Discovering Requirements through Goal-Driven Process Mining

Jacek Dąbrowski*†, Fitsum Meshesha Kifetew*, Denisse Muñante*, Emmanuel Letier†,
Alberto Siena‡ and Angelo Susi*
*Fondazione Bruno Kessler, Trento, Italy
{dabrowski, kifetew, munante, susi}@fbk.eu
†University College London, London, UK
{j.dabrowski, e.letier}@cs.ucl.ac.uk
‡Delta Informatica SpA, Trento, Italy
alberto.siena@deltainformatica.eu

***Abstract*—Software systems are designed to support their users in performing tasks that are parts of more general processes. Unfortunately, software designers often make invalid assumptions about the users' processes and therefore about the requirements to support such processes. Eliciting and validating such assumptions through manual means (e.g., through observations, interviews, and workshops) is expensive, time-consuming, and may fail to identify the users' real processes. Using process mining may reduce these problems by automating the monitoring and discovery of the actual processes followed by a crowd of users. The Crowd provides an opportunity to involve diverse groups of users to interact with a system and conduct their intended processes. This implicit feedback in the form of discovered processes can then be used to modify the existing system's functionalities and ensure whether or not a software product is used as initially designed. In addition, the analysis of user-system interactions may reveal lacking functionalities and quality issues. These ideas are illustrated on the GreenSoft personal energy management system.**

***Index Terms*—process mining; process validation; crowdsourcing; requirements engineering; goal orientation**

I. Introduction

Software systems are designed to support users in performing tasks, which are parts of more general processes, that lead to the accomplishment of different users' goals [1], [2]. Often, in case of large software systems, such as governmental or e-shopping platforms, the number of users is large and diverse, so their goals can be heterogeneous. While such systems could also be seen in light of sociotechnical systems that comprise humans in continuous interaction with a software system to achieve a collective goal [3], in this paper we refer to software systems which interact directly with a user and the process is contained within the software system (application). We defer the treatment in the context of sociotechnical systems to future work.

To deal with the complexity that arises with a large number of users with heterogeneous goals, systems generally support multiple execution paths, and let the users decide which path to actually follow, thus enabling them to define their own (preferred) processes. As a consequence, the number and complexity of processes supported by such systems tends to be large as well, and often not all the user processes are identified *a-priori* by the requirements analysts. This might lead to, possibly serious, misalignments between the designed interaction processes and the processes that the users actually follow [4]. For example, users may end up in a situation in which, to achieve their objectives, they go through over-complicated or unintuitive processes. This happens mainly because it is often impossible to successfully accomplish the requirements of all the involved stakeholders and, even if the required stakeholders are engaged in the design, it may not be possible to acquire all their needs [4], [5]. For this reason, requirements analysts should carefully consider not only the objectives of the user but also the actual behaviors and processes that the users of an application follow in their interaction with the system to pursue such objectives.

Several approaches were proposed to tackle this problem. For example, task analysis [6] aimed at addressing the dilemma. This method aimed at gaining insights into what tasks are required by users to accomplish a given process. The method provides valuable information about cognitive processes while performing a given task. Consequently, the designer can obtain information about what processes users wish to perform and refer against values provided by the developed functionalities supporting existing interaction.

Process mining [7], [8] refers to a set of methods that allow to analyze the event data, resulting from the execution of a system, and discover the underlying processes to form process models. This is done with the purpose of revealing, for example, typical patterns of behavior in the participants, possible process bottlenecks, misalignments or variants with respect to an intended nominal process. So, process mining answers questions about *what has been done during the processes?* and allows to confront it with the question *what should have been done?* This is particularly useful for a software engineer [4] that has limited possibilities to detect discrepancies between actual processes and intended processes, allowing to conduct conformance checking and indicating the points in the process that do not comply with the high-level user goals [9].

Nowadays, software applications produce a large amount of data related to their usage and performances. Such data can be extracted and analyzed using process mining, hence highlight-

ing an *implicit feedback* related to the interaction behavior of end-users with the software. Such implicit feedback obtained from monitored user interactions can be a relevant source to identify new requirements for the software [10].

In this vision paper, we introduce a new approach that combines process mining and crowdsourcing methods to identify and validate software process requirements based on the crowd's knowledge [11], [12]. To this end, our approach relies on the possibility of exploiting the perception of the crowd with respect to the software, and hence the supported processes. In particular, we envisage two types of users: the *end-users* and the *crowd*. Specifically, the processes of the *end-users* are analyzed to detect possible issues within the processes and to design the objectives and the activity to be performed by the *crowd*. On the other side, the *crowd* is probed to confirm those issues identified from the *end-users* and to explore behaviors that are rarely performed by the latter group.

