UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Supporting decentralized collaborative processes in the digital transformation

David Sánchez Charles

Supporting Decentralized Collaborative Processes in the Digital Transformation

David Sánchez Charles

Advisor: Josep Carmona, Universitat Politècnica de Catalunya

Co-advisor: Victor Muntés-Mulero, CA Technologies

Abstract

Crowdsourcing, the art of involving several individuals in the decentralized execution of business activities, is being positioned as the replacement of outsourcing, as it allows organization to reach a capable workforce whenever it is necessary for the business. Nevertheless, adoption among industry is still low, as the technology is yet not mature and, in particular, it is difficult to monitor the execution of the business activities in a crowdsourcing platform. In this thesis, we advance towards creating better monitoring tools for crowdsourcing processes and a mechanism for modelling the worker's behavior.

Formalizing the work to be done in a process is the first step for improving the overall efficiency and quality of problem resolution. Still, there is a lack of mechanisms for defining business processes capable of adapting to the needs of the crowd. Therefore, we start this thesis by introducing a graphical modelling language for describing decentralized collaborative processes. The focus of this work is to allow the definition of complex worker requirements, as well as provide a quick overview and assessment of the implemented quality assurance mechanisms. In a longer-term vision, having well-defined processes will help in making more predictable the performance of any crowdsourcing project.

For those cases in which there is no formal process and the crowd can self-organize how they execute the business activities, we have also made the first steps for designing a method capable of discovering processes by analyzing the factual work done in the platform. Assuming that all steps recorded by

the platform have some textual description of the work done, we propose to use novel natural language processing tools for generating groups of similar activities and, hence, enabling later analytics and insights, such as a process discovery for understanding, monitoring, or simply formalize the underlying crowd-process.

As for modelling the worker's behavior, we started by studying a particular crowdsourced process pattern that enables the platform to rank users based on their performance. The novelty of such prototype relies on the role of the reviewer, played by skilled individuals on the platform, that acts as reviewers of the translations done by in-training translators. The feedback provided by the reviewers is later reused for deciding if an in-training translator should be promoted to the reviewer role.

Unfortunately, there is no clear way of extrapolating the previous user evaluation to other processes. In this thesis, we propose to let the platform monitor the actions performed by individuals in order to create a profile of their behavior. We assume that those actions can be thougth as events that can be later processed by a discovery method, summarizing such actions in the form of a process model. Apart from the fitness of the resulting process models, precision is a key quality metric of these behavioral profiles. Low-precision models are more likely to describe the behavior of several users, reducing the insights obtained by analyzing or comparing process models. In particular, repetition of activities – very often due to the human nature – is one of the key trace characteristic that reduces precision of models discovered with most process mining techniques as we highlight, and palliate, during this chapter.

We also propose a new similarity metric between process models, enabling platforms to compare users based on the similarity of the user profiles. In particular, we have applied this similarity metric with an industrial dataset compromising several workers with access to a source code repository, and it turns out that their role in the organization is partially seen in how they access such source code repository.

Contents

CHAPTER 1 Introduction

Jeff Howe coined the term crowdsourcing as the *act of taking a job traditionally performed by a designated agent (usually an employee) and outsourcing it to an undefined, generally large group of people in the form of an open call* [45]. His article reviewed the successful reborn of the *Wisdom of Crowds* philosophy in the new digital era. The main idea behind the wisdom of crowds is that collaboration between several individuals produce better results than if individually performed, assuming some conditions on the diversity and independence of the group. By expanding this collaborative method for solving problems to the Internet, such conditions are almost trivially satisfied by considering the large population with access to the Internet.

In crowdsourcing platforms such as Amazon Mechanical Turk[1], complex problems can be solved through the use of unprecedented mechanisms to allow for the collaboration of thousands of remote Internet users, leveraging the potential of emerging intelligence from the crowd. Nowadays, millions of people are asynchronously analysing, synthesising, providing opinion and labelling and transcribing data that can be automatically mined, indexed

[1]`http://www.mturk.com`

and even learned. Human brain-guided computation is able to perform tasks that computers can hardly do, at overwhelming speeds.

The translation industry was one of the early-adopters of crowdsourcing. Hiring translators of minority languages is usually a long and costly process, whilst it is expected to easily find native speakers on a large community via the Internet. Besides, most translation companies suffer from a highly-variable workload that enforces them to hire freelancers in order to meet deadlines, and hence the elasticity of crowdsourcing is also a key point to translation companies. Moreover, current translators are used to sign contracts per project, or collaboration, that is the foundation of task resolution in Crowdsourcing.

The generalisation of on-line social networks, the increase in the unemployment rates in many countries because of the economic recession and the increasing need of industry to flexibly involve human beings in big data analytics and processing are just three important motivations for the growth of the number of platforms and solutions using crowdsourcing strategies. Besides, the current economic recession affects millions of families worldwide. Just as an example, the unemployment rate in Spain is 19.6, and 43.9 among citizens younger than 25 years old (July 2016)[2]. With this situation, unemployment is not only restricted to workers with low levels of education, but it also affects highly qualified professionals and, specially, those that are trying to find their place in the labour market. Apparently, this situation is just marking the beginning of a long depression period and it may be a

[2]"Seasonally adjusted unemployment". Eurostat. *Unemployment news release, 163/2016, 31 August 2016.*

catalyst for crowdsourcing to consolidate as a common new mechanism for outsourcing.

1.1 Problem Description

Jeff Howe narrowed the application of crowdsourcing methodologies to an industrial problem resolution with Internet as a medium for finding the right human intervention. During the last decade, it has already been proven that crowdsourcing is an efficient replacement for outsourcing, not only in industrial scenarios [33] but also as a useful tool for academia [13]. Nevertheless, current trends seem to indicate that crowdsourcing methodologies and technologies are going to change to satisfying new paradigms.

Researchers have proposed [36, 68, 91] processes in which computers *ask for help* to human reviewers (possibly in a crowdsourcing platform) in case of obtaining a classification, or prediction, with low confidence. These examples of collaboration between a flexible pool of workers and computers are not isolated cases, as a recent market research performed by Gartner[3] predicts that more than 6 billion connected devices will be requesting support to humans by 2018. Collaboration between those devices and humans may be automatized and monitored by formal processes, which describes and coordinates the activities performed by the individuals and computers. In this scenario, a crowdsourcing platform would act as the enabler for individuals and computers to collaborate, following the guidelines provided by formal

[3] *Top Strategic Predictions for 2016 and Beyond.* `https://www.gartner.com/doc/3142020/top-strategic-predictions-future-digital`

rules. One can imagine the possibilities for allowing Third Parties to answer those support petitions, and letting the platform decide which is the best candidate based on performance and cost.

We have also seen that the industry is moving towards Software-as-a-Service solutions, in which workers connect to a website or application in order to collaborate with their peers, and customers, in their daily work. In such scenario, the only purpose of the workplace is to socialize with your team members. Under such conditions, employees are starting to demand to their employers to work remotely from home. Gartner again predicts a shift on this direction for the near future, stating that more than 3 million workers will be supervised by a *roboboss* by 2018. Performance of employees will not be measured only by direct supervision of their manager, partially possible due to workplace collocation, but by a machine that constantly measures performance and behavior of employees. Thanks to the *roboboss*, managers will be able to focus on human aspects and career development of their direct reports. Besides, having an understanding of the user profile will help in the distribution of tasks to the best suitable workers.

Based on the growing education level of unemployed individuals and the two aforementioned predictions on how humans will collaborate in a decentralized manner with the involvement of intelligent machines, we believe that there will be a shift on how crowdsourcing platforms will be considered by the industry as a real industrial complement or replacement to outsourcing. In order to support this paradigm-shift on how people and devices collaborate, a crowdsourcing platform must

- broaden its scope to decentralized collaboration, reducing the relevance of microtask-oriented marketplace crowdsourcing projects in which workers individually perform low-effort tasks,

- provide a mechanism for describing and monitoring collaborative processes, including quality assurance of the outcomes of the process,
- and provide a worker management tool capable of profiling and measuring performance of all the involved actors.

Nevertheless, as we review in the following two sections, state-of-the-art crowdsourcing solutions are not yet capable of tackling such collaborative, process-oriented scenario.

1.1.1 Formal Processes and Quality Monitoring in Crowdsourcing

Although crowdsourcing has been proven as an efficient mechanism for replacing traditional outsourcing, there are two factors that hinders the penetration of this model into industry: lack of best practices and output quality.

Research on Crowdsourcing assumes that companies willing to adopt this technology are able to design and tune their tasks. Results are yet sparse and its application are narrowed to specific examples [6]. For instance, Threadless[4], a successful online T-shirt store, outsources the whole design process to the crowd, letting them to propose new ideas and choose those that will be manufactured. Nevertheless, there is no mechanism for sharing the specifics of the underlying process nor study its applicability and effectiveness in other contexts. In fact, a systematic review in crowdsourcing taxonomy [5] revealed that *process* aspects in crowdsourcing projects are only related to the usage

[4]`http://www.threadless.com`

of aggregation or validation methods, and task parallelization. There is a clear gap on the systematic usage of processes, in the sense of several activities interconnected to solve a larger problem. To the best of our knowledge, Action-Verification units [76] are the first steps to generalize crowd coordination. In their work, they formalize a small pattern in which a worker performs a generic *action* and, then, a set of individuals verifies and scores its validity.

It is clear that there is an industrial need for monitoring crowdsourced tasks and processes that ensures quality and warns process owners in case of deviation, but, due to the focus on sparse interactions between individuals and companies in crowdsourcing platforms, current quality assurance mechanisms focus on small choices in the task design that may improve quality. Besides, due to the lack of a common language for communicating, sharing and studying crowdsourced processes, there is no guarantee that those design choices will work on every context or with a different crowd configuration.

Quality evaluation methods can be broadly classified in three main families: (i) by direct inspection of the job provider, (ii) automatic, and (iii) methods using the crowd itself as evaluator. Evaluation by the job provider has a scalability problem, since the crowd can produce a large amount of work that the job provider may not evaluate as she has a finite amount of resources. Clearly for most of the tasks, an automatic evaluation is either impossible or can only guarantee a minimum quality, otherwise it would be possible to set up a completely automated solution without human intervention. Crowdsource the quality evaluation of the jobs performed by the crowd has been proposed as an alternative [38, 58], but this solution has the potential problem of trustworthiness and management of opinion and criteria disparity.

With respect to quality assurance mechanisms prior to the task resolution, crowdsourced projects try to motivate workers in order to solve it efficiently while reputation mechanisms have been implemented to remove consistently low-performance individuals. In general, industrial applications pursue lucrative objectives and, because of this, these applications are more constrained in terms of motivating the crowd and tend to reward workers economically, although it is not clear if this increases productivity [46]. Moreover, financial rewards also attract individuals that want to benefit from their participation whilst providing poor solutions that may impact the crowdsourced output. Finally, non-reward prior mechanisms usually forbid a significant amount of individuals to participate in the problem resolution – increasing the time to complete the task. For instance, it has been empirically proven that narrowing the worker search to individuals in the United States increases the overall quality [65], even though only 50% of the crowd is from the United States [51].

Whereas in the improvement of quality after the task completion, repetition of the same task by several individuals and appropriate aggregation mechanisms ensure quality even on noisy scenarios [42, 89]. It is estimated that 30% of the crowd workers are producing low-quality outcomes, and hence it is expected that the majority opinion is an acceptable answer. Nevertheless, such approach requires to ask several individuals to perform the same tasks, impacting on the overall cost and duration of the project. Besides, this approach is only applicable when data can be aggregated or compared. Alternatively, some crowdsourcing project may use *gold standards* for measuring workers' performance against a set of questions with verifiable answers [105], filtering out simple individuals that do not put effort on solving the problem. Unfortunately, this solution is only available for those crowdsourced tasks in which there exists such a gold standard.

1.1.2 User Profiling Techniques in Crowdsourcing

Current worker management mechanisms in crowdsourced platforms are still primitive. Let's take Amazon Mechanical Turk[5], the most popular crowdsourcing platform, as an example. Crowdsourcers may upload their tasks with some restrictions on the type of individuals that may accept and perform it. Those restrictions can be based on the location of the individual, age, educational level, and some other demographics based on recurrent population surveys conducted by Amazon Mechanical Turk. But, the three most important restrictions are:

1. *Location.* It has been empirically proven that location of the crowd worker is correlated to the quality of their work [65] and, hence, crowdsourcing platforms offer this feature as part of the profile of the users.

2. *Ratio of successfully solved tasks.* After an individual has performed a task on the platform, the task owner must assess the quality of the worker performance. A positive answer would immediately (financially) reward the worker, while a negative answer reduces the ratio and do not reward the worker.

[5] `http://www.mturk.com`

3. *Qualifications.* Task owners may create a mandatory test that the workers must pass in order to be eligible to perform the task.

The combination of those two constraints are the core of the quality assurance tools implemented in crowdsourced platforms. The ratio of successful tasks is a basic reputation system, which has been proven efficient in other contexts. Nevertheless, it is quite easy to find task owners that accept all solutions without reviewing its quality because it is a mechanism for attracting more workers. Also, due to the easiness of creating, answering and evaluating your own tasks, people can easily create a false profile with high reputation. On the qualifications side, it ensures that the worker possess some abilities at the moment of answering the test. But it does not evaluate the evolution of the worker over time. Besides, individuals usually must solve several qualifications for the same skill as these tests are designed by the task owners and are not managed by the platform.

An interesting research line for quality assurance mechanisms is the work done by Rzeszotarski et.al. [87], and later reaffirmed by Kazai and Zitouni [53]. In their experiments, they showed that individuals interact differently with the platform and, in particular, those users who differ the most from the *average* individual are those that produce worse results. Those results open the possibility of creating a profile of the behavior of users in the platform. Such profiles would not only allow the platform to assess the anomaly of the task resolution (i.e. a potential bad-quality outcomes), but also to compare individuals, or create a predictor that links behavior to performance in the platform.

1.2 Objectives

Based on the aforementioned limitations of current Crowdsourcing techniques in the definition, monitoring of crowdsourced processes, and the limited capabilities of user profiling in digital platforms, we set the following objectives for this thesis.

Objective O.1: Supporting crowdsourcing methodologies in industrial scenarios by allowing the definition of crowdsourced processes and monitorization of quality.

Crowdsourcing has the potential to be an efficient alternative to outsourcing. Still, its popularity has not grow during the last years. Based on our analysis of the current usage of crowdsourcing platforms, we believe that the lack of a proper mechanism for defining processes, monitoring its execution and the lack of efficient quality assurance mechanisms are the main drawbacks for the industrial application of crowdsourcing.

