CJM-ab: Abstracting Customer Journey Maps Using Process Mining

Gaël Bernard[1(✉)] and Periklis Andritsos[2]

[1] Faculty of Business and Economics (HEC), University of Lausanne, Lausanne, Switzerland
gael.bernard@unil.ch

[2] Faculty of Information, University of Toronto, Toronto, Canada
periklis.andritsos@utoronto.ca

Abstract. Customer journey mapping (CJM) is a popular technique used to increase a company's understanding of their customers. In its simplest form, a CJM shows the main customer paths. When dealing with complex customers' trajectories, these paths are difficult to apprehend, losing the benefit of using a CJM. We present a javascript-based tool that can leverage process mining models, namely process trees, and business owners' knowledge to semi-automatically build a CJM at different levels of granularity. We applied our approach with a dataset describing a complex process, and shows that our technique can abstract it in a meaningful way. By doing so, we contribute by showing how process mining and CJM can be put closer together.

Keywords: Customer journey mapping · Process mining
Process tree · Customer journey analytics

1 Introduction

A customer journey map (CJM) is a conceptual tool used to visualize typical customers' trajectories when using a service. In their simplest form, CJMs show the interactions between a customer and a service provider through time. A series of interactions is called a journey. Because CJMs give a company a better understanding of their customers, they are becoming increasingly popular amongst practitioners. A CJM can be used as a design thinking tool by internal stakeholders to anticipate the best – or worst – journeys possible. Such journeys, displayed on a CJM, are called the *expected journeys*. However, customers might experience a different journey from the one anticipated. For this reason, few researchers [4,5,10] propose leveraging traces left by customers in information systems to build CJMs from evidence. Because the journeys that will be displayed on the CJM are produced from facts, we refer to them as the *actual journeys*. Such approaches are in line with the urgent call from the authors Lemon and Verhoef to take a data-driven approach to map the customer journey [15].

J. Mendling and H. Mouratidis (Eds.): CAiSE Forum 2018, LNBIP 317, pp. 49–56, 2018.
https://doi.org/10.1007/978-3-319-92901-9_5

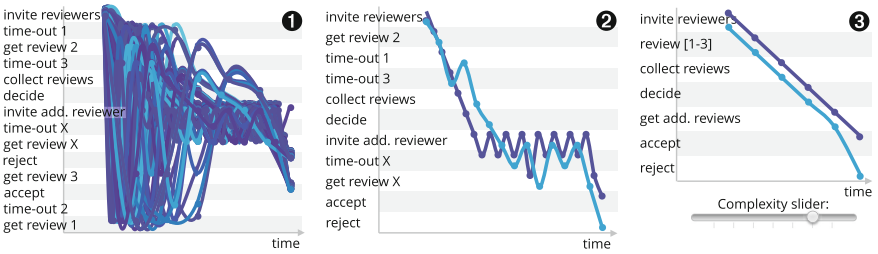

Fig. 1. Three possible ways to display the handling of reviews for a journal from [1] on a CJM: ❶ projecting the actual journeys (only the first 100 – out of 10,000 – journeys are displayed); ❷ using two representative journeys; and, ❸ using two representative journeys and abstracting the activities using the technique presented in this paper.

However, when dealing with numerous journeys, it becomes unrealistic to display all the actual journeys on a single CJM. For illustration purposes, Fig. 1 depicts 10,000 instances of the traces related to the handling of reviews for a journal, a synthetic dataset available in [1]. In the context of this dataset, *the service provider* is the conference's organizing committee, *the customers* are the researchers submitting their papers, and *a journey* describes the handling of the reviews, from the submission until the final decision. In Fig. 1, part ❶, it is difficult to apprehend the typical paths of the reviewing process. To this end, *representative journeys* have been introduced as a means of reducing the complexity. Indeed, the central CJM (❷) uses two representative journeys to summarize 10,000 actual journeys. Although representative journeys decrease the complexity by reducing the number of journeys, a CJM might still be difficult to apprehend when it is composed of many activities. Indeed, even though only representative journeys are used, quickly spotting the main differences between the two journeys visible in ❷ (Fig. 1) is not straightforward due to the high number of activities and the length of the journeys.

