refinement algorithm, higher accuracies should be reached and that way lead to a better insight into the processes.

1.2 The Client, the Customer and other Stakeholders

The client of this project is the Chair of Process and Data Science at the RWTH Aachen, which provides the supervision and support for this project. Furthermore, there are many different potential users. For example, students or researchers could use the website as a preprocessing step in order to continue research with the refined log. Additionally, business analysts from companies could use it in order to get a better insight from the refined log than from the original one. Since we will provide a web service, it is easily accessible and not limited to be used by a prespecified group of people.

2 Project Constraints

2.1 Mandated Constraints

2.2 Naming Conventions and Definitions

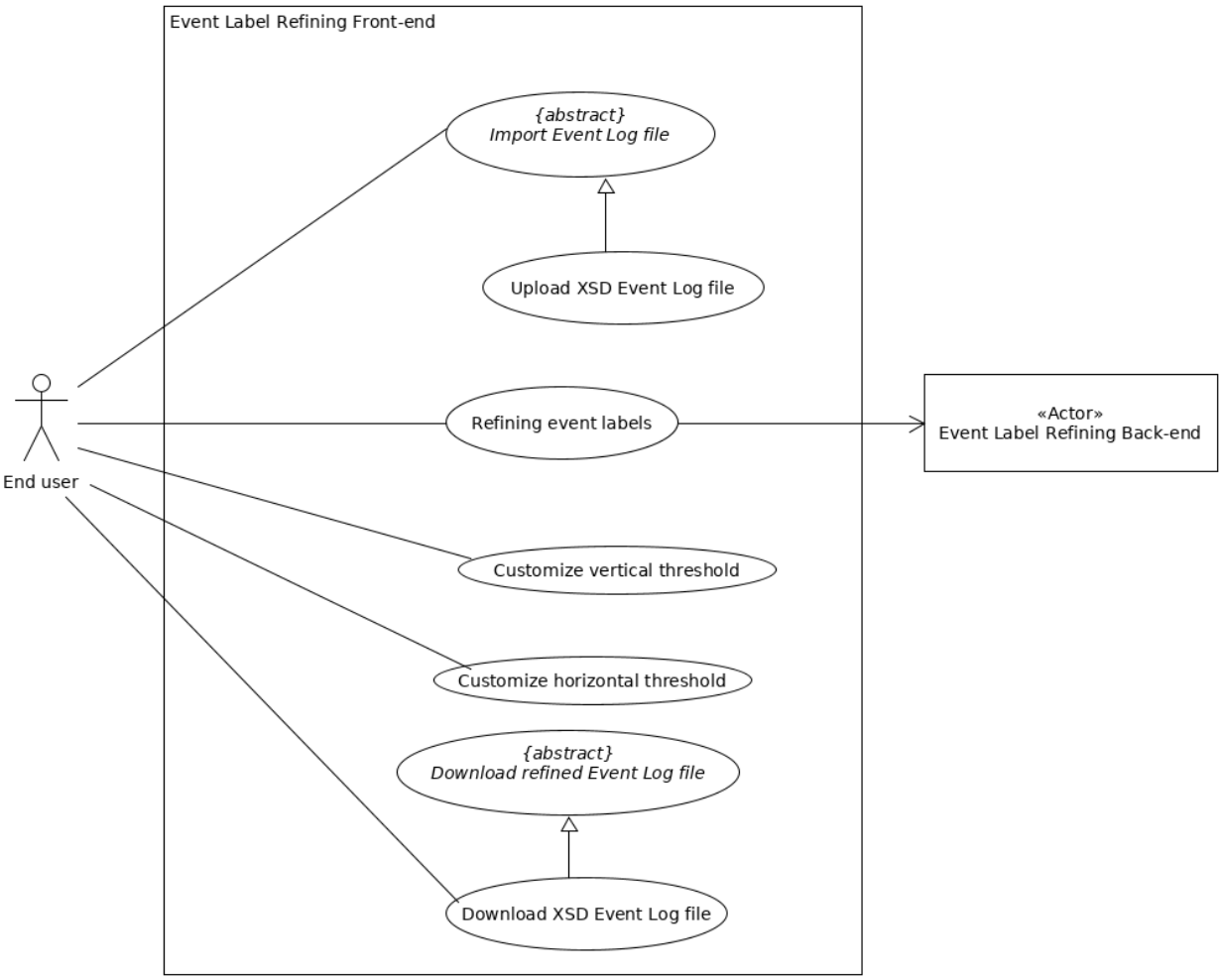
2.3 Relevant Facts and Assumptions

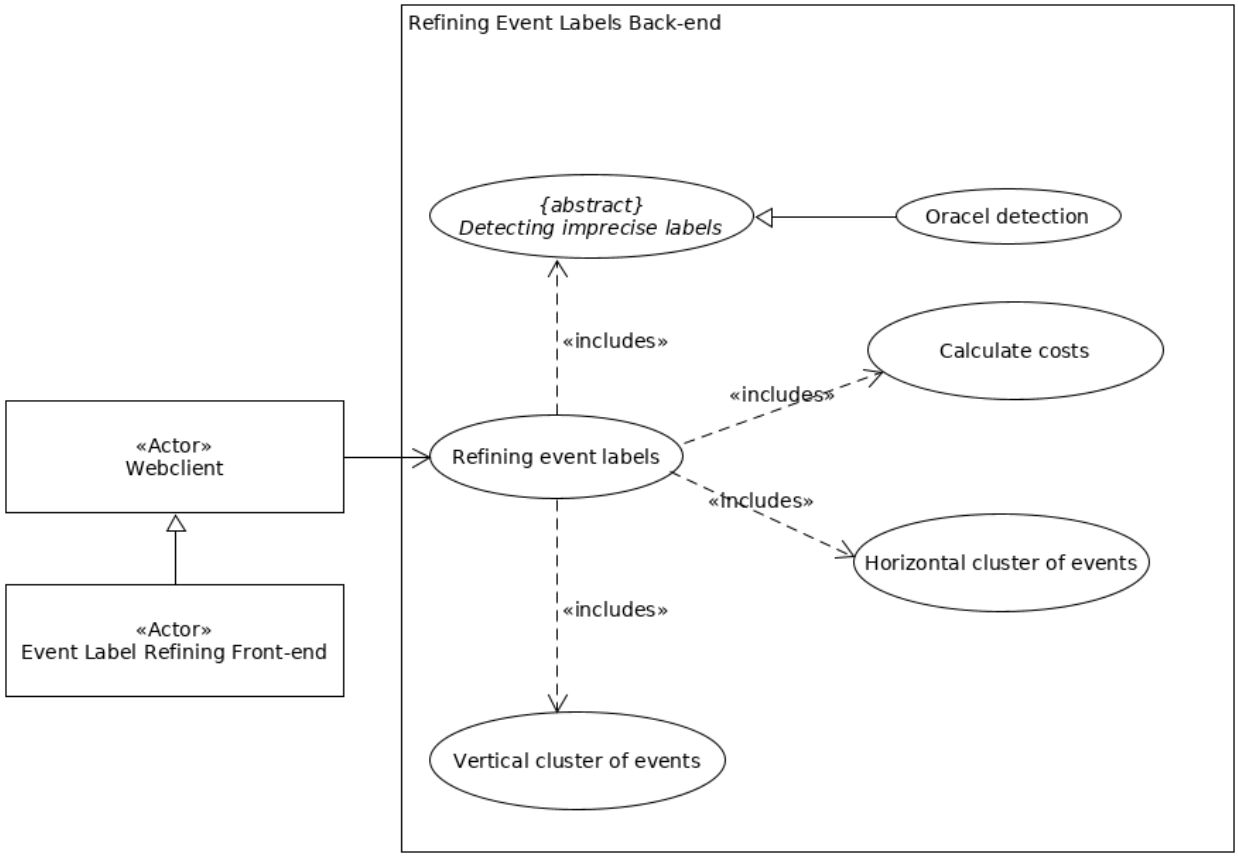
3 Functional Requirements

3.1 The Scope of the Work

3.2 The Scope of the Product

The final product consists of a web interface where the user can upload event logs, parameterize thresholds and then the Refining Event Labels algorithm is executed. In order to get an overview of the product, a use case analysis was carried out, where we identify the system actors and their functions. Because the project's focus is the Refining Event Labels implementation, the project was divided into two systems. This allows a better visualisation of the individual algorithm steps and system interactions. The following use cases were identified within the analysis.


Event Label Refining Front-end
{abstract}
Import Event Log file
Upload XSD Event Log file
Refining event labels
«Actor»
Event Label Refining Back-end
End user
Customize vertical threshold
Customize horizontal threshold
{abstract}
Download refined Event Log file
Download XSD Event Log file



**** under construction ***

3.3 Functional and Data Requirements

Requirement #1: Use case: *Back End*

Description: *Implement distance function between all pairs of traces.*

Details: *Given a pair of traces, calculate a cost function with three weighted components.*

Fit Criterion: *n^(n-1) values in range [0,1].*

Requirement #2: Use case: *Back End*

Description: *Cluster the traces.*

Details: *A threshold for the cost function decides which traces are clustered together.*

Fit Criterion: *A partitioning of the traces into sets.*

Requirement #3: Use case: *Back End*

Description: *Refining labels accros variants*

Details: *Each imprecise label has the same identifier in traces of the same cluster and different identifiers in traces of different clusters.*

Fit Criterion: *Precise labels remain unchanged, relabeling equivalent to variants of traces.*

Requirement #4: Use case: *Back End*

Description: *Refining labels within variant*

Details: *An unfolding threshold is used to decise if and where relabeling takes place within a trace.*