The novelty of the approach lies in its ability to extend Requirements Engineering (RE) techniques by revealing new requirements exploiting process mining techniques together with crowdsourcing. Existing works in the literature that make use of crowdsourcing focused mainly on extracting explicit user feedback [10], [11]. We argue that implicit feedback could lead to capturing misunderstanding of users needs, and thus identifying ignored requirements. Implicit feedback provides requirement engineers with information that cannot be easily expresses by users such as their cognition or perception about a software system. The synergy between process mining technique and the crowd could prove an important source of new requirements. Process mining techniques enable us to extract implicit feedback in the shape of processes conducted by users. However, to provide statistical significance and rationale for obtained outcomes, a large number of heterogeneous interactions with the system must be executed, hence motivating the use of crowdsourcing.

In contrast to human-computer interaction techniques focused on analyzing user behavior to improve user experience (UX) or usability [13], we consider another perspective, i.e., users' processes, and we aim at analyzing those processes: i) to detect mismatch between designed and developed processes, ii) to validate domain assumptions, and iii) to elicit new requirements and refine existing ones. Those requirements are not only limited to UX or usability ones, but they may involve other requirements such as security or performance as well.

The remainder of the paper is structured as follows. In Section II a motivating example is presented. Section III presents an overview of the envisaged approach. Sections IV and V highlight some challenges and envisage some new research lines.

II. Motivating example

This section presents a toy example, *i.e.,* GreenSoft case study, to illustrate the motivation behind the envisaged approach.

Background. GreenSoft is a Small-Medium sized Enterprise (SME) developing a software product allowing prospective customers to monitor and analyze their household's energy consumption [14]. The software is available to customers in the form of a mobile version for portable devices as well as a web application for desktop users. To reduce the cost of maintaining two versions of the software product, the solution architect decided to customise only the presentation layer according to the client-side. The business logic and data layer are maintained at the back-end side, common to both variants, so that if any change in application logic are required, then modifications at the back-end will be automatically reflected on both the mobile and web applications.

The main functionalities of the application support processes such as: Analyzing energy consumption, Controlling different energy elements, Determining save energy plan.

Business problems. Let us imagine a scenario in which the company received feedback from its users that highlights problems experienced by the users while interacting with the software application. For instance, Problem A: *the interaction with the software is unintuitive and accomplishing intended users' goals is obstructed by tedious and awkward tasks*, Problem B: *the application is not easily learnable*, or Problem C: *carrying out fundamental activities is lengthy and time-consuming*.

Consequently, the requirement engineers supposed that the *Problem A* may be driven by software not aligned with real users' processes, in which case the processes would require redesigning existing user processes, which consequently would impose new requirements.

A hypothesis behind *Problem B* was that maybe some group of users are offered functionalities they do not really need or are not familiar with, but they are brought out in the software application and cover other functionalities that are more important from the users' perspective. On the other hand, designers assumed that navigating the application by new users and accomplishing their goals should not be time-consuming, provided that the application delivers high usability.

Finally, for the *Problem C*, the conclusion of the engineers was twofold: either there are ill-prepared processes forcing users to spend excessive amounts of time to accomplish their goals or performance problems occurred because some unrealistic assumptions were made during the design phase.

To confirm the hypotheses and address the identified business problems, the requirement engineers determined the following objectives:

Objective 1: Identify real user goals and supporting processes when using the applications and confront outcomes against the design assumptions. Mismatch between design assumptions and actual users-system interaction could prove the presence of *Problem A*.

Objective 2: Identify alternatives of users' processes leading to the same goals as well as deadlock paths. If such phenomena occur, then it could be an indicator for *Problem A* and *Problem B*.

Objective 3: Identify the time users spend to perform a given process or a particular task. If one activity took an

excessive amount of time, then it may be a performance issue confirming the problem *Problem C*.

Technical problems. To accomplish the aforementioned objectives, real user processes have to be identified and analyzed. Though GreenSoft may involve selected users and conduct contextual task analysis, the method may be biased and limited by factors such as a selective choice of participants or their limited numbers. Therefore, the company intends to involve the Crowd, overcoming those limitations and providing a heterogeneous group of users. However, an emerging technical problem, and thus research question is:

RQ: How to Discover Requirements through Goal-Driven Process Mining?

