BPI Challenge 2016 (Business Process Intelligence)

Samir Ghoudrani

December 2017

What is BPI Challenge ?

Since 2011, the IEEE Task Force on process mining organizes a yearly Business Process Intelligence Challenge, or BPI Challenge. The goal of this challenge is to bring together practitioners and researchers in the field to show the direct impact of academic work when facing the challenges real-life cases bring.

For 2016, the data was provided by UWV (Employee Insurance Agency), a Dutch autonomous administrative authority (ZBO) which is commissioned by the Ministry of Social Affairs and Employment (SZW) to implement employee insurances and provide labor market and data services in the Netherlands.

Problems to solve: understand customer behaviour accross chanels

UWV (Employee Insurance Agency) is interested in insights on how their channels are being used, when customers move from one contact channel to the next and why and if there are clear customer profiles to be identified in the behavioral data.

What is the typical customer journey ? (e.g. ~ 10 clicks on website -> 2 messages through workflow -> then a phone call -> then a potential complain)
Do each customer use a unique chanel, or multiple chanels ?
In case of multiple chanels, is there an order / journey following a particular pattern ?
Do all these behaviours vary according to the type of customer ? (age, gender, type of claim…) ?
Do all these behavious vary according to the resources / desks handling the demand from the company side ?
Any other insights from all this log data ?

Furthermore, recommendations are sought on how to serve customers without the need to change the contact channel.

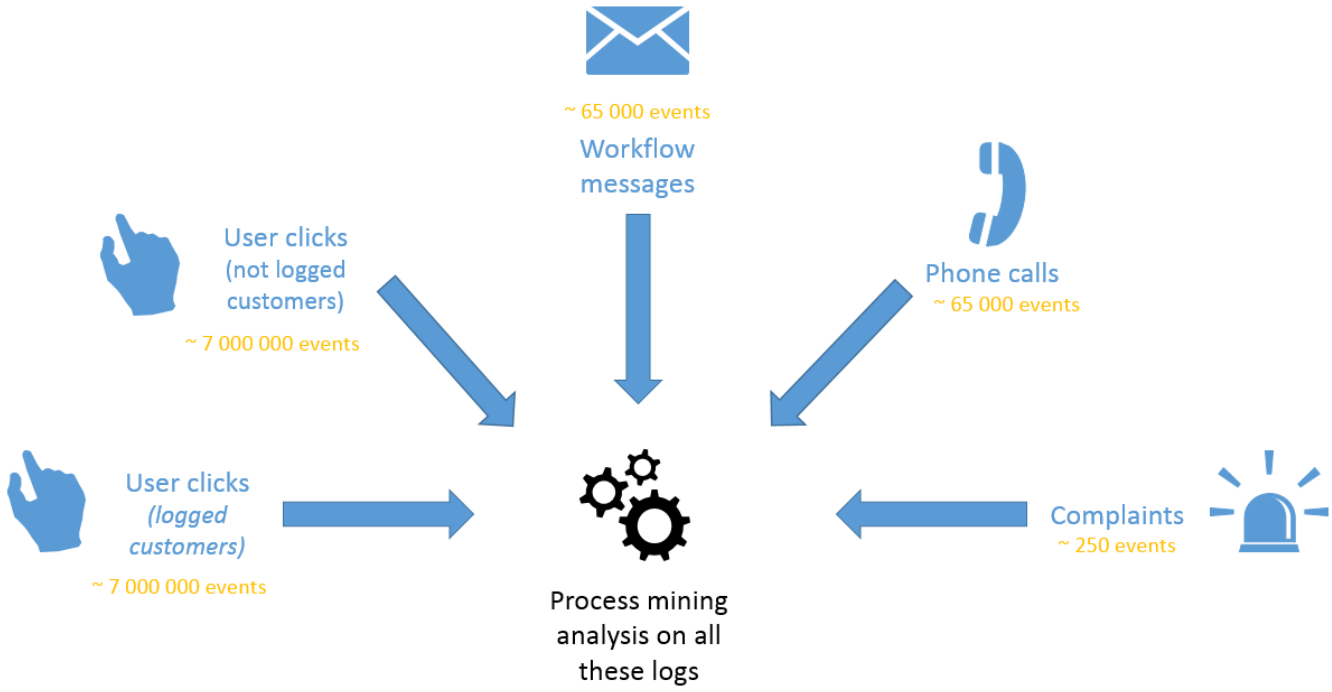
What data do we have ?