Objective O.2: Design of a mechanism capable of discovering the *normal behavior* of a user in a digital platform.

Current quality assurance techniques in crowdsourcing are sustained in the idea that quality can be ensured when considering the *average* solution or idea. Based on the early work done by Rzeszotarski et.al. [87], we plan to continue the research on creating a *normality* score of the user's behavior in the platform by the means of comparing user's behavior and, hence, detecting anomalies that might be a threat to the quality of crowdsourced processes.

We plan to study the usage of process mining techniques as a mechanism for summarizing the sequence of actions performed a user in the platform. A process model may be the perfect representation of the user behavior, as it allows to predict future behavior and generalize the user's behavior. With such a process model, we will be more close to a smart automatization of the platform. For example, automatic assignation of tasks to the best workers in the platform, or categorization of users based on their expertise, could then be possible.

1.3 Contributions

Figure 1.1 positions our contributions in relation to the involved actors in decentralized problem resolution: Users (or workers), processes and the platform. Some subtopics have been depicted towards the resolution of the three aforementioned objectives.

On the bottom half of the diagram, one may find solutions related to users and processes without the commitment of the platform. State-of-the-art quality assurance mechanisms in crowdsourcing focus on this arena, by proposing processes, designing mechanisms for aggregating information, or filtering out low-quality workers. In general, the platform only acts as an enabler of the collaboration, and leave the responsibility of monitoring processes, quality and workers to the task owner. Nevertheless, solutions in this space are typically tailored towards the specifics of the problem to solve.

On the other side of the diagram, the platform steps up and acts as a monitoring tool for both processes and workers. On the later, user profiling

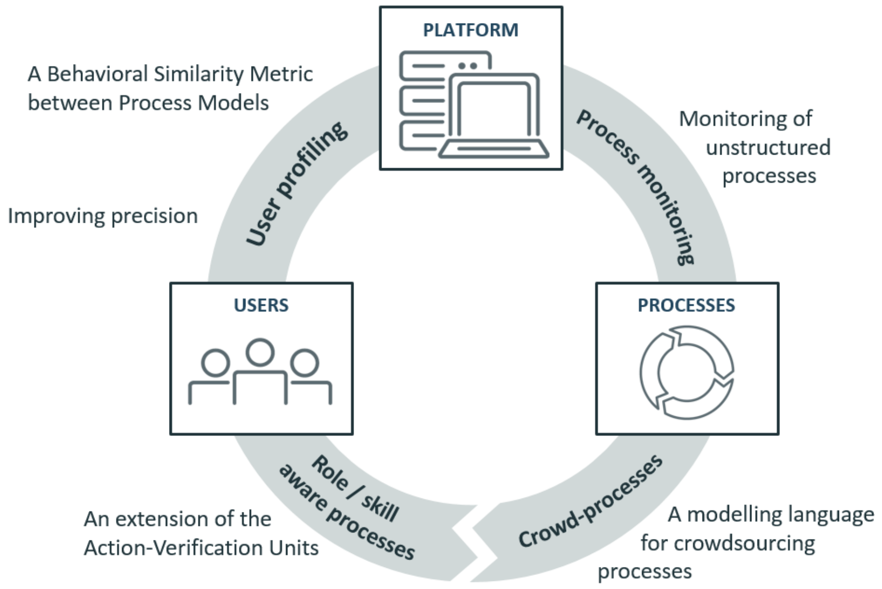
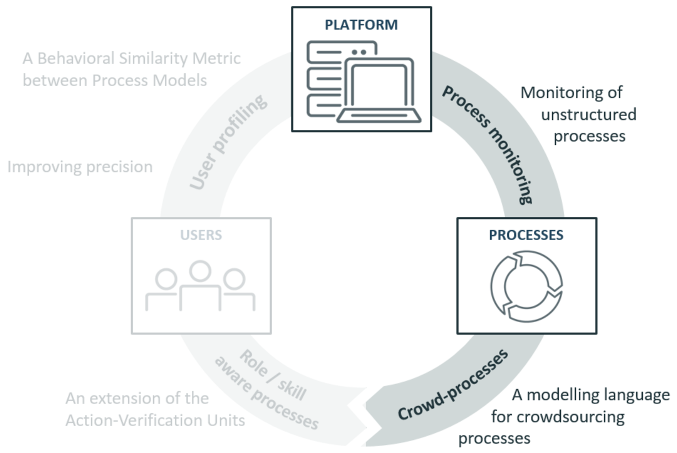

PLATFORM
USERS
PROCESSES
User profiling
Process monitoring
Role / skill aware processes
Crowd-processes
A Behavioral Similarity Metric between Process Models
Improving precision
Monitoring of unstructured processes
An extension of the Action-Verification Units
A modelling language for crowdsourcing processes

Figure 1.1: Framework of our contributions and their relationship towards the three actors in any decentralized collaborative process. The platform is acting as a technology provider and monitoring tool, while processes are governing how users should interact and collaborate.

is highlighted as recent findings pointed out to the possibility of detecting low-quality workers as those individuals whose behavior deviate from the average. On the former, the platform must monitor the execution of process and predict its outcomes, warning task owners in case of suspicious activities or predicted low-quality executions.

Despite this clear dichotomy of solutions with, and without, the commitment of the platform, we have structured this thesis and contributions based on the two objectives: supporting the design and monitoring of collaborative processes (Objective O.1) and lack of user profiling techniques (Objective O.2). Bellow one can find a list of the specific contributions on both arenas.

Description and Monitoring of Collaborative Processes


PLATFORM
A Behavioral Similarity Metric between Process Models
User profiling
Improving precision
Process monitoring
Monitoring of unstructured processes
USERS
PROCESSES
Role / skill aware processes
An extension of the Action-Verification Units
Crowd-processes
A modelling language for crowdsourcing processes

1. A modelling language for crowdsourcing processes.

   We propose a graphical modelling language for Crowdsourcing processes capable of tackling with industrial needs. More specifically, deadline and worker management. Besides, the graphical part of the language puts emphasis on the aggregation mechanisms used on the process, that is usually an indicator of the quality assurance mechanisms implemented.

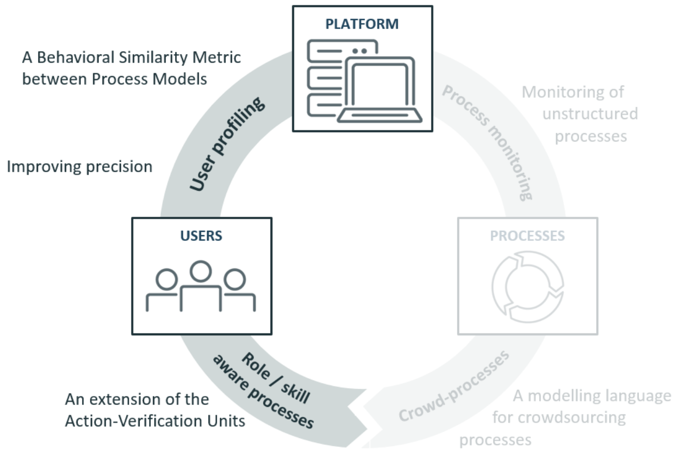
   An implementation of this modelling language will set the foundation for designing further monitoring tools, a simulation tool for helping in the design of crowdsourced processes, or predicting the expected quality (or delays) based on the behavior of the crowd. Moreover, it will help on the dissemination of good practices in crowdsourcing processes.

2. Monitoring of unstructured processes.

   On the other hand, it is expected that some crowdsourced projects will not be governed by a clearly defined process. Instead, the involved actors collaborate in deciding how the task should be performed. For instance, IT support center solve customer issues by establishing a conversation between the two sides. As the conversation evolves, the support engineer refines the root-cause search through the conversation.

   In order to advance in this area, we propose an unsupervised method for combining events based on the semantically similarity of event names. This is an opportunity for process model discovery methods, in which domain-knowledge on events is needed in order to generalize and simplify the process model.

Managing an Elastic Pool of Workers


PLATFORM
A Behavioral Similarity Metric between Process Models
User profiling
Improving precision
Process monitoring
Monitoring of unstructured processes
USERS
PROCESSES
Role / skill aware processes
An extension of the Action-Verification Units
Crowd-processes
A modelling language for crowdsourcing processes

1. An extension of the Action-Verification [76] units for measuring skill acquisition.

   The Action-Verification units are a small collaborative process in which one individual performs an action, and then the crowd verifies its correctness. We built a mechanism for measuring the skill acquisition of individuals in the platform, allowing us to create a more detailed profile of the expertise of crowd workers.

2. A method for improving the precision of process models.

   This thesis relies on the hypothesis that human actions can be though as event in a system, and, therefore, human behavior could be modelled as a process model. Unfortunately, most current process discovery methods produce models without repeated activities. Such constraint may induce an over-generalization of iterative processes, reducing the effectiveness of process discovery techniques for discovering a model of human behavior. We describe an algorithm capable of improving precision of loops in structured process models so that a precise description of workers is always obtained.

3. A behavioural similarity metric between process models.

   As a first use case of the process models as user profiles, we propose an extension of the Cophenetic distance [23] for measuring similarities between process models and, hence, between individuals' behavior. The benefit for using this extension is that it captures some differences on the behavior of the model while being efficient on cost and memory.

The following table links the contributions of this thesis to their corresponding publications, patents, and the structure of this thesis – which is described in detail afterwards.

Process Design and Monitoring	**Thesis Structure**
A modelling language for crowdsourcing processes David Sánchez-Charles, Victor Muntés-Mulero, Marc Solé, and Jordi Nin. *CrowdWON: A Modelling Language for Crowd Processes based on Workflow Nets*. In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, AAAI'15, 2015.	Chapter 3

Worker Profiling and Monitoring	**Thesis Structure**
An extension of the Action-Verification Units David Sánchez-Charles, Jordi Nin, Victor Muntés-Mulero and Marc Solé. *Worker ranking determination in crowdsourcing platforms using aggregation functions*. In Proceedings of the 2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), 2014.	Chapter 5
A method for improving the precision of process models David Sánchez-Charles, Josep Carmona, Victor Muntés-Mulero and Marc Solé. *Improving Process Model Precision by Loop Unrolling*. In Pre-Proceeding of the Sixth International Symposium on Data-driven Process Discovery and Analysis, 2016.	Chapter 6
A behavioural similarity metric between process models David Sánchez-Charles, Josep Carmona, Victor Muntés-Mulero and Marc Solé. *Process Model Comparison Based on Cophenetic Distance*. Business Process Management Forum, 2016. David Sánchez-Charles, Josep Carmona, Victor Muntés-Mulero and Marc Solé. *Clustering Software Developer Repository Accesses with the Cophenetic Distance*. In Pre-Proceeding of the Sixth International Symposium on Data-driven Process Discovery and Analysis, 2016.	Chapter 7

During the development of this thesis, the following patents were filed to the United States Patent and Trademark Office (USPTO) as an internal exploitation at CA Technologies. The key concepts of these patents are partially described in Chapters 4, 6 and 7.

Filed patents	**Thesis Structure**
David Sánchez-Charles, Jaume Ferrarons Llagostera, and Victor Muntés-Mulero. Filed patent: *Use of process mining techniques and natural language processing to classify textual conversations based on automatic detection of topics and topic evolution.* David Sánchez-Charles and Victor Muntés-Mulero. Filed patent: *Escalation prediction Based on Timed State Machines.*	Chapter 4
Marc Sole Simo, David Sánchez-Charles, Victor Muntés-Mulero and Josep Carmona. Filed patent: *Increasing the precision of mined process models with loops.*	Chapter 6
David Sánchez-Charles, Steve Versteeg, Victor Muntés-Mulero, Marc Sole Simo and Li Sun. Filed patent: *User Behaviour Anomaly Detection Based on Process Mining Patterns.*	Chapter 7

1.4 Outline

An introduction to Crowdsourcing, including a review of the nomenclature and existing quality-assurance techniques, and a background review of process modelling are discussed in Chapter 2.

The first part of this thesis focus on techniques for improving the monitoring of decentralized collaborative processes. In chapter 3 we introduce a graphical modelling language for Crowdsourcing processes. The described language emphasizes on the worker and deadline management of processes, two indispensable aspects for industry. Besides, the graphical language also highlights the aggregation mechanism implemented in the processes, allowing a first assessment of the risk of low-quality outcomes.

Then, we explore how the platform may help in the monitoring and definition of collaborative processes. Chapter 4 reviews novel Natural Language Processing techniques, that could help practitioners in the monitoring of processes that are not yet elicited. In particular, we describe a method capable of combining events with different names, but semantically similar. The merging of events allowed us to find a graphical representation of the underlying process.

On the second part of this thesis, we refocus our efforts on improving the monitoring of users participating in the platform. First, in chapter 5, we review an existing patterns to iterative increase quality of translations in a CrowdSourcing platform, and extend it in order to design a mechanism capable of ranking workers.

Then we advance to tackle the issue of automatically generate a profile of the workers based on the actions taken in the platform. In this thesis, we assume that the human behaviour can be summarized in the form of a process model. Quality of such process model is a key point for understanding its behaviour and, in Chapter 6, we present a technique to improve the precision of process models.

Continuing with the automatic user profiling, we formalize a extension of the Cophenetic distance in Chapter 7 in order to tackle the issue of measuring the similarity of two process models and, hence, the similarity of two users in the platform. We also apply this technique in the problem of making groups of users with similar behaviour.

Finally, Chapter 8 concludes this thesis and presents some future lines of research.

CHAPTER 2 Background

We start this chapter by introducing the task resolution model in Crowdsourcing platforms, and then we illustrate the usage of this technology with some successful and on-going projects. Then we highlight the most common quality assurance mechanisms used in Crowdsourcing as this is currently the only quality monitoring present in crowdsourcing. Then, we review the terminology in Business Process Management and some graphical modelling languages.

2.1 Crowdsourcing

The term *crowdsourcing* was coined in 2005 by Jeff Howe to describe a recent trend in businesses that *outsourced tasks to the crowd* [45]. Even though Jeff Howe provided a definition of crowdsourcing in his article, the scientific community did not accept this definition and several slightly different concepts of crowdsourcing coexist in the literature. After reviewing more than 40 definitions, Estellés-Arolas et.al. [34] proposed the following integrated definition of crowdsourcing.

