

Software Design Specification

Label Refinement by Behavioral Similarity

Document owners:

Bianka Bakullari
Christopher Beine
Nicole Ventsch
Juan Garza

Last edited: May 10, 2019

1	Introduction	1
1.1	System Overview	1
1.2	Design Map	1
1.3	Supporting Materials	1
1.4	Definitions and Acronyms	1
2	Design Considerations	1
2.1	Assumptions and Dependencies	1
2.2	General Constraints	1
2.3	Goals and Guidelines	1
2.4	Development Methods	1
3	Architectural Strategies	1
4	System Architecture	1
4.1	Client Machine	2
4.2	Front-end	2
4.3	Refining Label API	2
4.4	File Store	2
4.5	Refining Event Labels	2
4.6	Event Log Converter	3
5	Policies and Tactics	4
6	Detailed System Design	4
6.1	Module 1.1: File Converter	4
6.2	Module 1.2: Preprocessing Log	4
6.3	Module 2	5
7	User Interface Design	5
7.1	Application Control	5
7.2	The Screens	5
7.2.1	Screen 1	6
7.2.2	Screen 2	6
7.2.3	Screen 3.1	7
7.2.4	Screen 3.2	8
7.2.5	Screen 4	9

1 Introduction

1.1 System Overview

1.2 Design Map

1.3 Supporting Materials

1.4 Definitions and Acronyms

2 Design Considerations

2.1 Assumptions and Dependencies

2.2 General Constraints

2.3 Goals and Guidelines

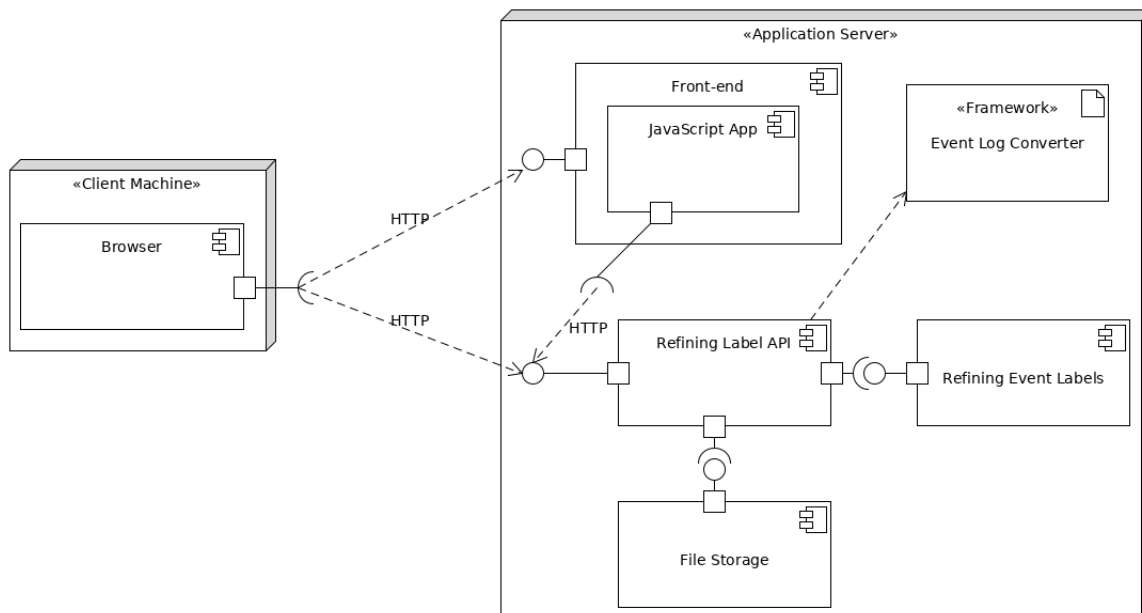
2.4 Development Methods

3 Architectural Strategies

**** under construction ****

- Factory Pattern
- Decorator (Variaton)