III. Proposed Approach

To achieve the business objectives and overcome technical problems, we propose an approach leveraging the synergy between Requirements Engineering, Crowdsourcing and Process Mining. The GreenSoft case study is used to explain a sketch of the proposed approach, modus operandi of the approach, and a possible scenario using the proposed approach.

A. A sketch of the proposed approach

A sketch of the proposed approach (*addressing RQ*) is illustrated in the Figure 1. The proposed approach is composed using three main components: `Crowd`, `Process Mining` and `Planner`. The other components, i.e. `GreenSoft application` and `End-users` are ingredients required for the proposed approach, though they do not constitute its integral parts.

`End-users` depicts users conducting their daily processes using the `GreenSoft application`, whereas `Crowd` denotes a large and heterogeneous group of users delegated to carry out some processes using `GreenSoft application` to accomplish specified objectives [11]. Though both `Crowd` and `End-users` could be perceived as actors, we term them as *components* for the sake of simplicity. The users-system interaction is marked by a dashed arrows, while the stored information about the conducted processes in the form of logs are illustrated as gray boxes.

`Process Mining` depicts a component responsible for extracting users' processes from system logs. Firstly, it filters the obtained processes from noise that could be a result of randomly executed processes. Then, it synthesizes and generalizes the information to produce models of users' processes. Those models are next exploited for further analysis that could lead to identification or confirmation of new users' goals and underlying requirements as well as misalignment between designed and executed processes. The analysis outcome, i.e., artifacts are presented using blue and red boxes in Figure 1. The blue boxes depict potential new users' goals and underlying requirements as well as misalignments between designed and executed processes, whereas the red boxes illustrate confirmation for new goals, requirements and processes misalignments. In addition, the latter artifact constitutes the input information for `Planner`.

Finally, `Planner` is a software component aiming at defining objectives and tasks, illustrated by the black box, to be delegated to `Crowd`.

The main objective of the proposed approach is to examine whether domain assumptions about user processes (*D*) hold as well as design specification (*S*) is satisfied by the software [1]:

$$D, S \vDash G \tag{1}$$

According to *Equation 1*, if users conduct their processes via the system according to the assumed processes D and if the software satisfies its specification S, then the system's goals G are satisfied.

Therefore, the approach aims at identifying real users' goals and supporting processes leading to the accomplishment of these goals. Consequently, the outcome could provide justification for design specifications and correctly presumed domain assumption or result in further requirements elicitation.

B. The modus operandi based on GreenSoft

The modus operandi of the proposed approach can be divided into two phases: *problem identification* and *problem confirmation*. The first one aims at identifying misalignment between designed processes (S) and actual ones (D) leading to adopting hypotheses about new requirements. Whereas the other is supposed to confirm or turn down presumed hypotheses.

We assume that *problem identification* phase is composed of the following three steps. Firstly, `End-users` conduct their daily processes (D) using `GreenSoft application` for a given time period. Next, the information about these processes (D), i.e. logs is proceeded to the input of `Process mining`, and then the component produces artifacts containing a list of possible new requirements and misalignment between designed (S) and actual processes (D) executed by `End-users`. Finally, these artifacts are forwarded to `Planner` and *problem confirmation* phase is set up.

Once `Planner` received the artifacts, it orchestrates objectives and tasks to be accomplished by `Crowd`. Consequently, `Crowd` conducts their processes (D) by `GreenSoft application` in accordance with the guidelines from `Planner`. Their interaction with the software is analogously stored as logs and subsequently analyzed by `Process mining` component. Providing that misalignment between designed processes (S) and actual ones (D) are confirmed, then it may lead to the implementation of new requirements and `GreenSoft application` re-designing.

C. Possible scenario using the proposed approach

One could imagine that GreenSoft decided to achieve the business objectives **O1-O3**, and addressed the technical problem **RQ** using our approach, as illustrated in Figure 1. Thus, the GreenSoft requirement engineers could design an experiments including `Crowd`; One motivation behind involving `Crowd` could be to provide statistical significance for revealing new requirements and misalignment between designed and actual processes. Such significance could not