Definition 2.1 (Estellés-Arolas et.al. [34]). *Crowdsourcing is a type of participative online activity in which an individual, an institution, a non-profit organization, or company proposes to a group of individuals of varying knowledge, heterogeneity, and number, via a flexible open call, the voluntary undertaking of a task. The undertaking of the task, of variable complexity and modularity, and in which the crowd should participate bringing their work, money, knowledge and/or experience, always entails mutual benefit. The user will receive the satisfaction of a given type of need, be it economic, social recognition, self-esteem, or the development of individual skills, while the crowdsourcer will obtain and utilize to their advantage that what the user has brought to the venture, whose form will depend on the type of activity undertaken.*

Notice that for the rest of this thesis, we will deliberately violate such definition by considering crowdsourcing as an activity in which not only individuals participate, but also intelligent machines may solve and propose problems to solve. Besides, we believe that crowdsourcing processes should be closer to internal business processes and, hence, the open call condition may be violated for some tasks performed or supervised by in-house workers. Still, we left intact the essence of a participative online activity in which a decentralized collaboration helps in the resolution of a problem.

Brabham's [16] classification of crowdsourced tasks provides an initial intuition of the potential of crowdsourcing, in which projects are categorized in 4 possible goals behind the collaborative resolution of tasks:

- **Knowledge Discovery and Management**. Ideal for information gathering, data completion, organization, and reporting problems.

- **Broadcast Search**. Ideal for ideation problems with empirically provable solutions.
- **Peer-Vetted Creative Production**. Ideal for ideation problems where solutions are matters of taste or market support.
- **Distributed Human Intelligence Tasking**. Ideal for largescale data analysis where human intelligence is more efficient or effective than computer analysis.

In the rest of this subsection we will review the nomenclature of crowdsourcing, and some examples of successful crowdsourced projects.

2.1.1 The Crowdsourced Task Model

Even though task execution in a crowdsourcing platform is not significantly different than in traditional in-house processes, it is worth the effort to revisit its execution and highlight some of its uniqueness.

Definition 2.2. *A* ***task*** *in a crowdsourcing platform is any action required for achieving an objective of the task owner. Such action is usually performed by an individual on the crowd, or a computer.*

The granularity and the level of details of the task are up to the task owner. When a task cannot be trivially subdivided in smaller tasks, and the task description provides a clear procedure and guidelines for performing such action, one uses the term ***micro tasks*** *instead of task.*

Figure 2.1 depicts the flow of actions occurring during the task resolution. Two actors are involved in the task execution, the *Requester* is the agent

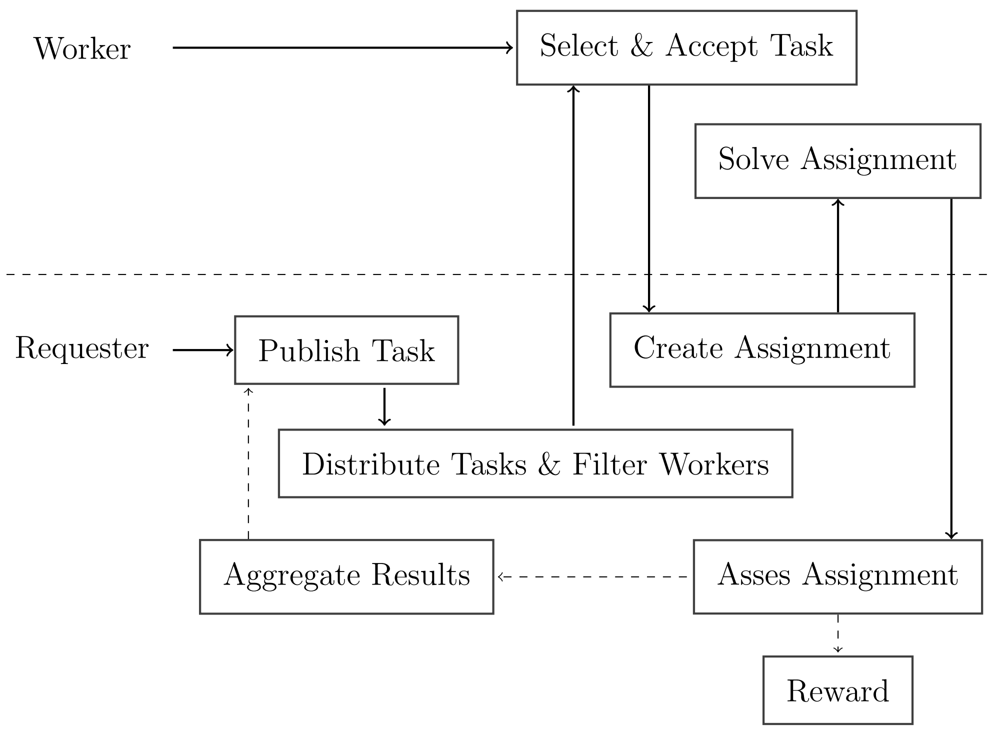


Figure 2.1: Simplification of the Task creation and execution process in a typical Crowdsourced platform. Dashed arrows are optional.

expecting results from the crowd and the *Worker* is any individual who joins the Crowdsourcing platform. Following, one can find some details on certain actions. Full details of the usual task execution can be found in [70].

Distribute Task & Filter Workers During task creation, the requester may establish some conditions on the workers who can contribute to the problem resolution. Then, the task details are distributed along workers who satisfy those requirements.

Such requirements may involve filtering out newcomers to the platform (by setting a lower threshold on the number of tasks already submitted), number of tasks submitted in the last day, passing a test, being part of a group of individuals or by limiting the physical location of the individual to a selected set of countries. Those requirements are usually a trade-off between ensuring high-quality outcomes and time to resolution.

Select & Accept Task Contrary to other task distribution models, crowdsourcing usually does not involve the assignment of a task to a particular individual, even though some research shows that this might increase quality. The most common approach is to show a list of tasks that are available to the user, which can choose whatever task feels more comfortable.

Create Assignment The task itself is a description of the problem to be solved, as well as a template to create multiple assignments. But data is not send to users until they accept the task. The template contained in the task is used to visualize the data to the user, as well as establishing the tools and rules to submit the answer or solution.

This model allows the requester to ask users to solve a batch of the problem, i.e. only a small piece of the problem. How these batches are distributed among crowd workers is up to the requester, and this choice may impact on the final quality of the project. For instance, one may send the same batch to several individuals to retrieve the *most popular answer*. Another strategy would be to append some questions with an already known answer, so the requester can quickly assess the quality of the solution.

Asses Assignment The final responsibility of assessing the quality of the assignment is up to the requester. Most Crowdsourcing platforms allow requesters to flag those assignments that do not meet the expected quality. This information may be later used by the platform to filter tasks to a particular set of individuals. For example, a requester may establish that a worker must have 80% of their previous assignments accepted in order to be eligible to enroll the task.

Reward The requester may offer a reward to those workers to participated in an Assignment. There are multiple mechanisms for financially rewarding workers, the quantity may depend on the quality of their outcomes. Notice that the rewards are not received immediately after finishing an Assignment, as the requester must assess its quality first.

Aggregate Results After all assignments have been solved, the requester may need to aggregate the results into a unique outcome. The results may return to the crowd in the form of another Task.

2.1.2 Examples of Crowdsourcing projects

Amazon Mechanical Turk The basic idea behind Amazon Mechanical Turk is that anybody with a few available minutes can connect to their website and answer some questions, or solve small tasks, that may help a company to solve a problem. For instance, one may be asked to tag [106] or describe an image [85], to classify a set of objects [37], translate sentences [4], answer a survey [13], or any type of work that humans are expected to perform better than computers. By designing tasks that are solvable with low effort from the user perspective and providing small financial rewards, companies are able to solve the aforementioned problems within large datasets whilst controlling costs.

This marketplace of human work set the standard for most of the Crowdsourcing terminology and methodology, and proved that people is willing to spent some time in solving industrial problems in exchange of a financial reward. It also provided a perfect scenario for researchers to create their own experiments and get real data from humans.

Kaggle In some cases, the effort needed to solve a task is not worth a small reward of platforms as in platforms such as Amazon Mechanical Turk. And hence, extra financial reward and motivation are needed. Kaggle[1] provides a platform for companies to upload their data-dependant problems along with a suitable dataset, and then a competition starts to retrieve the best machine learning solutions from the crowd. Those who obtain the best accuracy and precision on their solutions get a prize, whilst the rest get nothing from their participation.

Such model is perfect for solving problems in which it is understood that there exists a solution, quality of such solution is measurable but the space of solutions is so huge that it is almost unfeasible to find by an individual. Each participant provides their knowledge and intuition to the resolution of the problem, and hence each individual tackles the problem with a different approach. As the number of individuals grows, the more solutions are

[1]`https://www.kaggle.com/`

explored.

Threadless Contrary to data-dependent problem resolution as in Kaggle, there are some problems in which quality of solutions are not easily measured. For instance, Threadless[2] asks the crowd to design and chose the T-shirts that they will sell on their online shop. Again, those designers that pass the crowd evaluation receive a monetary compensation as well as a share of the profit. But this company does not only crowdsourced the process of making the design, but also they ask the crowd to measure the quality of the received designs.

2.1.3 Quality Assurance in Crowdsourcing

One of the major drawbacks of crowdsourcing solutions is the lack of quality assurance mechanisms provided by existing platforms. Table 2.1 summarizes some proposed methods for requesters to implement, so they can have some control on the final quality. Quality Assurance mechanisms can be roughly divided on those prior to the task assignation, and those posteriors to the submission of the task. On the former, some requirements on the workers are defined or some examples of task designs are explained. On the latter, several comparisons of results are proposed and some review tasks are recommended. Unfortunately, all of them are on the requester-side, with few efforts from the platform side to palliate this issue.

[2]`http://www.threadless.com`

Prior-task QA		**Post-task QA**		
Worker	**Task Design**	**Redudancy**	**Control Group**	**Gold Standard**
Reputation [49]	Defensive task design [21]	Majority voting [90]	Validation Review [43]	Gold Standard [81]
Selective Assignment [44]	Granularity [55]	Majority Decision [43]	Multilevel review [14]	
	Bias / error distinction and recovery [48]	Multiple annotations [80]	Grading / voting	
		Repeated labeling [47]	Improving review [66]	

Table 2.1: Summary of Quality Assurance (QA) techniques used in Crowdsourcing.

2.1.3.1 Collaborative Processes

Research in crowdsourcing has greatly focused on the study of individual tasks, reducing the scope of quality assurance mechanisms to those studying the effects of particular properties of individual tasks. The design of collaborative workflows is reduced to some examples, and therefore there are not so many results linking quality to process design choices. In fact, very few tools are available to design and deploy sequences of tasks. TurKit [67] is a tool to automatically create tasks in Amazon Mechanical Turk and gather the solution from the platform. Its design allows administrators to create a flow of tasks. Besides, Jabberwocky [3] focuses in the reusability of small processes. CrowdWeaver [57] is the first tool to allow users define workflows with a visual interface.

The most discussed process in the literature is the **find-fix-verify** pattern, introduced by Soylent. Soylent [14] is an extension to Microsoft Word that allow users to ask for a revision (correct it, or make it shorter) of a text using crowdsourcing. In the study, they noticed that asking individuals to perform the whole revision was not feasible and quality was below expectations. Then, they decided to split the task in three steps:

1. In the **find** phase, individuals are asked to find any sentence, or paragraph, that needs improvements.

2. In the **fix** phase, several individuals propose a modification of the sentences found in the previous phase.

3. Finally, in the **verify** phase, individuals choose the best fix proposed by the community.

2.1.3.2 Gold Standards

Adding a human review to assess quality is costly, therefore everything that we can implement in order to automatically reject useless work is welcome in crowdsourcing platforms. In this category, crowdsourcing has traditionally focused on asking individuals to solve task with known answer (or golden truth) and tasks not-yet solved. That way, one can measure the quality of the working by assessing the answer to the golden truth data set. The problem with this approach is that not all tasks may have a golden truth, and in some cases the generation of this data set is too costly.

2.1.3.3 Ground Truth and Majority Decision

Since the golden standard approach is costly (and sometimes very difficult to implement), the Amazon Mechanical Turk team recommends to design tasks with quality in mind. One of their recommendations is to add comparable questions that are difficult to answer without doing the task itself. For example, one can ask workers to summarize a text in only three keywords. Honest workers would probably agree on at least one keyword.

When possible, another used quality control mechanism is assigning the same task to several individuals. Then an aggregated solution is considered as the valid solution [9,26,72]. There are several ways of aggregating information, and sometimes it is useful to ask the crowd to aggregate the information provided by themselves [56]. When possible, an outlier detection mechanism is used to maximize the utility of the aggregation mechanisms by removing noise from the received contributions.

2.2 Business Process Management

In this section, we will informally review the basics of business process modelling and process mining. For more details on both arenas, one could check [109] for an in-depth review of process modelling and [101] for an introduction to process mining. Afterwards, a more in-depth introduction to 3 business process modelling languages is provided.

The idea of business processes was born from the observation that businesses' outcomes are derived from the combination of several activities performed. Business processes are the key instrument to organize these activities and to improve the understandability of their interrelationships.

Most of nowadays business' activities are performed by humans and supported by an information system, which provides the necessary tools for enabling humans to efficiently perform the activity, but their automatization is being increasingly demanded by the industry and, hence, information systems should be able to cooperate with human workers. A company can efficiently and effectively achieve its business goals only if people and other enterprise resources play together well. Again, business processes are the key instrument to govern this collaboration.

Definition 2.3 (Mathias Weske [109])**.** *A* ***business process*** *consists of a set of activities that are performed in coordination, with the objective of achieving a particular milestone or business goal.*

Each business process is tailored for a single organization, but it may interact with business processes performed by other organizations.

In nowadays application economy, companies must quickly react to changes and must be able to continuously improve their performance. While at an organizational level, business processes are essential to understand how companies operate, business processes also play an important role in the design, analysis and optimization of businesses' execution. In the business process management arena, business practices and computer science collaborate for helping businesses to achieve their goals in the most efficient manner.

Definition 2.4 (Mathias Weske [109])**.** ***Business process management*** *is a set of methodologies and tools to support the design, administration, configuration, monitoring, and analysis of business processes.*

Traditionally, business processes are enacted manually, guided by the knowledge of the company's personnel and assisted by the organizational regulations and procedures that are installed. Enterprises can achieve additional benefits if they use software systems for coordinating the activities involved in business processes. These software systems are called business process management systems.