Fit Criterion: *Identical traces must get identical relabeling, dissimilar traces remain dissimilar.*

Requirement #5: Use case: *Front End*

Description: *Upload event log.*

Details: *A clear sign showing where the user should click for upload.*

Fit Criterion: *Files with the right format are loaded successfully, otherwise return error message.*

Requirement #6: Use case: *Front End*

Description: *Set thresholds for the algorithm.*

Details: *The selected threshold appears on the screen.*

Fit Criterion: *Thresholds within right range are accepted successfully. Otherwise, return error message.*

Requirement #7: Use case: *Front End*

Description: *Download refined log.*

Details: *The user chooses to download the refined log through a click.*

Fit Criterion: *Download is successful and the refined log has XES format.*

4 Nonfunctional Requirements

4.1 Look and Feel Requirements

4.2 Usability and Humanity Requirements

4.3 Performance Requirements

4.4 Operational and Environmental Requirements

4.5 Maintainability and Support Requirements

4.6 Security Requirements

4.7 Cultural and Political Requirements

4.8 Legal Requirements

5 Project Issues

5.1 Open Issues

Our team has a rough idea of the steps needed to implement the algorithm and the interface. However, we are unsure of the relative time needed for each of these parts. Many unexpected bugs in our algorithm could hinder the time we have in our disposal to design a user-friendly web-interface.
While we will try to achieve writing an efficient algorithm, there are no concrete performance measures required. Drawbacks will probably be found during implementation and decisions will have to be made on the spot.
Since each of the members has to be fully aware of and actively participate in the coding process, it is still unclear how the functionalities will be divided between the group members. Whether or not some functionalities should be implemented together or individually depends on type and importance of functionality, personal schedules, deadlines and so on.

5.2 Off-the-Shelf Solutions

There has already been an implementation of the algorithm in ProM. This might be helpful in the designing steps of our algorithm.
We will use a code written in Java to automatically generate events logs which we will then use as small tests for our code.
@juan: Where did you get the java code you showed us once?

5.3 New Problems

The Label Refinement algorithm intends to give the user an alternative event log on which process discovery algorithms can be applied. Whether or not the models resulting from the new refined log have better precision or fitness compared to the original models is beyond our scope. Since the result also depends on the thresholds and set of candidate activities which are chosen by the user, we assume that the user has some background on Process Mining and has clear intentions for trying to work with a new refined log. The algorithm does not filter out events or features in the data.
Optimally, the new models resulting from the refined event log should keep the same old labels even for the refined ones. This requires from the process discovery algorithm itself to compare the original event log with the refined one and then map the refined labels to their old names after having obtained the refined model. We consider this beyond our scope. Since the set of new labels to use for refinement is unspecified, our implementation will assign new endings to the original label, so that the context is still clear even with a model containing the new labels.
A common mistake the user could make is upload an event log in some other format (e.g. .xlsx format) instead of XES or CSV format.

5.4 Tasks

5.5 Migration to the New Product

5.6 Risks

Risks	Description	Category	Mitigation
Inaccurate expectations.	Stakeholders develop inaccurate expectations (believe that the project will achieve something not in the requirements, plan, etc.).	Stakeholder	Clearly state in the requirement documentation what are the deliverables meant to be done and the scope of the project.
Process inputs are low quality.	Inputs from stakeholders that are low quality (e.g. business case, requirements, change requests).	Stakeholder	Kindly ask the stakeholder for a more detailed and clearer version of any input they may provide i.e., requirements, business cases.
Misunderstood requirements.	When requirements are misinterpreted by the project team.	Communication	Meet with the stakeholders and discuss the requirements again until the team is sure that they have completely understood them.
Learning curves.	Project team needs to acquire new skills for the project.	Team	Motivate the project team, give them the best practices on the IT field and make experts instruct them using their knowledge and own experience.
Integration failure.	Product components will fail to integrate with each other.	Integration	Establish standards for product development and make sure that the individual components passed flawlessly the unit test.
Requirements are incomplete.	Requirements are not fully captured or are overlooked.	Requirements	Make a peer-review of the requirement documentation and make sure that nothing is being left out.

5.7 Costs

5.8 User Documentation

1. Technical documentation:
   - Software code documentation.
   - Technical specifications.
2. User documentation including:
   - How to use the UI.
   - Examples of inputs and outputs.
   - Explanation of error messages.
   - Information to contact the developers (in case of further questions).

5.9 Waiting Room

- Additional feature which enables the user to choose a Business Process Model Discovery (BPMD) technique to visualize the resulting process model and to pick the one which is considered to be the best one (according to user's expertise).
- Additional feature that allows for the automatic detection of "imprecise labels" by using properties of the Inductive Miner (IM).

References

[1] Lu, Xixi, et al. "Handling duplicated tasks in process discovery by refining event labels." International Conference on Business Process Management. Springer, Cham, 2016.