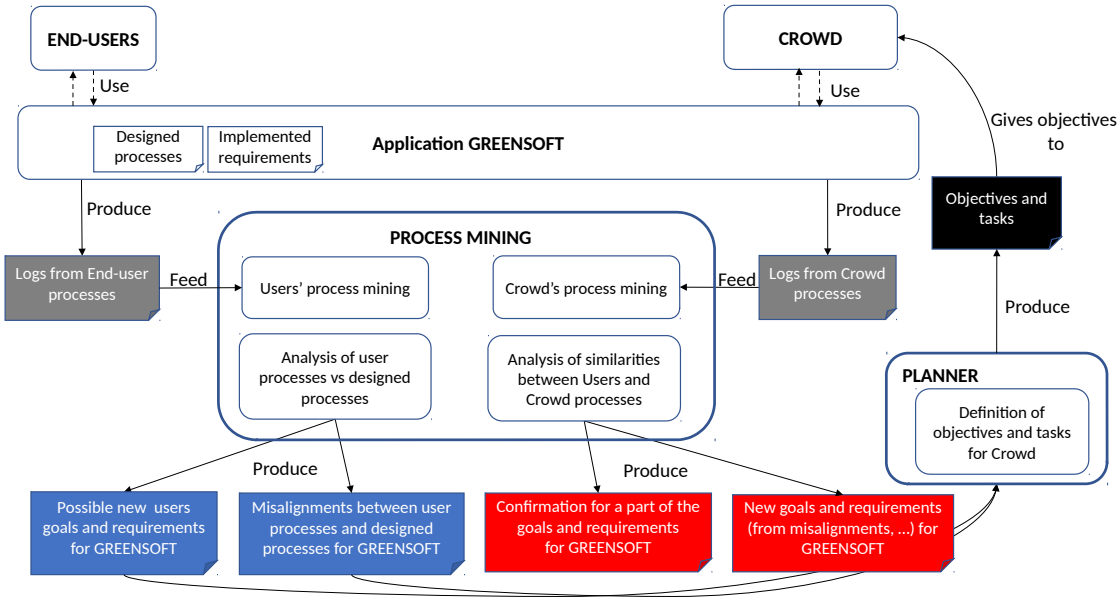

Fig. 1: A sketch of the proposed approach.

be obtained by involving limited numbers of `End-users`. Second motivation could be that the GreenSoft has a tight schedule and aims at revealing possibly the largest number of new requirements in a limited period. Therefore, even dozens of `End-users` could not execute as many processes as `Crowd`.

Thus, an experiment could firstly begin by monitoring daily processes of `End-users` (e.g., *Analyzing energy consumption, Determining save energy plan* and *Controlling different energy elements*). Consequently, one could imagine results leading to surprising conclusions. For instance, one process (e.g., *Determining save energy plan*) was hardly used by mobile users, whereas the process was at the high usage by desktop users.

A requirement engineer could conclude that busy users (with limited time) often accessed the mobile application to perform the most significant processes from their perspective (e.g., *Controlling different energy elements*). Therefore, other functionalities on the screen could have blurred (make it difficult to reach) the ones that are the most desired. Another conclusion could be that some user goals (e.g., *Displayed payment for the current month*) should be achieved by clicking a single button on main menu rather than executing a complex process (e.g., *Analyzing energy consumption*) since users were mainly interested in this information and not other energy consumption details.

Then the experiment could involve `Crowd` to carry out processes towards achieving delegated objectives. The aims of the study could be to recognize the misalignments between actual processes and designed ones as well as to confirm *candidate requirements* aiming at improving existing processes. Thus, one could imagine that the `Crowd` were given a goal (e.g., *Getting to know about appliances consuming the most energy*) without specifying a concrete process to achieve the goal.

Theoretically, the goal could have been formulated by `Planer`, as a results of analyzed processes that could have been earlier executed by `End-users`. Also in this case, one could imagine a surprising outcome: `Crowd` executed varied processes to accomplish the same specified goals (e.g., *Getting to know about appliances consuming the most energy*). Instead of following a designed process providing the information in a theoretically straightforward manner (e.g., *Determining save energy plan*), some members of `Crowd` could have carried out another process (e.g., *Analyzing energy consumption*) and made several manual steps to achieve their goal (e.g., *Getting to know about appliances consuming the most energy*).

IV. Challenges

The methods and techniques discussed in the previous sections are perfect candidates for the concrete implementation of the envisaged approach. However, several research challenges are still to be addressed.

A first aspect is related to the definition of the set of concepts characterizing the part of the approach that, starting from the analysis of the user interaction processes, allows to detect candidate requirements and misalignment between designed processes and real user interactions.

A second aspect concerns the definition of principles and guidelines to design the objectives to be accomplished by the members of crowd and connect them to the objectives of discovering new requirements, confirming requirements that have been already identified (e.g., thanks to the application users), discovering processes that are rarely exercised by the

application users or not in line with the processes designed by the developers of the application.

A final challenge pertains to the capacity of extracting useful information from crowd in order to support the confirmation of the findings from the users behavior analysis and/or the existence of new interaction behaviors not exercised from the users of the application.