We propose CJM-ab (for CJM abstractor) a solution that leverages the expertise of process discovery algorithms from the process mining discipline to abstract CJMs. More precisely, we take as an input a process tree, we parse it, starting from the leaves, and iteratively ask the end-user if it is relevant to merge the activities that belong to the same control-flow, and, if so, to provide a name for this group of activities. By doing so, we let the end-user decide which activities should be merged and how they should be renamed. Then, one can visualize the same CJMs at different levels of granularity using a slider, which is visible in Fig. 1, part ❸. At a certain level of granularity, we clearly observe, given the end activities, that one representative journey summarizes the accepted papers, while the other one depicts the rejected papers. The importance and originality of CJM-ab is that it explores, for the first time, a seamless integration of business process models with customer journeys maps.

The paper is organized as follows. Section 2 introduces process mining and the process discovery activity. Section 2.2 describes the customer journey discovery activity. Section 3 describes our algorithm, and Sect. 4 provides a demonstration. Finally, Sect. 5 opens a discussion and concludes the paper.

2 Background

2.1 Process Mining and Process Discovery

Our approach is a seamless integration of Process Mining with Customer Journey Mapping and showcases the impact that the latter can have in the analysis of journeys. Process mining is an emerging discipline sitting between machine learning and data mining on the one hand, and process modeling and analysis on the other [2]. In this research, we focus on the discovery of process models, one of the three types of process mining along with conformance and enhancement.

The idea behind the discovery of process models is to leverage the evidence left in information systems to build process models from event logs. The resulting process models are, therefore, based on factual data, showing how the process was really executed. To build such a model, process mining uses an input data format called event logs. An event log is a collection of traces, a trace being a single execution of a process composed of one or multiple activities.

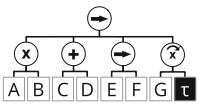

Fig. 2. One of the possible process trees given the event log $T = (\langle \text{BDCEF} \rangle, \langle \text{ACDEFG} \rangle, \langle \text{BCDEFGG} \rangle)$

For illustration purposes, let $T = (\langle \text{BDCEF} \rangle, \langle \text{ACDEFG} \rangle, \langle \text{BCDEFGG} \rangle)$ be an event log composed of 3 traces and 7 distinct activities. Regardless of the notation, the resulting models can express the control-flow relations between activities. For instance, for the event log, T, the model might express the following notation: (1) A and B are in an XOR relation ($\times$); i.e., only one of them is executed; (2) C and D are executed in parallel ($+$); i.e., both activities are executed in any order; (3) E and F are in a sequence relation ($\rightarrow$); i.e., F always follows E; (5) G is in a XOR loop (Combination of $\times$ and $\circlearrowleft$); i.e., it can be executed 0 or many times. Note that τ denotes a silent activity. It is used to correctly execute the process but it will not result in an activity which will be visible in the event logs. Figure 2 displays the five aforementioned relations using a process tree.

Discovering a process model from event logs is a challenge. Indeed, state-of-the-art algorithms should be robust enough to generalize (to avoid overfitting models) without being too generic. They should also try to build process models that are as simple as possible [3]. Many representations exist to express the discovered process models: Petri nets, YAWL, process trees, state machines, or bpmn models, to name a few. The next section introduces the notation used by our algorithm: process trees.

Process Tree. A process tree is an abstract hierarchical representation of a process model introduced by Vanhatalo et al. [17], where the leaves are annotated with activities and all the other nodes are annotated with operators such as $\times$ [14]. One interesting characteristic of process trees is that they guarantee the soundness of the models. A model is considered to be not sound when some activities cannot be executed or when the end of the process cannot be reached. The soundness guarantee is one reason that we choose the process tree notation. There are also three other reasons. First, process models in block structure

achieve best performance in terms of fitness, precision, and complexity [3]. Second, the hierarchical structure of process trees is ideal to derive multiple levels of granularity. Finally, according to Augusto et al. [3], process trees are used by top-performing process model algorithms, such as the inductive miner [11–13] or the Evolutionary Tree Miner [6].