Definition 2.5 (Mathias Weske [109])**.** *A* ***business process model*** *consists of a set of activities and a execution strategy, i.e. constraints specifying when an activity may be performed based on the status of previous activities. A* ***business process instance*** *represents a particular execution of the activities, which were performed for a particular business case.*

2.2.1 Business Process Modelling Languages

Several languages have been defined for visually depict business process models. During this thesis, we will use four different business process modelling languages. It is well known that the choice of the modelling language assumes some underlying properties of the business process and its execution, and, hence, we will choose the most appropriated language depending on the context and later usage of the process model. For instance, the Business Process Modelling Notation[3] (Section 2.2.1.1) is well known among business people and practitioners as understandability of processes is one of their objectives. Nevertheless, execution of models described using the BPMN is not straightforward and other notations have been considered for such purposes. For example, the Business Process Execution Language[4], Petri nets (Section 2.2.1.2) and colored Petri nets (Section 2.2.1.3) cover the necessary formalization for execution of process models. Finally, we have also considered process trees (Section 2.2.1.4) during this thesis, as their structure provides interesting properties not seen in the other modelling languages.

2.2.1.1 BPMN, Business Process Modeling Notation

Business Process Modeling Notation (BPMN) is a standard for business process modeling that provides a graphical notation for specifying business processes as a flowchart. The objective of BPMN is to support business process management, for both technical users and business users, by providing a notation that is intuitive to business users, yet able to represent complex process semantics. In particular, a mapping to Business Process Execution Language (BPEL) is provided, allowing practitioners to automatize the execution of a process model designed with BPMN.

[3] `http://www.bpmn.org`

[4] `https://www.oasis-open.org/committees/tc`$_{h}ome.php?wg_{a}bbrev = wsbpel$

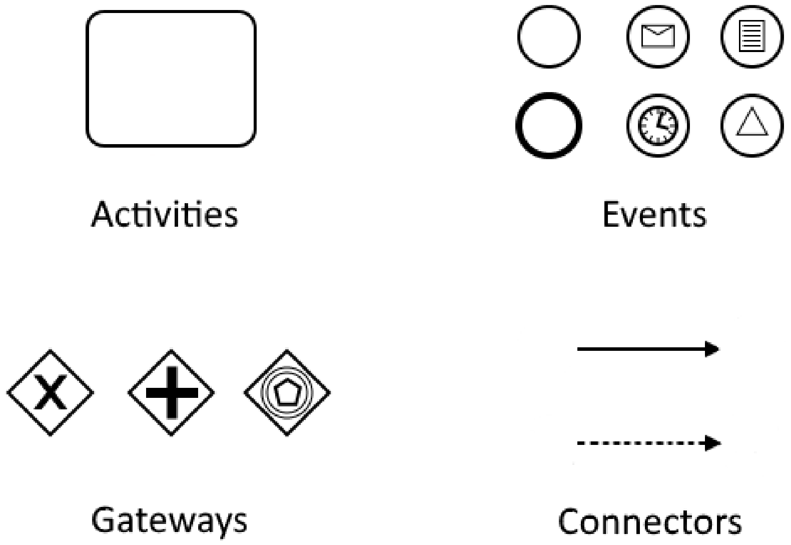


Figure 2.2: Some examples of the 4 main components in a BPMN process model.

Figure 2.2 depicts the 4 main components that are used for modelling a business process with BPMN: tasks, events, gateways and connectors. A brief description of each component is provided below. Besides the four main constructs, BPMN provides other tools for modeling synchronization between different actors, or roles, and other annotations.

Tasks represent a single unit of work that is done by the company or organization. Typically, it is an atomic activity, i.e. it cannot be broken down to a further level of business process detail, and hence is the lowest level specification of the business process. A task is represented with a rounded-corner rectangle and describes the kind of work which must be done.

Events are usually described as "*something that "happens" during the execution of the process model*". Typically, events have a *cause* (or trigger) and an *impact* (or result) on the execution of the process model. An Event is represented with a circle containing an icon, which denotes the type of event (e.g., an envelope representing a message, or a clock representing time).

Gateways are a graphical representation of the execution strategy for divergences and convergences of the flow in a process model. An internal marker identifies the type of gateway, expressing strategies such as *exclusive* (only one path is executed), *event based* (an event decides the flow of execution), *inclusive* (at least one of the following paths are executed) or *parallel* (all paths are executed concurrently).

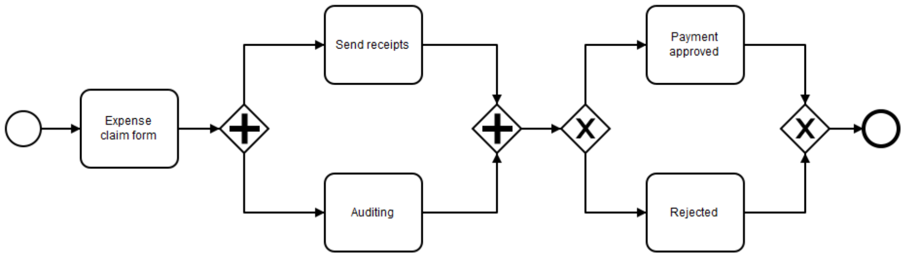


Figure 2.3: Example of an expense claim process, modeled using BPMN.

Connectors connect different components of the process model, and provide an ordering on the execution of the different tasks. Besides, it might also depict an exchange of messages between participants.

An example of a BPMN diagram can be found in Figure 2.3, describing an expense claim process. When an individual fills an expense claim form, she must send the receipts to the Finance department. While the documents are in transit to the Finance team, they might start their internal process for auditing the expense. Notice that finance may not need to check the original receipts, but they need to store them for later use. The result of the expense claim check might end in a refund of the expense to the worker, or in a rejection. This process model does not specify if the individual is allowed to fill a new expense claim form for a previously rejected expense claim.

2.2.1.2 Petri nets

A Petri net [95] is a graphical notation for describing distributed systems, yet its simplicity and expressive power made possible use Petri nets as the *de* *facto* modelling language for formalizing and studying systems in a plethora of areas.

Definition 2.6. *A **Petri net** is a tuple (P, T, A, M_0, W, K) such that*

- *$(P \cup T, A)$ is a bipartite directed graph.*
  - *Elements P are known as **places** of the Petri net, T are its **transitions**, and elements of A are known as **arcs**.*
  - *Every arc a is composed of a place p_a and a transition t_a.*
- *M_0 is the initial marking function, i.e. M_0 is a map between places and $\mathbb{Z}_{\geq 0}$.*
- *W is a function that maps arcs (A) to integers in order to define the flow of data in the net.*
- *K is a function that maps places (P) to integers in order to define the capacity of the net. It must satisfy that $M_0(p) \leq K(p)$ for all $p \in P$.*

In the rest of this thesis, we will assume that our Petri nets are labelled:

Definition 2.7. *A **labelled Petri net** is a tuple $(PN, label)$ such that PN is a Petri Net (P, T, A, M_0, W, K) and 'label' is a mapping between T and the universe of labels $U \cup \{\emptyset\}$. Typically, the universe of labels is a set of strings representing the business activities explained by the process model. One says that a transition t is **silent** if $label(t) = \emptyset$.*

One of the major benefits of Petri nets is the marking function. This marking function enables practitioners to model the initial status of the system, and then the petri net defines how this marking function evolves over time (i.e. the system behaviour). One usually says that *the place* p *has* $M(p)$ ***tokens*** instead of using the marking function nomenclature.

Definition 2.8. *Let* t *be a transition in a Petri Net.* $\bullet t$ *is the set of arcs incoming to transition* t*, and* $t\bullet$ *is the set of arcs outgoing from transition* t*.*

There exists a one-to-one relationship between places and arcs in $\bullet t$ *(altern.* $t\bullet$*), and, hence, both concepts will be alternatively used when referring to elements of* $\bullet t$ *(altern.* $t\bullet$*).*

Definition 2.9. *A transition* t *is* ***enabled*** *if, and only if,* $M(p_a) \geq W(a)$ *for all* $a \in A$*.*

Definition 2.10. *A transition* t *can be* ***fired*** *only if it is enabled for a given marking function* M*. After firing the transition,* M *is replaced by a new marking function.*

$$M'(p) = \begin{cases} M(p) - W((p,t)) & \textit{if } (p,t) \in A \\ M(p) + W((t,p)) & \textit{if } (t,p) \in A \\ M(p) & \textit{otherwise} \end{cases}$$

In particular, this transformation states that $W(a)$ *tokens haven been consumed (removed) from incoming places (i.e.* $a \in \bullet t$*) in order to execute transition* t*. Its outcome are* $W(b)$ *tokens that are added to its outgoing places (i.e.* $b \in t\bullet$*)*

Marking functions allowed practitioners to model the execution of a business process as a Petri net, in which transitions are activities performed by the organization, or a formalism for modelling the collaboration between activities. See Figure 2.4 for an example. This Figure depicts an example of

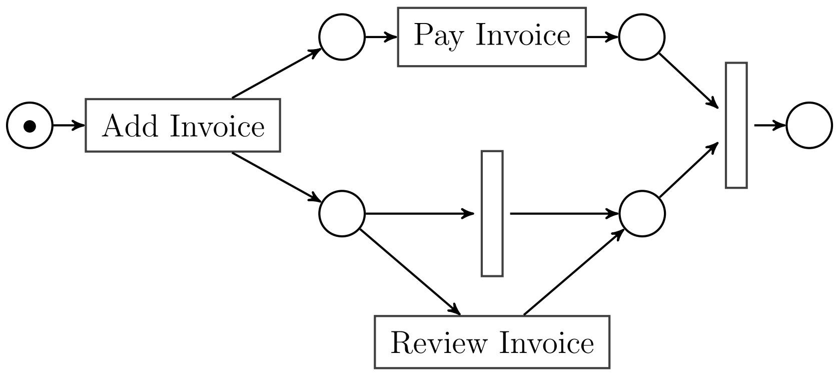


Figure 2.4: Example of a business process model, using a Petri net formalism. Bullets (●) represent tokens.

a business process model using the Petri net formalism. Three activities are depicted: *Add Invoice*, *Pay Invoice* and *Review Invoice*.

Figure 2.5 depicts the same process model after firing transition *Add Invoice*. One token was consumed from the initial marking, and one token was generated for each outgoing place. Since we did not define function W and K, we are assuming that we have infinity capacity and $W(p) = 1$ for any place. Two tokens are not available, enabling up to three transitions: *Pay Invoice*, *Review Invoice* and an unlabeled transition.

Figure 2.6 depicts the previous process model after firing transition Pay Invoice. Notice that only two transitions are now enabled: *Review Invoice* and the unlabeled transition in the middle. The right-most transition is not enabled, as it is expecting a token from each incoming place. The next step in this process would be to perform *Review Invoice* or to simply skip it by firing its corresponding unlabeled transition. Afterwards, the right-most transition

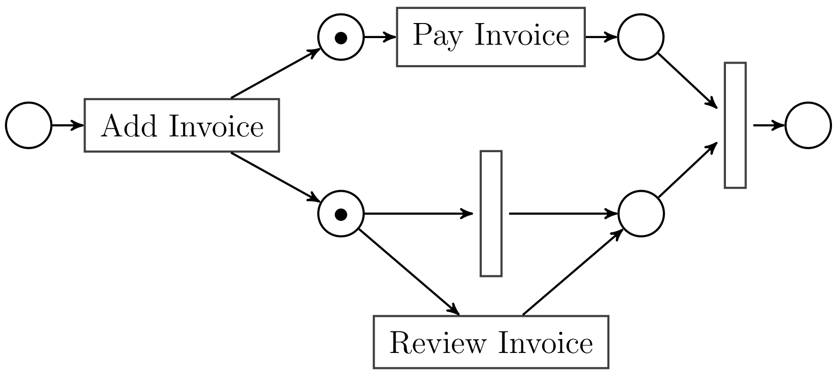


Figure 2.5: Example of a business process model after firing transition *Add Invoice*. Bullets (•) represent tokens.

could be fired and the execution of the process would have ended.

The Petri net of the previous example belongs to a particular subclass of Petri nets named *Workflow nets*. This subtype of nets have been largely used for modelling business process models, as there is a special *entry place* (a place with no incoming edges) and a place modelling the end of the process (a place with no outgoing edges).

Definition 2.11. *A* ***workflow net*** WF *is a tuple* (P, T, A, W, K) *such that*

- $(P \cup T, A)$ *is a bipartite directed graph.*
- *There exists a place* p_s *(resp.* p_e*) such that its in-degree (resp. out-degree) is* 0.
- (P, T, A, M_{p_s}, W, K) *is a Petri Net, where* M_{p_s} *is a function such that* $M_{p_s}(p_s) = 1$ *and* 0 *for any other place.*

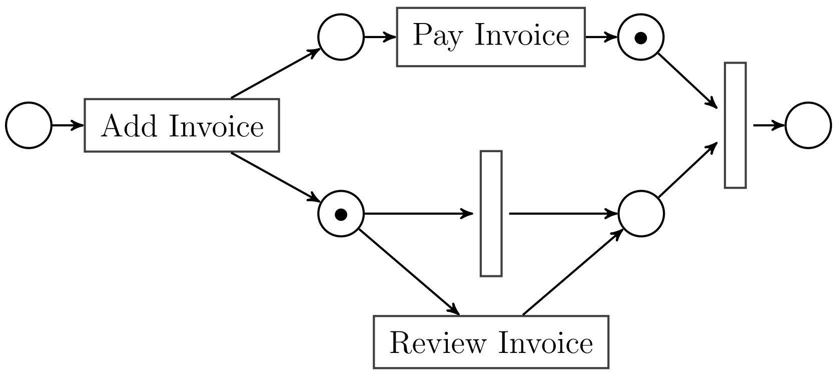


Figure 2.6: Example of a business process model after firing transition *Pay Invoice*. Bullets (•) represent tokens.

2.2.1.3 Colored Petri Nets

Colored Petri nets [50] are an extension of Petri nets in which tokens are enriched with data and, hence, increases understandability and expressive power of the models. For instance, colored Petri Nets allow models to clearly define conditions on when a transition may be fired, whereas transitions are usually triggered by an external agent in traditional Petri Nets.

The information assigned to a token is formalized with two concepts: Colour types and colours. The first is the description of the data structure that will be used to store the information, allowing the formal definition of conditions over such data, whereas colours are a particular instantiation of the just mentioned data structure.

Definition 2.12. *A* ***colour type*** *is a specification of the data associated to a token in a Petri Net. This object is similar to types in a programming*

ID	**type**
User ID	*String*
Has English certificate	*Boolean*
Score	*Integer*

Table 2.2: Definition of a colour type modelling a basic user profile. Apart from the user ID, this colour also enables us to state if the user has an English certificate, and the obtained score.

language, and, hence, they can be arbitrarily complex. In this Thesis, we will informally simplify the definition of type to a collection of pairs (ID, type) in which

1. *ID is a unique identifier of the type.*
2. *type is either*
    - *another colour type*
    - *one of the primitive types allowed by any programing language. I.e.* ***Boolean****,* ***string****,* ***float*** *or* ***integer****.*
    - *a* ***tuple****,* ***sequence****, or* ***set*** *of simpler colour types.*

The symbol Σ will denote the set of all possible colours.