The data in this collection pertains to customer contacts over a period of 8 months. The data is focused on customers in the WW (unemployment benefits) process.

Data has been collected from several different sources, namely:

1. Clickdata from the site www.werk.nl collected from visitors that were not logged in,
2. Clickdata from the customer specific part of the site www.werk.nl (a link is made with the customer that logged in),
3. Werkmap Message data, showing when customers contacted the UWV through a digital channel,
4. Call data from the callcenter, showing when customers contacted the call center by phone,
5. Complaint data showing when customers complained. All data is accompanied by data fields with anonymized information about the customer as well as data about the site visited or the contents of the call and/or complaint.

The full dataset is available from https://data.4tu.nl/repository/uuid:360795c8-1dd6-4a5b-a443-185001076eab (https://data.4tu.nl/repository/uuid:360795c8-1dd6-4a5b-a443-185001076eab).



Example of log and intuition about the data

This is very important for a process mining tool: the 'CASE', a way to identify each instance of the process (a case is composed of several events, each line being an event)

Theoratically process discovery algorithms only needs ordered events, but we have several sources => when we will 'join' on CustomerID we will loose the 'ORDER'.. unless we use accurate timestamps to order to resulting log (some pre-processing with R will be needed)

CustomerID	AgeCategory	Gender	Office_U	Office_W	SessionID	IPID	TIMESTAMP	VHOST	URL_FILE	PAGE_NAME	Activity
2025826	50-65	V	313	313	12956475	620841	2015-10-05 10:12:56.880000000	www.werk.nl	/werk_nl/werkn	50plus	
2025826	50-65	V	313	313	13243433	620841	2015-09-30 15:14:35.943000000	www.werk.nl	/werk_nl/werkn	50plus	
1503890	30-39	V	247	247	14805466	1690840	2015-09-01 19:35:06.707000000	digid.werk.nl	/portal/page/po	aanvragen-tw	
2063574	50-65	M	296	301	12710639	1632512	2015-11-06 10:47:42.137000000	www.werk.nl	/werk_nl/werkn	50plus	
2185161	18-29	V	327	327	44281847	757955	2016-01-11 18:44:07.877000000	www.werk.nl	/werk_nl/werkn	bijstandsuitkeri	

(CustomerID is the key to join all 'logs'. Note that we only address here logged users, a second study will be needed for non logged users => the join would be achieved based on IP addresses)

We will hand-write activity for each log to be usable: either 'Phone Call', 'Message' or 'Click'

(of course, for each log, we set the value of the action captured by the log, ex: 'phone call' for the phone call log)

Mandatory variables for a process mining tool to run its basic algorithms (key statistics, process discovery)

'Customer variables' we will use in further analysis, to assess how customer journey varies according to customer typology

'Service Desk variables' we will use in further analysis, to assess how customer journey varies according to resources in the organization

Approach and tools used

Step 1	Explore logs and identify 'case', 'activity' and 'timestamp' Pre-process data to be understandable by a process mining tool
Step 2	Import data to ProM and run the adequate plugins ⇒ Key statistics ⇒ Process Discovery with Heuristic Miner algorithm
Step 3	Interpret the results of so far analysis (see comments with * prefix in next screenshots) Identify possible further more advanced analysis