2.2 Customer Journey Discovery

In [5], we proposed a process mining based model that allows us to map a standard event log from process mining (i.e., XES [9]) to store customer journeys, a first attempt to bring customer journeys and process mining closer together.

Discovering a set of representative journeys that best describe the actual journeys observed in the event logs is a challenge inspired by the process discovery challenge introduced in the previous section. However, instead of describing the control flows of activities using a business process model, the main trajectories (i.e., the representative journeys) are shown using a CJM. It encompasses three important challenges: (1) choosing the number of representatives. Let k be this number of representative journeys used on a CJM; (2) grouping actual journeys in k clusters; and (3) for each k, finding a representative journey. We briefly present these three challenges and ways to overcome them.

The first challenge is to set the number of representative journeys used to summarize the entire actual journeys. Looking at ❶ from Fig. 1, it is difficult to say how many representative journeys should be used to summarize the data. We identify two ways to solve this challenge. The number of representative journeys can be set manually, or it can also be set using standard model selection techniques such as the Bayesian Information Criterion (BIC) penalty [16], or the Calinski-Harabasz index [7].

Once k has been defined, actual journeys should be split in k clusters and a representative journey per cluster must be found. One of the ways, presented in [4], is to first define a distance function between actual journeys, such as the edit distance, or shingles, and to build a distance matrix; then, to split the actual journeys in k groups using hierarchical clustering techniques. Next, the representative can be found using a frequent sequence mining algorithm [4], by counting the density of sequences in the neighborhood of each candidate sequence [8], by taking the most frequent sequences [8], or by taking the medoid [8]. Instead of inferring the representative from the distance matrix, it is also possible to obtain it using statistical modeling [8]. We can employ an Expectation-Maximization algorithm on a mixture of k Markov models, and then for each Markov model the journey with the highest probability becomes the representative [10].

The next section describes a novel way to leverage business process models to abstract customer journey maps.

3 Abstracting Customer Trajectories Using Process Trees

CJM-ab uses four steps to render a CJM at different levels of abstraction. They are depicted in Fig. 3. This chapter introduces each step. In the first step, the

goal is to build a process tree given an event log. This can be done using the technique introduced in Sect. 2.1. Next, using the same event log, the goal is to build a CJM using the technique introduced in Sect. 2.2.

Fig. 3. Rendering a CJM at different levels of abstraction in four steps

The third step consists of parsing the tree obtained in step 1. To this aim, we developed a script in javascript which parses the process tree (i.e., XML file) and performs a reverse Breadth-first search; i.e., traversing the operators in the tree from the lowest ones to the root in a level-wise way. Let ℓ be the number of operators in the process tree. At each of the ℓ operators of the process tree, we offer the opportunity to the end-user to merge the leaves under the operator. If the user chooses to merge the activities, she should provide a new name and the operator virtually becomes a leaf. If the end-user chooses not to merge the activities, we keep the leaves intact. If the answer is no, we keep the activities separated at all levels of granularities, and we also disable the parents' steps. Indeed, we postulate that if a user does not want to merge two activities at a low level of granularity, it does not make sense to merge them later at a higher level of granularity.

Input : cjm, customer journey map
λ, level of abstraction
pt, process tree annotated with merging decisions
Output: cjm_λ, cjm at the level of abstraction λ

```
Function GetLevelAbstraction(cjm, λ, pt)
    for i ← 0 to λ do
        cjm → Abstract(cjm, pt.operator_i)
    return cjm
Function Abstract(cjm, op)
    foreach journey in cjm do
        journey.replace(op.leaves, op.new_name, removeSeqRepeats=True )
    return cjm
```