V. Future directions

To address the challenges we envisage several research lines. Goal oriented modeling and reasoning is a candidate approach to design the objectives related to the process mining and analysis in order to direct the identification of relevant information about new requirements and process misalignments. Similar methods, coupled with statistical or machine learning methods, could also be exploited to design methods to ultimately maximize the capacity of the crowd to support the discovery of new requirements. Finally, research on the orchestration of the different activities in the whole process is another aspect to be considered.

The other aspect we would like to stress here is that the implicit feedback from mining the crowd's processes could be exploited as a complementary source of information to *explicit feedback*. Explicit feedback, e.g., in the form of textual feedback, has been recently proposed to refine properties of previously elicited requirements and help improve requirements prioritization [15]. A promising research direction could involve the use of implicit feedback in the form of user processes, in conjunction with explicit feedback, to further enhance requirements prioritization activities.

Acknowledgement

This work is a result of the SUPERSEDE project, funded by the H2020 EU Framework Programme under agreement number 644018.

References

[1] A. Van Lamsweerde, "Goal-oriented requirements engineering: A guided tour," in *Proceedings of the Fifth IEEE International Symposium on Requirements Engineering*, RE '01, (Washington, DC, USA), pp. 249–, IEEE Computer Society, 2001.

[2] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos, "Tropos: An agent-oriented software development methodology," *Autonomous Agents and Multi-Agent Systems*, vol. 8, pp. 203–236, May 2004.

[3] A. K. Chopra, F. Dalpiaz, F. B. Aydemir, P. Giorgini, J. Mylopoulos, and M. P. Singh, "Protos: Foundations for engineering innovative sociotechnical systems," in *IEEE 22nd International Requirements Engineering Conference, RE 2014, Karlskrona, Sweden, August 25-29, 2014*, pp. 53–62, 2014.

[4] V. A. Rubin, A. A. Mitsyuk, I. A. Lomazova, and W. M. P. van der Aalst, "Process mining can be applied to software too!," in *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM '14, (New York, NY, USA), pp. 57:1–57:8, ACM, 2014.

[5] "Usability first operated by foraker labs of boulder, colorado." http://www.usabilityfirst.com/about-usability/requirements-specification/, 2017.

[6] L. Teixeira, C. Ferreira, and B. S. Santos, "Using task analysis to improve the requirements elicitation in health information system," *29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 2007.

[7] V. Rubin, I. Lomazova, and W. M. P. v. d. Aalst, "Agile development with software process mining," in *Proceedings of the 2014 International Conference on Software and System Process*, ICSSP 2014, (New York, NY, USA), pp. 70–74, ACM, 2014.

[8] V. A. Rubin, C. W. Günther, W. M. P. van der Aalst, E. Kindler, B. F. van Dongen, and W. Schäfer, "Process mining framework for software processes," in *ICSP*, vol. 4470 of *Lecture Notes in Computer Science*, pp. 169–181, Springer, 2007.

[9] J. Dąbrowski, "Towards an adaptive framework for goal-oriented strategic decision-making," *25th IEEE International Requirements Engineering Conference, RE 2017, Portugal, Lisbon, September 4-8, 2017*, 2017 (to appear).

[10] W. Maalej, M. Nayebi, T. Johann, and G. Ruhe, "Toward data-driven requirements engineering," *IEEE Software*, vol. 33, no. 1, pp. 48–54, 2016.

[11] E. C. Groen, N. Seyff, R. Ali, F. Dalpiaz, J. Doerr, E. Guzman, M. Hosseini, J. Marco, M. Oriol, A. Perini, and M. Stade, "The crowd in requirements engineering: The landscape and challenges," *IEEE Softw.*, vol. 34, pp. 44–52, Mar. 2017.

[12] R. Snijders, F. Dalpiaz, M. Hosseini, A. Shahri, and R. Ali, "Crowd-centric requirements engineering," in *Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, UCC '14, (Washington, DC, USA), pp. 614–615, IEEE Computer Society, 2014.

[13] "Google user experience research." https://www.google.com/usability/faq/, July 2017.

[14] "The second international workshop on crowd-based requirements engineering." https://crowdre.github.io/ws-2017/submissions.html/, 2017.

[15] I. Morales-Ramirez, D. Muñante, F. M. Kifetew, A. Perini, A. Susi, and A. Siena, "Exploiting user feedback in tool-supported multi-criteria requirements prioritization," in *25th IEEE International Requirements Engineering Conference, RE 2017, Lisbon, Portugal, September 4-8*, 2017 (to appear).