Step 1: pre-processing with R

Process Mining tools need the data to be in a specific format, so we need some pre-processing with R.

```
## Read csv data
## clicks_not_logged <- read.csv(file = "BPI2016_Clicks_NOT_Logged_In.csv", sep = ";") // ana
lysis for not logged users will be achieved later

##clicks_yes_logged <- read.csv(file = "BPI2016_Clicks_Logged_In.csv", sep = ";")
##phone_calls <- read.csv(file = "BPI2016_Questions.csv", sep = ";")
##workflow_messages <- read.csv(file = "BPI2016_Werkmap_Messages.csv", sep = ";")

##complaints <- read.csv(file = "BPI2016_Complaints.csv", sep = ";") ## Only 250 events

#############################################################################################

## Simplify complaints
##complaints_prom <- data.frame(customerID= complaints$CustomerID, activity = "complaint", da
te = complaints$ContactDate)
## hist(complaints$ContactChannelID) ## mostly channel 8

## Simplify phone_calls
##phone_calls_prom <- data.frame(customerID = phone_calls$CustomerID, activity = "phone_cal
l", date = phone_calls$ContactDate)

## Simplify workflow_messages
##workflow_messages_prom <- data.frame(customerID = workflow_messages$CustomerID, activity =
 "workflow_message", date = workflow_messages$EventDateTime)

## Simplify clicks_yes_logged
##clicks_yes_logged_prom <- data.frame(customerID = clicks_yes_logged$CustomerID, activity =
 "clicks_in_website", date = clicks_yes_logged$TIMESTAMP)

##date <- as.character(phone_calls$ContactDate)
##hour <- as.character(phone_calls$ContactTimeStart)
##full_date <- paste(date, hour, sep=" ")

##phone_calls_prom$date <- full_date

## Check variables
##names(clicks_yes_logged_prom)
##names(workflow_messages_prom)
##names(phone_calls_prom)
##names(complaints_prom)

## Check dates
##clicks_yes_logged_prom$date[1]
##workflow_messages_prom$date[1]
##phone_calls_prom$date[1]
##complaints_prom$date[1]

##clicks_yes_logged_prom$date <- as.character(clicks_yes_logged_prom$date)
##workflow_messages_prom$date <- as.character(workflow_messages_prom$date)

## Now we convert dates to date format
##clicks_yes_logged_prom$date <- as.POSIXlt(clicks_yes_logged_prom$date)
##workflow_messages_prom$date <- as.POSIXlt(workflow_messages_prom$date)
##phone_calls_prom$date <- as.POSIXlt(phone_calls_prom$date)
```