4 System Architecture



4.1 Client Machine

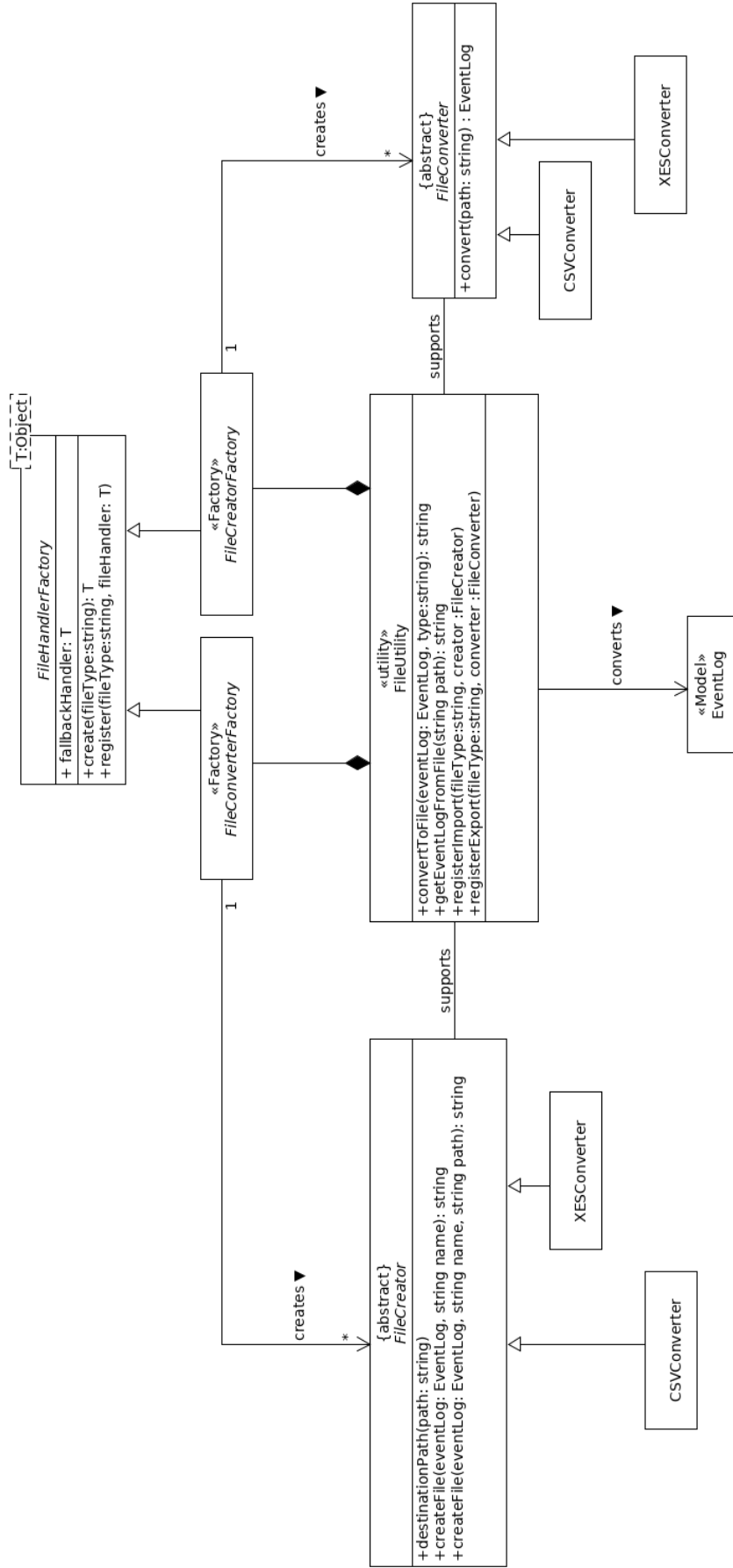
4.2 Front-end

4.3 Refining Label API

4.4 File Store

4.5 Refining Event Labels

4.6 Event Log Converter



5 Policies and Tactics

6 Detailed System Design

The main algorithm "Refining Event Labels" will be split up into multiple modules that contain the main parts of the algorithm. These modules will be explained in detail in the following subsections.

6.1 Module 1.1: File Converter

Name: File Creator

Type: module

Description: This module is responsible for creating a table from the data the user uploads. The event log provided by the user is read and stored internally containing all the original columns.