Table 2.2 depicts an example of a colour type that might be used for describing the profile of a user. In this particular example, a token might state if a user with *User ID* has an English certificate and the score that the user obtained during the last certification exam.

ID	**type**	**Token 1**	**Token 2**	**Token 3**
ID	*String*	*W1*	*W1*	*W2*
Has English certificate	*Boolean*	True	True	False
Score	*Integer*	10	90	0

Table 2.3: Different colours for the colour type defined in Table 2.2.

Definition 2.13. *The* ***colour function*** $C : P \rightarrow \Sigma$ *assigns a color type to each place.*

The colour type of a place P specifies the type of data that tokens must contain while being in state P. Then, the colour of the token is a particular instance of the colour type.

Definition 2.14. *We will denote by* $Colour(\sigma)$ *to the set of all the possible values following the data structure defined by* $\sigma \in \Sigma$. *Additionally, there exists a function* $value_{ID}$ *such that it returns the value associated to the variable* ID*, if exists, for a given token.*

For the sake of understandability, it will be understood that $value_{ID}(token)$ is considered whereas ID appears outside the colour type definition. To which token is applied this function, will be understood from the context.

See Table 2.2 for an example of three instances of the same colour type. *Token 1* and *Token 2* share the same *User ID*, as they are modelling the same individual but in two different stages: Initially, this user has a score of 10 and, after retaking the exam, her score went up to 90.

In order to facilitate the definition of new instances, we will use the notation $\langle \{ID = value_{ID}\}_{ID} \rangle$ to indicate the values associated to a token. For instance, Token 1 in Table 2.3 could be represented as $< ID = W1$, $HasEnglishcertificate = True$, $Score = 10 >$.

Definition 2.15. *An* ***expression*** *is any function that maps a list of tokens to an element of Colour(σ) for a given $\sigma \in \Sigma$. In the particular case of $\sigma = \{(Guard, Boolean)\}$, we will also use the name* ***guard expression****.*

Definition 2.16. *A* ***marking*** *function is a function that maps elements in P to a multiset of tokens.*

Definition 2.17. *A colored Petri Net is a tuple ($\sum$, P, T, A, C, G, E, I) satisfying the following requirements:*

1. *(P, T, A) is a Petri Net and the graph ($P \cup T$, A) is a Workflow Net.*
2. *$\sum$ is a finite set of non-empty types, called* ***colour sets****.*
3. *C is a* ***colour*** *function. It is defined from P into $\sum$.*
4. *G is a* ***guard*** *function. It is defined from T into a guard expression.*
5. *E is an* ***arc expression*** *function. It is defined from A into an expression.*
6. *I is an* ***initial marking****.*

In the rest of the subsection, we review the semantics and execution of colored Petri Nets.

Figure 2.7 depicts the most basic structure in colored Petri Nets: one transition (labeled as TASK) and two places. Every transition must have at

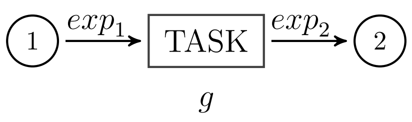


Figure 2.7: As in traditional Petri Nets, every transition is connected to places. Transitions are the elements in the model that are allowed to move tokens from incoming places (place 1 in the figure) and putting them on all outgoing places (place 2). Functions exp_1 and exp_2 define how this transformation is performed, and function g decides if the transition may be fired.

least one incoming and one outgoing edge, and always connecting the transition to other places. Besides, places are not allowed to be directly connected to other places. Additionally, colored Petri Nets add two expression functions (exp_1 and exp_2) to the edges from/to a place, and a guard function g to the transition.

In traditional Petri Nets, a transition is said to be enabled if there is at least one token in each incoming place. In the case of Figure 2.7, the marking function must state that the place 1 has at least one token. When a transition is enabled, then it may be fired. This is the process of consuming one token from each incoming place, and create one new token in each outgoing place. This is formalized by modifying the marking function. Again, in the case of Figure 2.7, a firing of the transition *TASK* would lead to a marking equation in which place 2 has one more token, and place 1 has lost one token.

In addition to the token count condition in Petri Nets, guard functions of transitions in coloured Petri Nets may specify when a transition is enabled based on the information associated to tokens.

Definition 2.18. *Let t be a transition in a colored Petri Net CPN, and M a marking of CPN. The transition t is **enabled** if, and only if,*

$$\forall a \in \bullet t, \exists k_a \in M(a) \text{ such that } g_t(\{e_a(k_a)\}_{a\in\bullet t}) \text{ is true.}$$

Where k_a is a token in place a, g_t is the guard function of t, and e_a is the expression of the arc a.

Intuitively, this definition states that a transition t is enabled if there is a selection of tokens, which are known as a **binding** of the transition, such that their combination satisfies the guard expression g_t. If there exists such a selection, then firing the transition will consume such tokens and create new tokens after the transformation described on the respective expressions. The next definition formalizes this concept. Notice that if there is no expression defined for an edge, we will assume that the identity function is considered.

Definition 2.19. *Let t be a transition in a colored Petri Net, enabled for a particular marking M. A **firing** of the transition t is the following transformation in the marking M:*

1. *For each $a \in \bullet t$, find a token $k_a \in M(a)$ such that*

$$g_t(\{e_a(k_a)\}_{a\in\bullet t}) \text{ is true}$$

2. *Define a marking M' as follows*
    (a) *For any place not in $\bullet t$ nor $t\bullet$, $M'(p) = M(p)$.*
    (b) *For any place in $\bullet t$, $M'(p) = M(p) \setminus k_p$*

(c) For any place in $t\bullet$,

$$M'(p) = M(p) \cup \{e_p(\{e_a(k_a)\}_{a \in \bullet t})\}$$

2.2.1.4 Process Trees

During this thesis, apart from traditional notations for business process modelling as Petri Nets and BPMN, we will use the process tree notation for defining a series of techniques and metrics. The process tree notation assumes that the process model has a certain structure, that allows practitioners to define the process model as a tree in which every subtree is a valid subprocess.

As we already defined, a business process model is a directed graph consisting of activities $\mathcal{A}$, some edges $\mathcal{E}$ connecting activities and control elements $\mathcal{C}$, which defines the execution relationship between activities. Typically, control elements are any of the following types *start finish*, *split-choice*, *join-choice spilt-parallel*, *join-parallel.*

A **rooted tree** is a directed graph with a distinguished node, called the root, from which every node can be reached with exactly one path. A **weighted rooted tree** is a pair (T, ω) consisting of a rooted tree T and a weight function $\omega : E \rightarrow \mathbb{R}_{>0}$ that associates every arc $e \in E$ a non-negative real number $\omega(e) > 0$. A **labeled rooted tree** is a rooted tree T such that there exists a mapping between a subset of the nodes of the tree and a set of labels S.

Definition 2.20. ***Structured process*** *imposes extra conditions on the control elements of a process: all split-parallel nodes (resp. split-choice) must have a unique corresponding join-parallel node (resp. join-choice) such that*

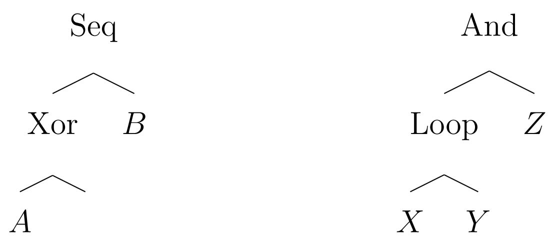


Figure 2.8: Example of two structured processes represented as Process Trees.

all paths connecting these two nodes must visit zero or two of any other pair of control elements. This correspondence is unique in the sense that if two split nodes u and v have the same corresponding join node, then u and v are the same node.

This definition allows us to consider structured processes as smaller subprocesses or individual activities that are interconnected via control elements.

Definition 2.21 (Buijs et.al. [18])**.** *A **process tree** is a labeled rooted tree T in which activities are represented as leaves of the tree and internal nodes describe the control-flow of the process.*

Figure 2.8 depicts two processes trees modelling all possible control elements. On the left, a choice construct is executed before activity (or subprocess) B. Notice that the absence of a label indicates a silent transition, and hence $\{B, AB\}$ is the language of the left model. On the right, a *parallel* construct and a *loop* are depicted.

Let $T = (V, E)$ be a rooted tree. Whenever $(u, v) \in E$, we say that v is a child of u and that u is the parent of v. The nodes without children are the leaves of the tree, and the other nodes are called internal. Whenever there exists a path from a node u to a node v, we say that v is a descendant of u and also that u is an ancestor of v. An internal node is **elementary** if it only has one child. The **depth** of a node u in a tree T, denoted by $\delta_T(u)$, is the sum of the weights of the arcs in the path from the root to u. Weights are usually set to 1, but we will later see that we can encode behavioural information from the process by modifying these weights.

We say that a process tree is **deterministic** if there is a one-to-one mapping between activity labels and leaves of T. For the sake of simplicity, we will label internal labels as OR[5], AND, SEQ and $LOOP$ to represent the usual behavioural structures in a process model. We will also denote these internal nodes by **gateways**, following the BPMN nomenclature. We allow silent activities by labeling them as $\emptyset$.

Definition 2.22. *A process tree is* ***reducible*** *if there are elementary nodes, silent transitions hanging over a gateway other than OR, or there exist a pair of internal nodes u and v such that (u, v) is an edge in the graph and both model the same type of gateway.*

Any *reducible* process tree can be converted into an *irreducible* tree by merging all conflicting nodes. We will suppose that all process trees are given in its irreducible form. Figure 2.9 depicts an example of a reducible process tree and its irreducible counterpart.

[5]Following the semantics of block-structured models in [18], only exclusive ORs are modeled.

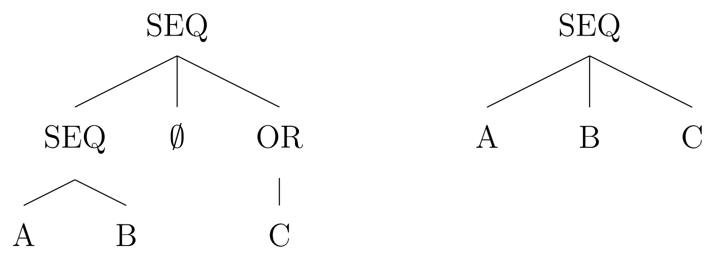


Figure 2.9: Two process trees modeling exactly the same behaviour. The left model is reducible, and the right model is its irreducible representation. The silent transition ∅ is removed because it is not part of an OR structure. The OR elementary node does not provide behavioural information.

2.2.2 Process Mining

A business process model acts as a blueprint for the execution of a business process instance, also known as **traces**. Each activity performed during a business process instance is also depicted as an **event** of the trace. Notice that some variations from the activities, and their interdependency, might be found in the process instances, as mistakes and some deviations are expected and accepted.

Definition 2.23. *An* ***event log*** *is an ordered set of traces for a (possibly unknown) business process model.*

An event log is typically linked to a known business process model. Nevertheless, it is also common that information system registers data from the organization that is yet not organized, or formalized, as a business process model. In that case, some techniques (known as *process discovery* or *process* *mining*) help practitioners to find a business process model that summarizes the event log.

Definition 2.24. *A **process discovery** technique is a method that, given a log L, produces a business process model N that approximates the real business process behind the log L.*

For measuring the quality of a business process model with respect to an event log, researchers and practitioners defined a series of metrics that compares the event log with the set of possible business process instances that the process model is capable to generate, which is known as the language of a process model.

Definition 2.25. *Given a process model N, we define the language of the process model, $\mathcal{L}(N)$, as all the possible business process instances of N.*

Business process **conformance checking** is a family of process mining techniques to compare a process model with an event log. It is used to check if the actual execution of a business process, as recorded in the event log, conforms to the model and vice versa. The most popular conformance checking technique [100] is based on finding a mapping between events of a trace to the process model that best maintains the behaviour explained by the process model.

Definition 2.26 (Buijs et.al. [20])**.** *The quality of a business process model N with respect to a event log L can be measured with the following four metrics.*

- *The **replay fitness** is the ratio of elements in $L \cap \mathcal{L}(N)$ with respect to L. I.e. the ratio of traces that are indeed included in the language of the process model.*

- *The* ***precision*** *is inversely proportional to the ratio of* $\mathcal{L}(N) \setminus L$ *with respect to* $\mathcal{L}(N)$. *I.e. the precision of a process model is high if the ratio of business process instances not seen in the log is fairly low.*

- *The* ***generalization*** *of a process model is the capability of the model for generating process instances not yet seen, but very likely to be part of the real (unknown) business process model.*

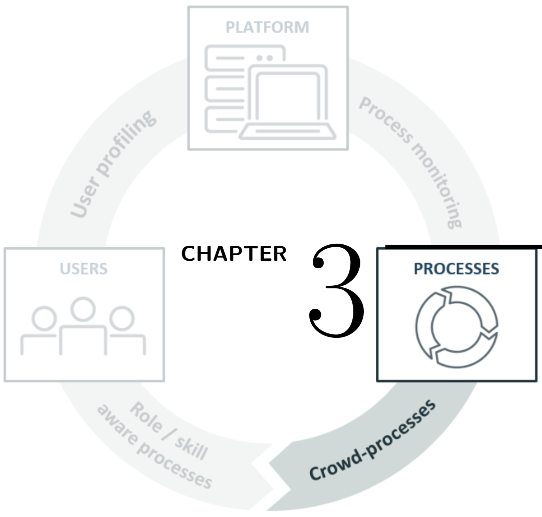
- ***Simplicity*** *is a score measuring how simple is the structure and behavior of a business process models.*

It is well known that none of these metrics is enough by itself to measure the quality of a process model. For instance, several examples show that there exist models with a perfect replay fitness, but non-precise. Even though highly precise process models have usually high fitness, generalization and simplicity might be significantly affected. It is expected that a good process model is well balanced in the four aforementioned quality aspects.

Notice that *precision* is defined in terms of a ratio of elements in a set that, potentially, might contain infinite elements (in case of an iterative process). Because of that, approximate metrics have been developed for measuring precision. In particular, we will use in the rest of this thesis the *alignment* approach for measuring fitness [2] and its application to measuring precision [1], consisting in counting unused edges during the alignment of the log to the process model.