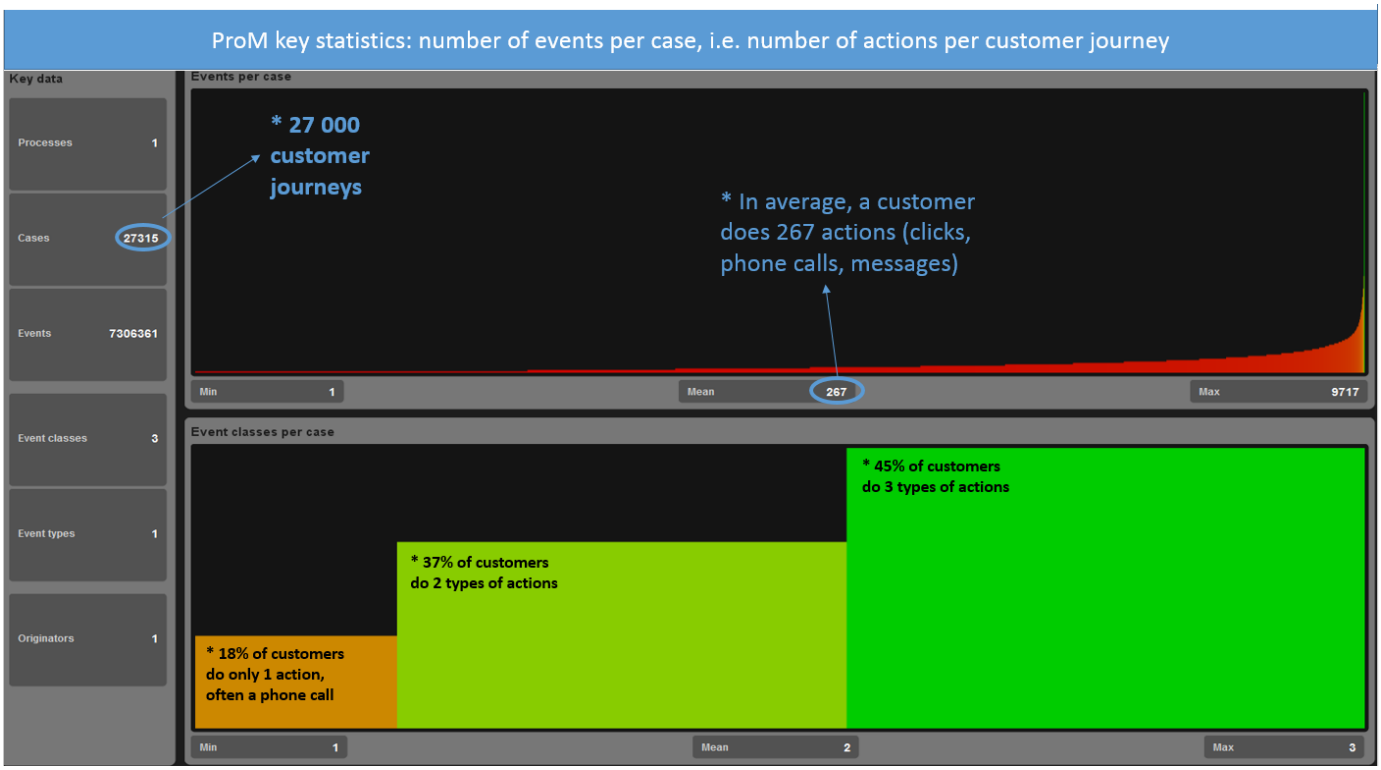
```
## Merge dataframes and sort by dates

##customers_journeys <- rbind(clicks_yes_logged_prom, workflow_messages_prom, phone_calls_pro
m)
##customers_journeys <- customers_journeys[order(customers_journeys$date, decreasing = FALS
E),]

##head(customers_journeys)

##write.csv(customers_journeys, file = "customers_journeys.csv")
```

Step 2: Analysis with ProM tool



ProM key statistics: all events, starting points and end points

Log Summary

Total number of process instances: **27315**
Total number of events: **7306361**

Event Name

Event classes defined by Event Name

All events

Total number of classes: **3**

Class	Occurrences (absolute)	Occurrences (relative)
clicks_in_website	7174934	98,201%
workflow_message	66058	0,904%
phone_call	65369	0,895%

* We said before an average customer journey involves 267 actions => we can claim now that most of these actions are clicks, with ~ 2 or 3 messages and 2 or 3 phone calls (average)

Start events

Total number of classes: **3**

Class	Occurrences (absolute)	Occurrences (relative)
clicks_in_website	22184	81,215%
phone_call	4975	18,213%
workflow_message	156	0,571%

* 18 % of customer journeys start with a phone call...