Attributes: None

Resources: None

Operations:

Name: `get_event_log_from_file()`

Arguments: path to a log file in csv format or XES format

Returns: event log in XES format

Description: The file at the path the user enters is read and stored in XES format. In case the file is originally in csv format, the file is first converted to XES format and then stored.

Precondition: the path provided by the user leads to a csv file or an XES file

Postcondition: the table is stored internally as an XES file

Exceptions: None

Name: `read_XES()`

Arguments: file in XES format

Returns: event log

Description: The XES file is read and stored as a table containing the original columns.

Precondition: an XES file was created using `get_event_log_from_file()`

Postcondition: a table containing all original columns is stored internally

Exceptions: None

6.2 Module 1.2: Preprocessing Log

Name: Preprocessing Log

Type: module

Description: This module is responsible for preprocessing the data. It is checked if the table provided by the user contains the right variables and an error is produced if it does not have. Moreover, a table containing all unique traces and the list of IDs corresponding to these traces is created.

Attributes: None

Resources: None

Operations:

Name: `check_event_log()`

Arguments: table imported using `read_XES()`

Returns: Boolean (True or False)

Description: It is checked whether the table is actually an event log, i.e., whether it contains an activity column, an ID column and a time stamp column. If these exist in the table, "True" will be returned, otherwise "False" will be returned.

Precondition: a file was created using `read_XES()`

Postcondition: if "True" is returned, the file is actually an event log, otherwise the file does not contain the necessary variables and an error will be displayed

Exceptions: None

Name: `get_variants()`

Arguments: table

Returns: lookup table containing a "variants" column and an "ID" column

Description: The table is converted, so that we get a table containing the unique variants for the IDs ordered by their time stamp and the set of IDs corresponding to the variant. In this lookup table, the variants are stored as arrays and the corresponding IDs as a list.

Precondition: the provided file is an event log

Postcondition: a lookup table is stored containing the variants and IDs corresponding to those variants