Part I

Process Design and Monitoring



CHAPTER 3 Business Processes in Crowdsourcing

Formalizing the work to be done in a process is the first step for improving the overall efficiency and quality of problem resolution. We have detected that there is a lack of mechanisms for defining industrial processes to be executed by the crowd and, therefore, we start this thesis by introducing a novel graphical modelling language for describing such decentralized collaborative processes. The focus of this work is to allow a quick overview and assessment of the implemented quality assurance mechanisms, as well as the necessary mechanisms to define worker requirements based on their profile and prior involvement in the execution of the process . In a longer-term vision, having well-defined processes will help in making more predictable the performance of any crowdsourcing project.

For those cases in which the process is yet not formalized, we have also made the first steps for designing a method capable of discovering processes by analyzing the factual work done in the platform. Check Chapter 4 for more details.

3.1 Introduction

Although business processes may require to solve complex tasks, crowdsourcing is mainly used to solve independent tasks. Previous work such as [14] shows that finding the proper combination of tasks to solve a complex problem using crowdsourcing is not straightforward. In fact, designing proper workflows is one of the major issues in crowdsourcing according to employees of the crowd-based company CrowdFlower[1] [57]. Although some previous works propose mechanisms to express collaboration patterns in a visual way, their expressive power is still not sufficient to describe the variety of current crowdsourcing processes. Just as an example, none of the visual workflow languages used for crowdsourcing allow users to define the complete workflow behind a continuously open competition such as those in Threadless[2] and Innocentive[3], in an easy way.

Recent research [91] shows the interest of industry on increasing the participation of in-house workers in crowdsourcing processes. In many of these applications, requesters need to express restrictions on the characteristics of individual workers. Unfortunately, previous workflow languages used in crowdsourcing cannot express usual rules on workers such as allowing to solve a task to only those individuals who contributed in a previous task of a process. Besides, processes in crowdsourcing may change dynamically depending on worker skillsets or deadline. Authors in [60], discuss the need for continuously adapting workflows depending on the context. Another important

[1]http://www.crowdflower.com
[2]http://www.threadless.com
[3]http://www.innocentive.com

concern is compliance with deadlines [73,103]. To the best of our knowledge, and independently confirmed by Kucherbaev et.al. [59], none of the workflow languages for crowdsourcing presented in the literature allows for expressing these usual requirements accurately.

In this section, we define CrowdWON, a modelling language suitable for describing crowdsourcing processes. Our main contributions are:

- We propose a flexible language that allows defining workflows for a large variety of scenarios, including open competitions.
- We propose the first graphical language for crowdsourcing platforms able to express sophisticated restrictions on the workers participating in a task.
- We propose a language that allows to define dynamic workflows that adapt to the status of the process; in particular, it adapts to deadlines and worker profiles.
- We also formalize CrowdWON process models as colored Petri nets, enabling execution, monitoring and simulation in combination with other existing tools tailored for colored Petri nets.

Our modelling language makes it easier to define business processes in crowdsourcing. Thanks to these proposal we can easily visualize and improve the collaboration between individuals in the crowd, computers and companies. Our adaptive flows make it possible to design collaboration patterns that allow to react when the performance of the process is below expectations. Besides, the formalization of CrowdWON as colored Petri nets allows

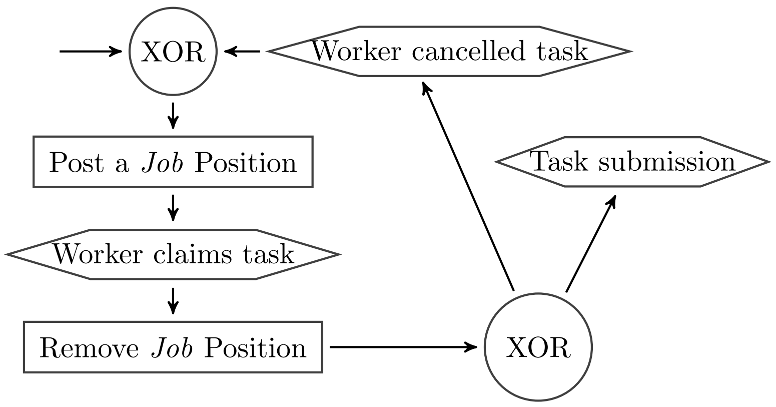


Figure 3.1: Description of the execution of a task in a generic crowdsourcing platform using an EPC diagram.

practitioners to transform visual process models in a language that allows them to execute, monitor and simulate crowdsourced processes.

3.2 Related work

Despite the increasing popularity of crowdsourcing, the number of specialized process-modelling frameworks proposed in the literature is still limited. With TurKit [67], anyone is able to describe the iterations of simple tasks in crowdsourcing. Jabberwocky [3] focuses on the reusability of sequential processes, but its application to iterative processes is not clear. In Turkomatic [60], individuals collaborate in the design of the process and contribute, through refining the textual description of tasks, to its resolution. CrowdWeaver [57] is the first graphical modelling framework for workflow definition and management. With this tool, individuals can design a crowdsourcing process by combining human and predefined computer tasks.

In the aforementioned modelling frameworks, tasks and processes are completed after a known number of individuals contribute to them. However, using these previous proposals, it is not straightforward to represent other scenarios, such as for instance, crowdsourced open competitions, in which the end of the process is defined by a deadline. Deadline management are usually based on simple mechanisms such as notifications to the process manager, so that she can manually adjust the workflow (*e.g.* increasing the financial reward in order to reduce the time-to-completion of tasks [73]). Task routing techniques [41] showed that tasks can be adapted to the profile of crowd workers. However, none of these previous proposals allows expressing automatic transformations of the workflow based on the context.

Petri Nets [82] are a common tool to model processes and workflows. Petri Nets can be defined as directed bipartite graphs, in which nodes represent inactive systems (*places*) or active systems (*transitions*). Information is represented through tokens (pointers to places) and the execution of a process changes the position of tokens. The basic idea of Petri Nets is that transitions transform information from one state (place) to another. Several extensions of Petri Nets can be found in the literature. We highlight three of them related to process modelling. Workflow Nets [99] are Petri Nets with two special places to represent the start and end of the process; dualistic Petri Nets [27], which allow tokens to point to places and transitions in order to represent that a transformation or task is being performed; finally, colored Petri nets [50] allows process designers to specify the structure of the data assigned to token in order to allow the formalization of workflows that adapt to the information contained in the token.

Due to the complexity of describing human collaboration and time management, other modelling languages–*e.g.* *BPMN*[4] and *EPC*[5]–include conditions based on human and timed events. Unfortunately, the use of these event-based frameworks produces complex, and very difficult to understand, diagrams in the crowdsourcing context. Figure 3.1 shows the execution of a single task using the EPC language.

3.3 Crowdsourcing Workflow Net model

In this chapter we propose CrowdWON, a graphical language for describing crowdsourcing processes which is a combination of Workflow Nets and Dualistic Petri Nets. Later in Section 3.5, we formalize CrowdWON as colored Petri nets. Colored Petri nets have enough expression power for formalizing the elements described in CrowdWON. We present both approaches as we envision that a combination of a graphical modelling language and a formal description will increase the impact of CrowdWON. For instance, due to the formalism of colored Petri nets, practitioners will be able to execute, monitor and simulate crowdsourcing processes. On the other hand, colored Petri nets may not be the best formalism for end-users due the complexity of its semantics.

As in Dualistic Petri Nets, a token pointing to a task (or transition) represents an individual performing such task. We define the status of tasks based on the position of the token. Figure 3.2 shows the different status

[4]Business Process Model and Notation. `hwwp://www.bpmn.org`

[5]Event-driven process chain

depending on the token position. Places are implicitly represented in any direct connection between two tasks. Besides, in order to adapt to usual processes in crowdsourcing, we allow tokens to return to a previous place. With this, we can design a *reverse firing rule*: If a worker canceled the task, the token should return to its previous place.

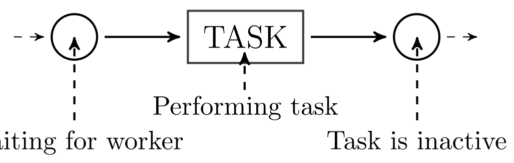


Figure 3.2: Status of the task depending on the token location.

It is important to remark the simplicity of CrowdWON thanks to this representation. As we mentioned before, Figure 3.1 describes the execution of a single task using EPC (an event-driven language). In our proposal, by considering the position of a token, we are able to simplify the representation to only three nodes: Two places connected to a task (see Figure 3.2). A *Job position* is created when a token arrives to the first place, and it will be removed when the token leaves the place. If the worker cancels the task, the token will return to the first place repeating the creation of the job position (describing the loop in Figure 3.1). The second place is only involved when the worker submits the task.

Tasks in CrowdWON are a tuple (*ID*, *typein*, *typeout*, *kernel*) that defines the execution of the task. *ID* is a unique identifier allowing future references of the task; *typein* is a specification of the structure of the input data (*typeout* specifies the returned structure); and *kernel* defines the method used to perform the task. Depending on the type of task, *kernel* may contain different information: Human tasks (represented with a box) require at least a user interface; computer tasks (double-edged box) require a service protocol; a special case of tasks are control flows, used to describe non-sequential processes. In Subsection 3.3.2, we describe the different control flows available in our model.

Tokens in CrowdWON are a tuple (*current*, *data*). The token is pointing to the *current* node, and *data* is the information that is being processed by the *current* node. If *current* is a task, then *data* must be structured as specified in *typein*. That allows individuals to complete the task using the information contained in *data*. After the task is completed, the contribution is stored in *data* as specified in *typeout*. Therefore, it is important that two consecutive tasks have compatible data structures. At the end of the process, *data* is the proposed solution to the problem. Note that this construction only allows one individual to perform the task with the same piece of information. Later in Section 3.3.2, we introduce a mechanism to describe resolution of tasks by multiple workers.

3.3.1 Human Tasks and Worker management

Popular crowdsourcing platforms allow defining restrictions on which individuals can claim the task. Those restrictions are usually related to the ratio of successful submitted tasks, passing some preliminary tests and geo-location. One of the main contributions of CrowdWON is the proposal of a language that allows expressing worker selection constraints in a simple way. These restrictions may be complex: a worker might not be able to take a task if she participated in a particular previous task in the process, workers with a lack of certain skills might only be allowed to solve a task when deadline compliance is at risk, as the last resort, etc. Although these restrictions may be usual in many circumstances, previous languages do not allow to express them. CrowdWON allows for expressing these types of constraints.

Definition 3.1. *Let us call **worker requirement** to a set of constraints on the group of individuals allowed to choose a certain task. Constraints are defined over any property of the profile of individuals (such as ratio of approved contributions, knowledge about a topic, age, gender or location), belonging to a group of individuals or any combination (using conjunctions, disjunctions or negations).*

CrowdWON provides a set of predefined groups of a given task ID to facilitate the design of a worker requirement:

- *ended*(ID), or simply e(ID) is the set of workers who completed the task.
- *revoked*(ID), or simply r(ID), is the set of workers who failed to complete the task or revoked it.
- *accepted*(ID), or simply a(ID), is the set of all workers that at some point claimed the task.

Definition 3.2. *Let us call **worker selection** to an ordered list of tuples* (wr, d)*, where* wr *is a worker requirement and* d *is an optional deadline.*

The list defines a priority on the requirements: Individuals must satisfy the first worker requirement in the list in order to claim the task. When a requirement is an empty set, or the deadline has already been met, the tuple is removed from the list. An empty list puts no restriction on individuals, and so anybody can claim the task.

In Figure 3.3, an example of our worker selection model is depicted. The *Review* task only accepts workers with a success ratio greater than 90%. On the other hand, the selection node has a more complex worker selection: firstly, only workers with a ratio greater than 80% are considered. After 1 day, the worker selection will decrease the threshold to 40%. After 2 days, anyone can claim the task as all deadlines in the worker selection have already passed.

3.3.2 Control Flow Tasks

In CrowdWON, analogously to other modelling languages, the two most basic tasks to control the data flow are AND and OR operators. The AND operator must wait until it receives a token from each of the incoming edges. Then, all these tokens are automatically merged into a single one that remains in the flow of tasks. If the operator has more than one outgoing edge, then a copy of the token is created for every outgoing path. In this case, the *kernel* parameter may specify extra rules for the merging and creation of the tokens (such as splitting a list instead of creating copies). On the other hand, the XOR operator only accepts one token at a time from incoming edges, and only one outgoing path is executed. The operator will look at data contained in the token to decide the outgoing path. Conditions for execution will be written in terms of the received data structure i. The default path will be represented by an empty condition.

3.3.2.1 SELECTION

Authors in [41] show an example of a task that adapts to the expertise of workers: If an individual has knowledge in the American culture, she will answer more questions related to America than other countries. In order to describe these adaptable processes, we extended the XOR operator to include conditions over the profile of workers. The SELECTION node is a control flow task that first finds an individual following the worker selection criteria attached to the node; and then the profile of the selected worker is used to choose an outgoing path.

Figure 3.3 shows an example of a workflow that behaves differently depending on the individuals involved in solving the task. Inexperienced workers (*e.g.* success rate lesser than 90%) can perform the post edition task, reducing the workload of experienced workers.

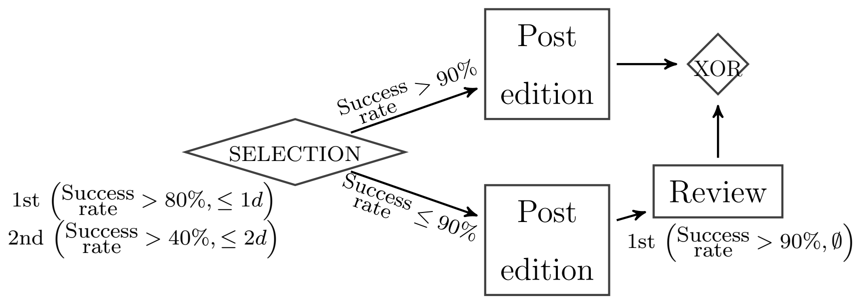

Post edition
XOR
Success rate > 90%
SELECTION
Success rate ≤ 90%
Post edition
Review
1st (Success rate > 80%, ≤ 1d)
2nd (Success rate > 40%, ≤ 2d)
1st (Success rate > 90%, ∅)

Figure 3.3: Example of the Selection operator. If inexperienced workers claim the first task, then an extra review phase will be required.

3.3.2.2 MAPREDUCE STRUCTURE

Collaboration in crowdsourcing processes follows an asynchronous approach: crowd workers are asked to perform tasks individually, and then an aggregation of their contributions is the final output. The MapReduce structure defines the level and mechanisms of collaboration.