Algorithm 1. Function to get to the level of complexity λ

Finally, in step 4, we transform the CJM at different levels of abstraction. Let λ be the number of abstractions which will be available for a CJM. It can be

seen as the number of steps that will be included in the sliders visible in Fig. 1, part ❸. Note that λ is equal to the number of times the end-user decides to merge the activities and that $\lambda = \ell$ when the end-user merges all the activities. Let $operator_\lambda$ be the λ^{th} operator to be merged. Let GetLevelAbstraction(cjm, λ, pt) be a function that returns a CJM at the λ^{th} level of abstraction. Algorithm 1 shows how the function Abstract is recursively called to get to the level of abstraction λ. The parameter removeSeqRepeats in Algorithm 1 in line 7 emphasizes that continuous sequence of activities that are to be replaced, will be replaced by only one instance of the new name given for this operator. For instance, if the journey is "AABCBAC", the leaves that are to be replaced, are "A" and "B" and the new name is "X", the journey will become "XCXC". This reduces the length of the journeys and, thus, increases the abstraction. One can go back from more abstract to fine granular again by calling GetLevelAbstraction() again with a smaller λ. The next section illustrates these four steps with a running example.

4 Demonstration

This section provides a running example of our developed tool. The running example is based on synthetic event logs describing the handling of reviews for a journal (from [1]) cited in the introduction. It contains 10,000 journeys and 236,360 activities. This demonstration is available on http://customer-journey.unil.ch/cjm-ab/. In the first step, we obtained a process tree by using the inductive miner [14] with default parameters[1]. It results in the process tree visible in Fig. 4. In the second step, we obtain a CJM by: (1) measuring the distance between actual journeys using the edit distance; (2) building a dendrogram using a hierarchical clustering algorithm; (3) finding k using the Calinski-Harabaz Score ($k = 2$); (4) finding representative journeys using the function 'seqrep' available in Traminer, a R package[2]. It results in a CJM which is visible in

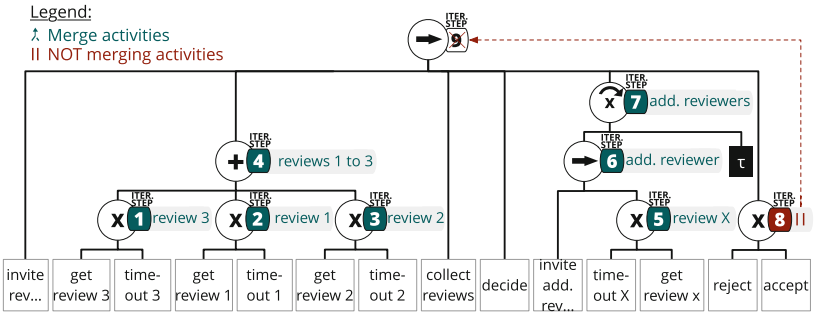

Fig. 4. Process tree annotated with the order in which the operators are parsed (i.e., 'iter. step') and the decisions to merge the activities or not (i.e., colors red and green).

[1] Using the software ProM available at http://www.promtools.org/doku.php.
[2] Available at: http://traminer.unige.ch/doc/seqrep.html.

❷ (Fig. 1). In the third step, we parse the XML in javascript. To traverse the tree, we are using a tree-like data structures [3]. The order in which the operators are parsed is depicted in Fig. 4 (i.e., 'step'). Figure 4 shows that we decided to merge 7 out of the 9 operators (in green in Fig. 4). Note that we decided not to merge the activities 'reject' and 'accept', which disabled the option of merging all the activities below step 9. The Fig. 5 shows a screen capture of the application when merging the activities during step 1. Finally, the Fig. 6 shows the resulting CJMs at three levels of abstraction.

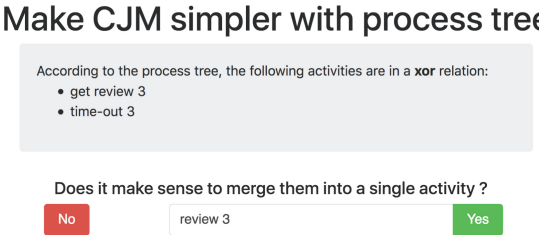

Fig. 5. Screen shot of the application during the merging process at 'step 1'

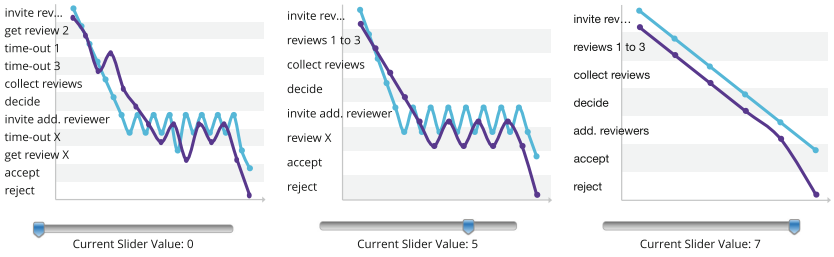

Fig. 6. Results at the levels of abstraction 1, 3, and 7.