Exceptions: None

6.3 Module 2

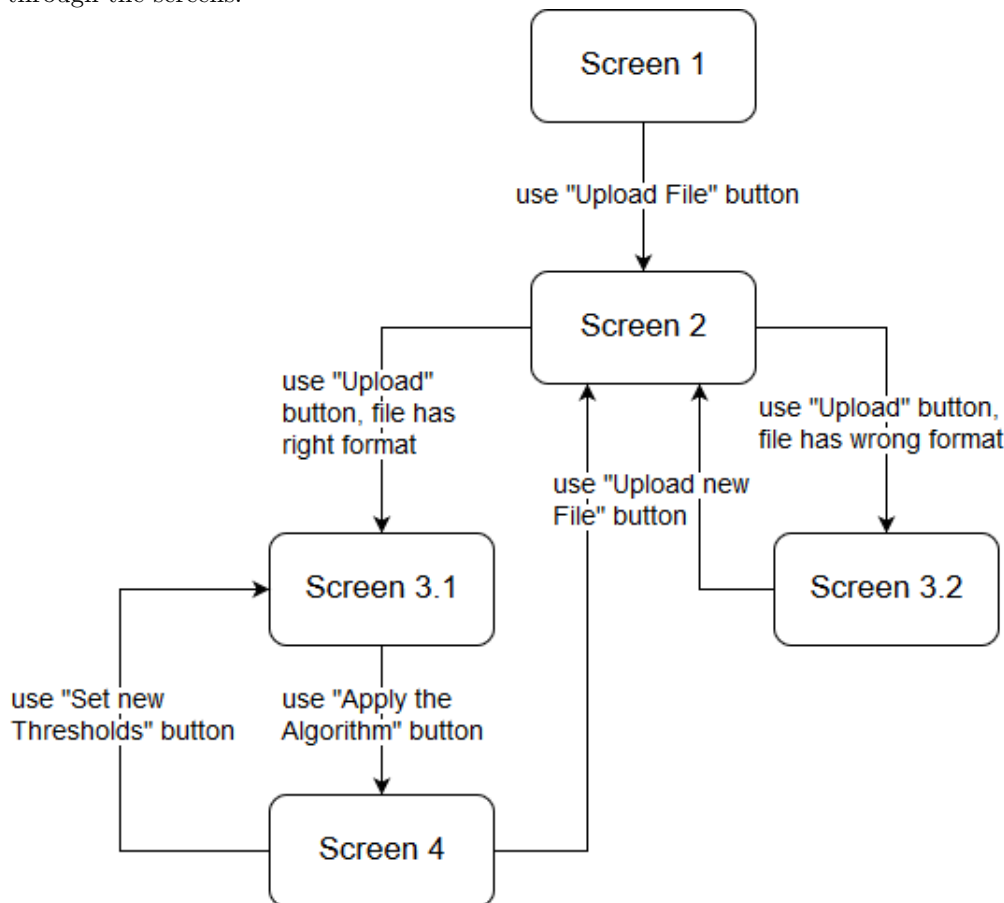
7 User Interface Design

7.1 Application Control

In the project, we will implement a web service. The web client will have a rather plain design that should focus on the main activities the service should provide, which are uploading event logs in csv or XES format, setting the threshold for the label refinement algorithm and download the refined log after the algorithm is finished. There will be buttons used to upload the file, apply the algorithm and download the refined event log. Moreover, the screens will have short explanations telling the user what to do (if not self-explanatory). For setting the thresholds, two boxes will be provided that include the default thresholds, but new values can be entered by the user. A draft of each of the main screen can be found in the next section, section 6.2.

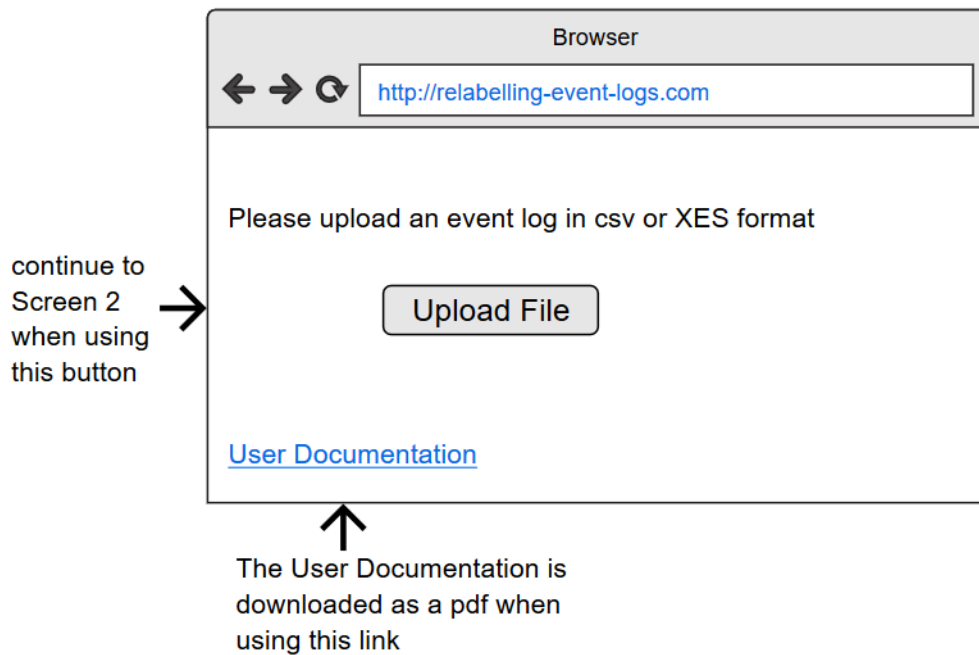
7.2 The Screens

The main screens will be visualized in the following subsections. In these screens include the main functionalities, which are described in the former section. The following diagram will show the flow of control through the screens.