Following the PartitionMapReduce approach of CrowdForge [56], we divided our structure in:

- A *sub-process*. The operator contains another crowdsourcing workflow net, defining how data will be processed. We can graphically represent it in the parent process as in Figure 3.4.
- A *generator of tokens*. Every token enters the structure through this generator. It has some rules to create copies (or chunks) of the received data. These tokens are processed independently as described in the sub-process. A list of common generators can be found in Table 3.1
- Finally, we have an *aggregation mechanism* that produces a single output from all the independent contributions. This last part is connected to the generator, so it knows how many tokens are being manipulated by the sub-process. It is also able to request the creation of additional tokens. A list of common aggregation mechanisms can be found in Table 3.2.

Figure 3.4 represents the basic definition of a crowdsourced contest: an *infinity generator* allows individuals to accept the *Submit* task at any moment; then, in order to decide a winner contribution, a review phase is used to score them by computing the average opinion of N workers.

Generator icon	**Description**
N	Create N copies of the token.
$N+$	Create N copies of the token. The generator will create more tokens if requested.
∞	The first task in the sub-process is always available to claim. A deadline specifies when the platform should stop offering the task.
For each	It is used to independently process elements in a list generated by the crowd.

Table 3.1: Examples of generators used in crowdsourcing processes.

As for token management, the token in the parent process points to the structure until the aggregation mechanism returns a value. Generated tokens are executed in parallel instances of the same sub-process. This particularly affects our worker management model. In Section 3.3.1, we introduced three

Aggregation icon	**Description**
MV	Majority vote. Only the most repeated answer will be considered.
$N\%$	Returns solutions repeated by more than $N\%$ individuals. Optionally, it can request more tokens if there is no consensus.
$\emptyset$	There is no aggregation. It returns a list of contributions.

Table 3.2: Examples of aggregation mechanisms used in crowdsourcing processes.

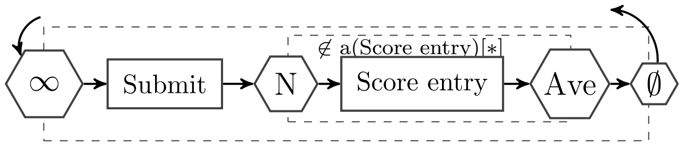


Figure 3.4: Basic description of a crowdsourced competition: An unknown number of submissions are independently reviewed. In the figure, an average of N evaluations is used to rank submissions.

functions on tasks that returns workers involved in their resolution. Due to the parallel execution of the sub-process, these sets only returns workers of the current instance. Nevertheless, it might be useful to also consider workers in parallel executions. See Figure 3.4 for an example. In order to avoid multiple reviews from the same individual, a restriction on workers must consider parallel review tasks.

In order to tackle this issue, we define the operator [$*$] that requests a higher-level point of view. By attaching the operator [$*$] to one of the sets, we are looking at the task from the parent process. So, we can use the condition $workers\ not\ in\ a(\text{Score entry})[*]$ to avoid the duplication of reviewers in Figure 3.4. The operator can be stacked, getting access to a broader point of view: subsequent parent processes will be considered.

3.3.2.3 LOOP STRUCTURE

Even though iterative processes can be modelled by a proper combination of XOR operators, this approach does not allow us to properly limit the number of iterations or measure changes between iterations. Consequently, we introduced a new mechanism to properly define iterative processes. Besides, this will allow us to improve the worker management of tasks inside a loop.

As in the MapReduce Structure, the LOOP Structure contains a sub-process that will be executed as many times as needed until an *exit condition* is satisfied. As an example, Action-Verification units presented in [77] are described using our model in Figure 3.5. An Action-Verification Unit is a pattern used in tasks where human review is required to measure and ensure quality. In this figure, the *Action* task is repeated until the *Verification* task accepts the contribution or the sub-process is executed at least 3 times.

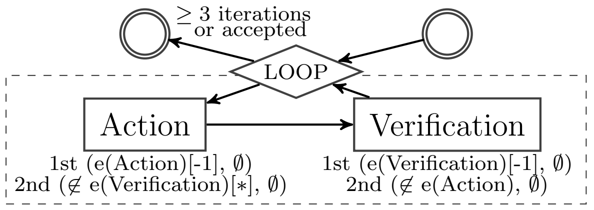


Figure 3.5: In this process, Action and Verification tasks are performed iteratively. Note that nobody can review an action performed by themselves, and reviewers are not allowed to contribute to following actions.

The LOOP structure also inherits the $[*]$ operator from the MapReduce. The absence of the operator represents workers in the current iteration. By using $[*]$, we will consider workers from all iterations. We can restrain the scope by replacing $*$ with a finite set of non-positive integers. 0 represents workers involved in the current iteration; -1 represents workers involved in the previous iteration; etc.

3.3.3 Workflow Transformation

We already described workflows that adapt to the profile of involved individuals. Nevertheless, Turkomatic [60] showed that changes can also come from decisions made by the crowd. And due to external necessities (such as deadlines), process designers may need to adapt the workflow while it is being performed by the crowd. CrowdWON models those unusual changes in the process with a map φ between the places of two workflows. When a transformation is requested (by a computer task or the process administrator), every token pointing to a place p will be now pointing to the place $\varphi(p)$. If a token is pointing to a task t, the platform will wait until it naturally arrives at a place. Since tasks are usually performed by humans, we should not discard their contributions without prior warning.

The description of a generic process in Turkomatic can be found in Figure 3.6. An initial task allows individuals to decide if the subsequent task must be completed at once or needs to be split into smaller tasks. If so, they provide a separation of the task and the *Request map?* task request the workflow transformation φ. Note that workflow (a) is now a sub-process in workflow (b), allowing further subdivisions of the problem.

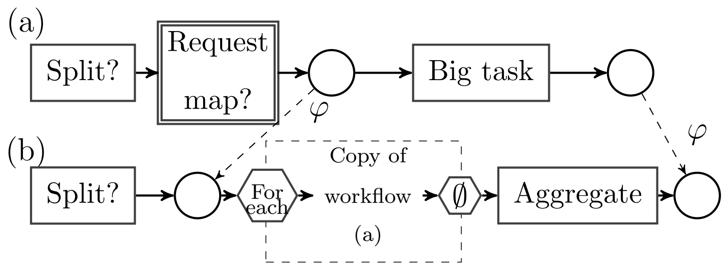


Figure 3.6: As in Turkomatic [60], workers may request to split a task in easier chunks.

3.3.4 Deadline Management

In many different scenarios, the production must be finished before a certain date. Deadline management is the use of any mechanism to ensure the process is completed within a certain time frame. Previous mechanisms in crowdsourcing were based on manual modifications of the process. All these manual modifications can be formalized with the use of our workflow maps. Nevertheless, CrowdWON provides a mechanism to automatize those transformations. Note that we already included time conditions in the selection of workers. In order to increase the capabilities of our deadline management, we introduce the following timed events:

- **Deadline of submission**. Individuals have an amount of time to complete claimed tasks.
- **Deadline of claim**. Individuals have a limited amount of time to claim a task.

The amount of time available can be fixed at the design phase of the process, or it can be computed when a token arrives at a node. Deadlines will be graphically represented with an extra circle, representing a clock. In general, the platform will perform an alternative path (represented with dashed arrows) when the deadline is met. In a more complex scenario, one may additionally trigger a workflow transformation.

An example of deadline management is depicted in Figure 3.7. Individuals have 8 hours to submit a contribution to the *Post edition* or *Review* task. If they spend more than that, the platform will revoke the claim and republish the task (the dashed arrow sends the token to the place before). The only mandatory task is the final edition. For this process we assume there is a overall deadline externally set (i.e. deliver translation before a specific day). For deadlines relative to this external deadline we use the notation $\leq X$, referring to the fact that this deadline will be met if the remaining time is less or equal than X. For example, the review phase of Figure 3.7 will be skipped if the process must end before 8 hours. If the process did not start and we have less than 24 hours to complete the process, then only the final

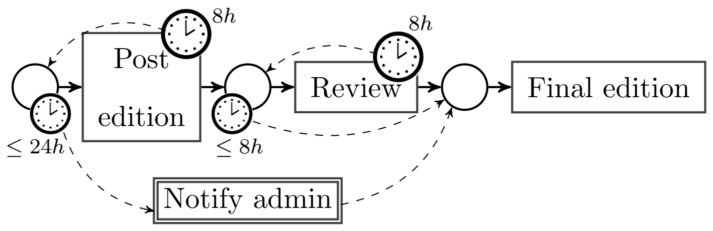


Figure 3.7: Deadline management in a post edition process. Individuals in the post edition and review have 8 hours to submit a task. Tasks may be skipped depending on the remaining time.

edition will be performed and a notification will be sent to the administrator of the process.

3.4 Examples

To show the potential of our model, we are going to describe two complex crowdsourcing processes. We chose one of the first collaboration patterns present in crowdsourcing as an example of marketplace-like tasks. Then, we describe an industrial process that combines the power of machine learning algorithms, in-house analysts and crowd workers.

The FIND-FIX-VERIFY is a pattern designed by Soylent [14] in order to reduce the length of a text. It consists of three phases: In the FIND phase, 10 individuals flag sentences or paragraphs as *potentially reducible*. If more than 20% of individuals send the same flag, then it is considered in the following phases; The FIX and VERIFY phases are executed for every discovered flag in the FIND phase. First, 10 users propose an alternative text with the same meaning. Finally, 5 individuals vote for the most accurate alternative. The original text is replaced by the most voted. We assume there is a time-constraint in the resolution of the process. In order to ensure finishing on time, we enforce the FIX phase to end in at most 2 days. After that period, the number of proposed fixes might be less than 10 or even none. In the latter, we assume that the sentence is already correct or too complex for the available crowd. This process is described in Figure 3.8 using our model.

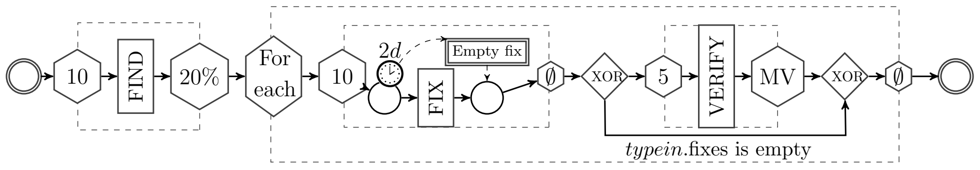


Figure 3.8: Crowdsourcing process used in Soylent [14]. Given a text, the crowd find sentences that can be shortened. The group also proposes 10 shorter versions for every sentence. Consensus is reached by a simple voting system.

Chimera is a crowdsourcing process designed by Walmart Labs [91] to classify a large set of products. Given the size of the set, using hired analysts to label products, or review the quality of machine learning algorithms, is unfeasible. But the crowd can help to reduce the workload of analysts. In Figure 3.9, one can find a graphical representation of the process. The whole process is repeated until all the products are labeled. In the first phase, all products are classified by a combination of machine-learning algorithms and a set of handmade rules. If the algorithm is not sure about the label of a product, it remains unlabeled and passes directly to the next phase. If the algorithm recommends a label, then it is reviewed by the crowd. The total number of reviewers depends on initial consensus: only three crowd workers are involved if the two first reviewers do not agree on the correctness of the classification. After that, some of the products will be examined in a second review. This second phase is done by hired professional analysts: They check the quality of the classification and create new rules for the machine-learning algorithm if needed.

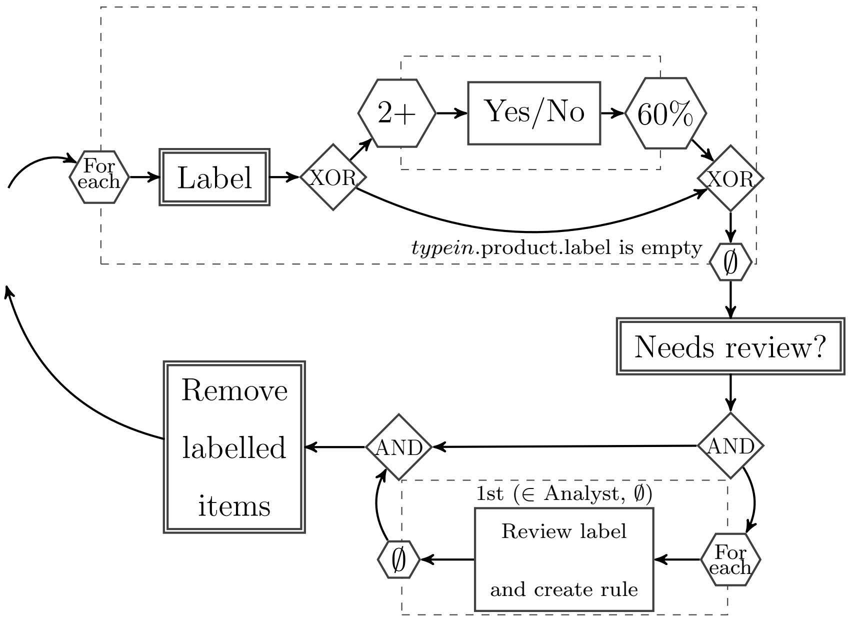


Figure 3.9: Process used by Chimera to label a list of products using machine learning, crowd workers and analysts. In the model, *Analyst* is a fixed group of individuals hired by the company.

3.5 CrowdWON Formalization as Colored Petri nets

Continuing the visual description of a CrowdWON process model, we propose a formalization based on colored Petri Nets. This formalization will enable practitioners to exactly define the behaviour of the crowdsourced process and, in combination with other tools for colored Petri nets (such as *CPN Tools*[6]), to monitor and simulate the execution of the process.

We will propose small additions in order to formalize human interaction, deadline management and subprocess specification. Although those additions seem to be rather small, they require a major refactoring of the original model in order to maintain the level of formalization of colored Petri Nets. Below one can find the formal definition of a CrowdWON process model. An introduction to colored Petri nets can be found in Section 2.2.1.3, and the concepts present in this definition will be introduced in the rest of this Chapter.

Definition 3.3. *A CrowdWON process model is a colored Petri Net ($\sum$, P, T, A, C, G, E, I, WS, D) such that*

- $\sum$ *contains the Workercolor and TimedEvents colors.*
- WS *is a* ***Worker Selection*** *function, mapping a subset of the transitions T to a guard expression defined, at least, over the color type Workercolor, such that every transition in the domain of WS has ex-*

[6]`http://www.cpntools.org`

ID	**type**
Worker ID	*String*
Passed Tasks	*Integer*
Failed Tasks	*Integer*
Pending Tasks	*Integer*

Table 3.3: Definition of Workercolor, a color type modelling the attributes associated to a Worker in a Crowdsourcing Platform.

actly one incoming and one outgoing edge[7].