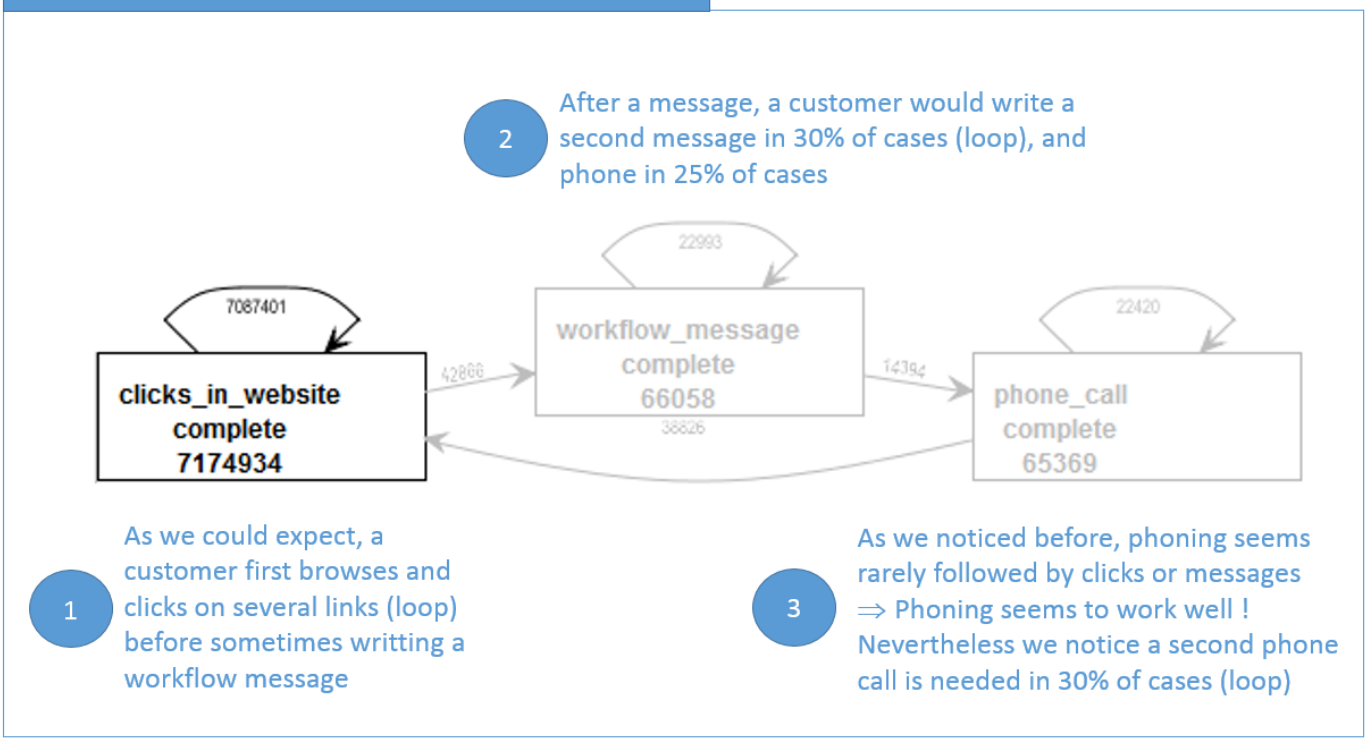
End events

Total number of classes: **3**

Class	Occurrences (absolute)	Occurrences (relative)
clicks_in_website	22233	81,395%
phone_call	3762	13,773%
workflow_message	1320	4,833%

... and it seems enough in 13 % of journeys (2 out of 3 times), so phone calling seems to work well !

Process discovered by ProM, 'Heuristic Miner' plugin

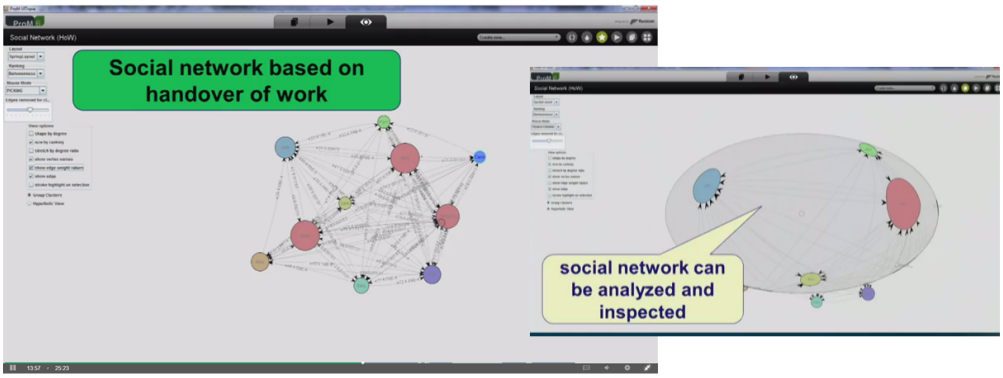


Summary of main conclusions:

In average a journey involves around **260 clicks**, **2 or 3 messages** and **2 or 3 phone calls**.
But when a customer **starts directy with a phone call**, things seem to be **solved quickly**.
(maybe a second phone call, but no further message or web-browsing).
A **typical customer journey** seems to be: **Web-Browsing** (clicks) -> **Message** -> **Phone call**.

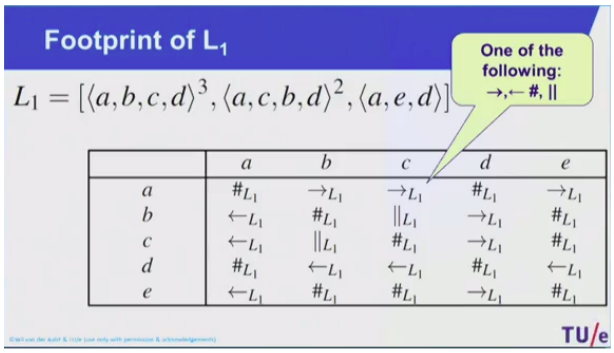
Next possible steps:

Add complaints data to the analysis: what is the specific journey of a customer ending with a complaint ? (or typology of users / or typology of service desk from organization).
Do the **same analysis for non logged users** (in this case logs will be joined based on IP addresses).
Achieve 'process cube' analysis: how the process vary according to the other variables ?
Add resources to the data integrated to ProM in order **to visualize** social interactions and **'handover of work'** within the organization.



Appendix: Alpha algorithm and Heuristic Miner

There are several algorithms able to discover a process model based on log analysis.
Alpha Algorithm was the first famous one: it consists in building an **'adjacency matrix'** from which it derives inferences of **causality** & **concurency** between activities. Here is an example of adjacency matrix, called 'footprint' of a specific 'trace L1' in this example:



	a	b	c	d	e
a	$\#_{L_1}$	$\rightarrow_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$
b	$\leftarrow_{L_1}$	$\#_{L_1}$	$\|\|_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$
c	$\leftarrow_{L_1}$	$\|\|_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$
d	$\#_{L_1}$	$\leftarrow_{L_1}$	$\leftarrow_{L_1}$	$\#_{L_1}$	$\leftarrow_{L_1}$
e	$\leftarrow_{L_1}$	$\#_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$

Basically, Alpha algorithm builds an adjacency matrix, then derives:
- causality if a -> b but never b -> a
- concurency if sometimes a -> b and sometimes b -> a

Some limitations of alpha algorithm:
- Decisions are made 'locally' so non local dependancies are not seen
- Noise is not 'filtered' (other algorithms handle this better)
- Loops of lenght 1 and 2 are not seen
- Other problems linked to representational bias of petri nets

As mentionned above, Alpha Algorithm has **many limitations**, most of which are **resolved by Heuristic Miner**.
The Heuristic miner (previously Little Thumb) derives XOR and AND connectors from dependency relations. It can **abstract** from exceptional behavior and **noise** (by leaving out edges) and, therefore, is also suitable for many real-life logs.

Sources and interesting sites about process mining:
http://processmining.org/ (http://processmining.org/)
https://fluxicon.com/blog/2010/10/prom-tips-mining-algorithm/ (https://fluxicon.com/blog/2010/10/prom-tips-mining-algorithm/)
https://www.coursera.org/learn/process-mining/home/welcome (https://www.coursera.org/learn/process-mining/home/welcome)