7.2.1 Screen 1

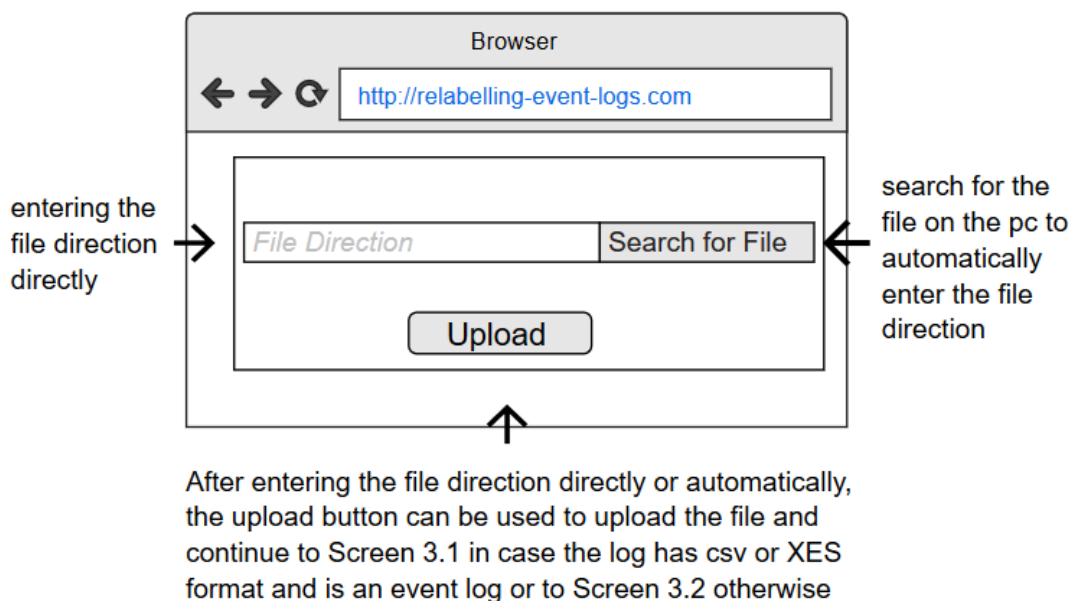
Screen 1:



The first screen visible to the user will show a description saying that an event log in csv or XES format should be uploaded. Moreover, a button "Upload File" is visible. By using this button, the user will continue to Screen 2. At the end of the page, there will be a link called "User Documentation". By clicking on this link, the User Documentation will be downloaded in pdf format.

7.2.2 Screen 2

Screen 2:



In the second screen visible to the user, the user can enter the file direction of the event log he wants to

upload. He can either directly type in the direction into the "File Direction" field or use the "search for File" button to search for a file on his pc, so that the direction will automatically be filled in after selecting a file. After using one of this alternatives, he can use the "Upload" button to upload the file with the given directory. In case this file is an event log, i.e., the data contains at least the attributes "id", "time stamp" and "activity name", and has either csv or XES format, the user will continue to Screen 3.1. If one of these conditions is not satisfied, he will continue to Screen 3.2.

7.2.3 Screen 3.1

Screen 3.1:

Browser

← → ↻

Please enter the names of the candidate activities for the refinement, the cost function threshold, the variant threshold and the unfolding threshold the algorithm should use:

Candidate activity names:

Variant threshold:

Unfolding threshold:

← The user can enter the names of the activities that should be relabelled in the white box

← The user can enter the thresholds the algorithm should use in the white boxes (use default values if no threshold is provided by the user)

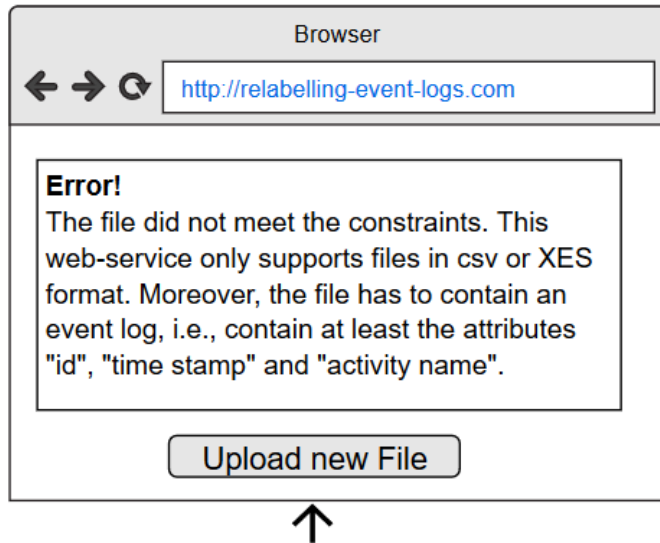
↑

By using this button, the algorithm will be applied to the uploaded event log using the thresholds provided above in order to refine the activity names entered. If the user does not enter a threshold, the default value of 0.05 or 0.60 will be used respectively. When the algorithm is finished, the user will get to Screen 4.