5 Conclusion

CJMs are being used more and more to help service providers put themselves in their customers' shoes. However, very little research has investigated automated ways of building them. We contribute by showing how a process mining model can be used to guide the abstraction of a CJM. By answering few questions about the merging of the activities and by playing with the abstraction sliders, we anticipate that our tool allows practitioners to gain new insights about their data. By leveraging process trees – a format built within the process mining community – we can bring customer journey analytics and process mining closer together. We expect that many algorithms and works from process mining are relevant for the discovery of customer journeys.

[3] Available at: https://github.com/joaonuno/tree-model-js.

References

1. van der Aalst, W.: Synthetic event logs - review example large.xes.gz (2010). https://doi.org/10.4121/uuid:da6aafef-5a86-4769-acf3-04e8ae5ab4fe
2. van der Aalst, W.: Process Mining: Data Science in Action. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49851-4
3. Augusto, A., Conforti, R., Dumas, M., La Rosa, M., Maggi, F.M., Marrella, A., Mecella, M., Soo, A.: Automated discovery of process models from event logs: Review and benchmark. arXiv preprint arXiv:1705.02288 (2017)
4. Bernard, G., Andritsos, P.: CJM-ex: goal-oriented exploration of customer journey maps using event logs and data analytics. In: 15th International Conference on Business Process Management (BPM 2017) (2017)
5. Bernard, G., Andritsos, P.: A process mining based model for customer journey mapping. In: Proceedings of the Forum and Doctoral Consortium Papers Presented at the 29th International Conference on Advanced Information Systems Engineering (CAiSE 2017) (2017)
6. Buijs, J.C., van Dongen, B.F., van der Aalst, W.M.: Quality dimensions in process discovery: the importance of fitness, precision, generalization and simplicity. Int. J. Coop. Inf. Syst. **23**(01), 1440001 (2014)
7. Caliński, T., Harabasz, J.: A dendrite method for cluster analysis. Commun. Stat.-Theory Methods **3**(1), 1–27 (1974)
8. Gabadinho, A., Ritschard, G., Studer, M., Müller, N.S.: Extracting and rendering representative sequences. In: Fred, A., Dietz, J.L.G., Liu, K., Filipe, J. (eds.) IC3K 2009. CCIS, vol. 128, pp. 94–106. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19032-2_7
9. Günther, C.W., Verbeek, E.: XES-standard definition (2014)
10. Harbich, M., Bernard, G., Berkes, P., Garbinato, B., Andritsos, P.: Discovering customer journey maps using a mixture of Markov models, December 2017
11. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering block-structured process models from event logs - a constructive approach. In: Colom, J.-M., Desel, J. (eds.) PETRI NETS 2013. LNCS, vol. 7927, pp. 311–329. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38697-8_17
12. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering block-structured process models from event logs containing infrequent behaviour. In: Lohmann, N., Song, M., Wohed, P. (eds.) BPM 2013. LNBIP, vol. 171, pp. 66–78. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-06257-0_6
13. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering block-structured process models from incomplete event logs. In: Ciardo, G., Kindler, E. (eds.) PETRI NETS 2014. LNCS, vol. 8489, pp. 91–110. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07734-5_6
14. Leemans, S.: Robust process mining with guarantees. Ph.D. thesis, Eindhoven University of Technology (2017)
15. Lemon, K.N., Verhoef, P.C.: Understanding customer experience throughout the customer journey. J. Mark. **80**(6), 69–96 (2016)
16. Schwarz, G., et al.: Estimating the dimension of a model. Ann. Stat. **6**(2), 461–464 (1978)
17. Vanhatalo, J., Völzer, H., Koehler, J.: The refined process structure tree. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 100–115. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85758-7_10