This screen appears if the file uploaded by the user meets the requirements. In this screen, the user can set the thresholds for the algorithm, i.e., the variant and the unfolding threshold. He can enter these in the corresponding white boxes. If he does not enter the thresholds, the default values of 0.05 and 0.60 will be used respectively. Using the button "Apply the Algorithm", the web service will start applying the algorithm using the provided thresholds. After the algorithm is finished, the user will get to Screen 4.

7.2.4 Screen 3.2

Screen 3.2:

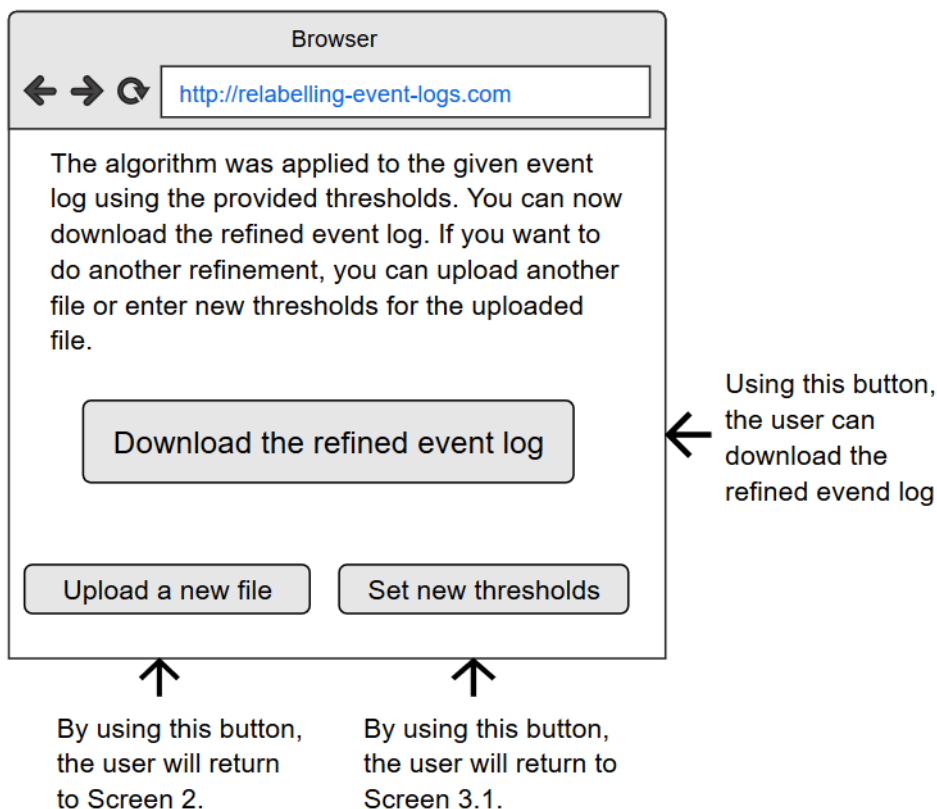


By using this button, the user will return to Screen 2.

This screen appears if the upload was not successful because the file did not meet the assumptions. If the file does not have the right format, the user can click on the button "Upload new File" to return to Screen 2 and upload a file that meets the constraints.

7.2.5 Screen 4

Screen 4:



This Screen will be shown after finishing the algorithm. The user can now download the refined log using the corresponding button. After this step, the user is done and can exit the page, but if he also wants to apply the algorithm to another event log or to the same event log using different thresholds, he can use the corresponding buttons and will be redirected to Screen 2 or Screen 3.1 respectively.

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.