- *D is a* ***Deadline*** *function, mapping some places and transitions to a guard expression defined over the color type TimedEvents.*

3.5.1 Human Tasks and Worker management

Every transition in a CrowdWON process is expected to perform some modifications on the data associated to the involved tokens. In general, we will assume that these transformations will be performed by humans in a crowdsourcing platform, although some tasks may be automatized by a machine.

Definition 3.4. *Individuals in a crowdsourcing platform, or workers, will be modeled by a token. Such token will be enriched with data following the color type Workercolor, as specified in Table 3.3.*

[7]This implies that we will not allow human tasks that uses information from different tokens. If needed, a normal transition could be used to merge multiple tokens.

Popular crowdsourcing platforms allow defining restrictions on which individuals can claim the task. Those constraints are defined through the worker requirement, which is a guard expression over the Workercolor (see Table 3.3). Notice that Workercolor is the most basic profile of an individual that we may ask the platform to implement. The combination of *Passed Tasks*[8], *Failed Tasks*[9] and *Pending Tasks*[10] provides a rough approximation to the performance of the individual in the platform. Nevertheless, the crowdsourcing platform may include also other properties such as test marks, knowledge about a topic, age, gender or geo-location.

Definition 3.5. *Let us call* ***worker requirement*** *to a guard expression mapping the Workercolor to a Boolean value. An individual satisfying the worker requirement would be eligible to perform the human task.*

Figure 3.10 depicts the visual representation of crowdsourced tasks in a CrowdWON process model, in which a task is modeled with a transition in the underlying Petri Net. The only difference from any other regular transitions is that a worker requirement ws is linked to the transition through the **Worker Selection** function.

The Worker Selection function adds an extra twist to the semantics of colored Petri Nets. Following the notation of Figure 3.10, the color of a token at p_1 must satisfy the guard function g in order to enable the transition. Nevertheless, the Worker Selection function forces the platform to find a

[8]Number of tasks finished by the worker, which solution was accepted by the requester or the crowd.

[9]Number of tasks unfinished or finished by the worker, but with a solution not accepted by the requester or the crowd.

[10]Simply the number of tasks yet to be evaluated by the requester or crowd.

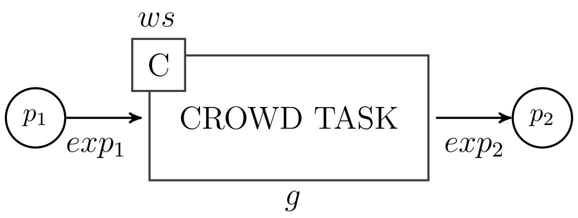


Figure 3.10: Tasks performed by the crowd are modeled by transitions in a colored Petri Net. Functions exp_1 and exp_2 are two arc expressions, and g is a guard function as in colored Petri Nets. Function ws is a worker selection function. Figure 3.11 depicts the behaviour of such construct in terms of a colored Petri net.

human satisfying the expression ws in order to fire the transition. Besides, any human needs some time to accomplish the modelled tasks and, hence, the platform should not allow any other individual to start the task unless the worker rejects the task.

Figure 3.11 depicts the colored Petri Net that models the aforementioned semantics. One may think that, prior to the execution of the process, any human task as in Figure 3.10 would be replaced by this colored Petri Net. Place p_W is a shared place between all crowd tasks, with color type *Workercolor*. At initialization, each worker has a token representing them in p_W. Notice that now the *Assign* transition is enabled (and fired) when a token satisfies the guard function g and a worker token satisfies the guard function ws. If *Assign* is fired, the platform consumes the worker token and the human task may start. Notice that the task cannot be assigned to any other individual unless *Reject* is fired. The *Reject* transition creates again the worker token in

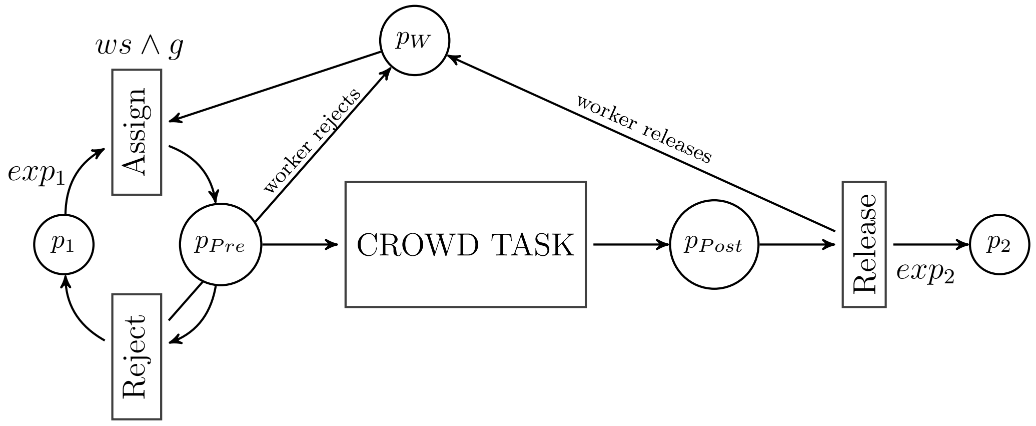


Figure 3.11: Any transition labeled as a crowd task will be replaced with this subprocess. Places 1 and 2 are the two points of connection of this subprocess with the rest of the process.

p_W so she is again available for another crowd task. When the task finished, the token is modified with the information provided by the human worker. And, finally, the worker token is released, allowing her to claim another task.

During the execution of a human task, some modifications on the data associated to the worker will be requested. The following expressions formalize this transformation the worker token.

Definition 3.6. *Given an instance of the Workercolor* $< ID = id,\ Passed = n_p,\ Failed = n_f,\ Pending = n_t >$*, we define*

- *The* ***worker releases*** *expression as the instance* $< ID = id,\ Passed = n_p,\ Failed = n_f,\ Pending = n_t + 1 >$.
- *The* ***worker rejects*** *expression as the instance* $< ID = id,\ Passed =$

n_p, $Failed = n_f + 1$, $Pending = n_t >$.

- *The **requester accepts** expression as the instance $< ID = id$, $Passed = n_p + 1$, $Failed = n_f$, $Pending = n_t - 1 >$.*

- *The **requester rejects** expression as the instance $< ID = id$, $Passed = n_p$, $Failed = n_f + 1$, $Pending = n_t - 1 >$.*

The last two expressions are hidden from Figure 3.11, as they are simply defined as transitions from and to place p_W. Notice that we leave open the possibility of adding extra behavior to pool of worker p_W, as the platform may add extra transitions and places for holding back suspicious individuals (e.g. if an individual has more than 5 *pending tasks* it might be worth to temporarily remove her token from p_W).

The color types of p_{Pre} and p_{Post} are an extension of the color types defined for p_1 and p_2 respectively. We will ask these color types to also hold the information related to the worker involved in performing the crowdsourced task. Apart from the semantic explained in Figure 3.11, we will also take for granted that the color types of all the places are closed with respect to the worker history of the process model.

Definition 3.7. *The closed version of a color type $\sigma \in \sum$ in a CrowdWON process model N is a color type $\overline{\sigma}$ such that*

- *σ is included in $\overline{\sigma}$.*

- *For each human task in N with identifier ID, the color type $\overline{\sigma}$ has the following properties of type 'Sequence of Workercolor':*

  - *ID_Claimed for storing workers who claimed the task ID at some point.*
  - *ID_Rejected for storing workers who participated in the task ID, but it was not finished or the requester flagged the worker contribution as unacceptable.*
  - *ID_Finished for storing workers who successfully finished the task ID.*

The inclusion of these closed colors are the foundation for defining the functions *accepted*, *ended* and *revoked* as defined in Section 3.3.1. This functions allow the process designer to design worker requirements which also consider previous contribution of the worker in the process.

Definition 3.8. *An **extended worker requirement** is an ordered list of elements (wr, c) such that*

- *Function wr is a worker requirement.*
- *c is a condition for discarding the requirements set by wr.*

Extended worker requirements can be used instead of simple worker requirements in order to model worker requirements that evolve over time. Whenever used, the first worker requirement wr in the list should be considered until the condition c is met. In that case, the first element of the list is discarded and the next element is considered as the new worker requirement. In case of of an empty list, all workers are eligible for this task. Conditions defined by c will be typically defined in terms of time, which will be formalized in the next subsection.

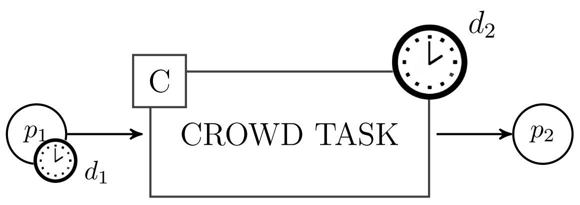


Figure 3.12: Example of a deadline of claim d_1 and deadline of submission d_2. If deadline d_1 is met, the execution continues in place p_2. For deadline d_2 to be considered, a worker must have claimed the task but not finished it yet. If the deadline is met, the platform reassigns the task to another worker. Figure 3.13 depicts the behaviour of such construct in terms of a colored Petri net.

3.5.2 Deadline Management

As already stated in 3.3.4, CrowdWON provides a mechanism to automatize changes in the execution flow based on timed events. In particular, we include basic constructs for defining the following deadlines:

- **Deadline of submission**. Individuals have an amount of time to complete claimed tasks.
- **Deadline of claim**. The amount of time that a task is claimable.

Figure 3.12 depicts a task with both types of deadlines, and Figure 3.13 depicts their default behaviour on the colored Petri net formalization. Notice that deadlines of submission are those deadlines defined over transitions, and deadlines of claim are defined over places.

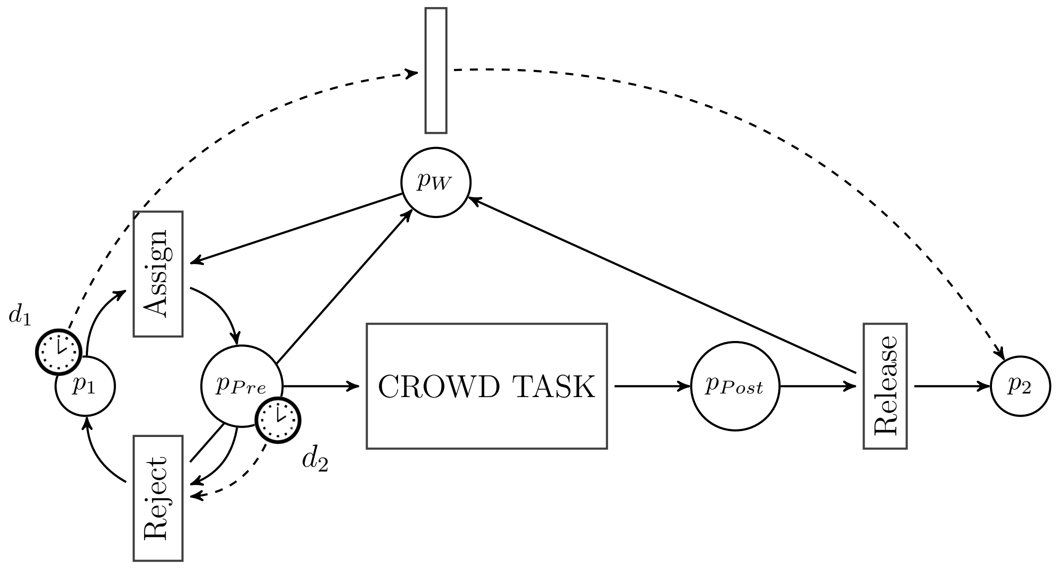


Figure 3.13: Behaviour of deadlines d_1 and d_2 in the task execution. Deadline d_1 may completely skip the task, but only if it is not claimed by anybody. Deadline d_2 on the other hand, controls how much time is allocated per worker to accomplish the task.

In order to formalize time information in the process, we will suppose that there is a shared token that is connected to all transition in the model, which contains information related to the current time of the system. We will denote by *CurrentTime* to the value of such token. The platform will need to update this value regularly (for instance, every second). Besides, we will also assume that all color types have an extra *TimeLastModified* such that will be updated with a copy of *CurrentTime* every time the token is transformed by a transition. We will deliberately skip all this formalization in our figures in order to increase understandability.

Definition 3.9. *The **TimedEvents** color type refers to the combination of the CurrentTime and TimeLastModified properties.*

By combining both values, one could define, for example, conditions over the time spent in performing a task. Notice that the TimedEvents color type could be enriched with other properties external to the execution of the process. For instance, one could also consider to include formal deadlines to customers.

In general, the platform will treat deadlines as those defined in Figures 3.12 and 3.13. Nevertheless, as we have the information about time hard-coded in all tokens, we could devise other possible behaviours. For instance, instead of skipping the task if a deadline of submission is met, one may execute an alternate path. Nevertheless, this behaviour should be hard-encoded into the process model, with the appropriated use of *timed guards* guiding the execution of the alternate path.

3.5.3 Collaborative Processes

Collaboration in CrowdWON is defined with the MapReduce structure, that is formalized as follows:

Definition 3.10. *A* ***MapReduce*** *structure in a CrowdWON process N is a tuple consisting of*

- *a transition t in the original process N such that it only has one input and one output place,*
- *another CrowdWON process N' such that the underlying Petri Net is a workflow net (i.e. there exist a a unique start place, and a unique end place),*
- *a generator, which a strategy for initialization of the subprocess N' as well as governing the creation and coordination of multiple instances of the subprocess N'. Some examples of generators may be found in Table 3.1.*
- *an aggregation mechanism, which is an strategy or method for combining the multiple results obtained through the diverse instances of the subprocess N' created by the generator. The most basic strategy is simply creating an unordered list with all the token information, but other strategies can be found in Table 3.2.*

Sub-process definition is already contemplated in colored Petri Nets, as a transition may be replaced by a process defined in a different colored Petri Net. Some places are labelled with special Input/Output labels, specifying the joint point with the places of the transition in the original process. This is usually done for improving understandability of models, but does not add new expressive power. Everything explained in this subsection could have been done in design time, but these additions make easier the work of the process designer.

In the rest, we will describe 4 usual collaborative processes. These examples will show how we can define a *generator strategy* as a modification on top of the original CrowdWON process N. Prior to start the execution of the CrowdWON process, the platform will perform this transformation. Contrary to the usual subprocess replacement, the choice done in the generator and aggregation mechanism may add extra places and transitions to the process model. For instance, a *Create N copies* generator will create exactly N copies of the subprocess. Notice that the aggregation mechanism could simply formalized with a final transition that processes a list of tokens and, hence, we will focus on explaining the transformations lead by the token generator.

We have decided to use visual templates for defining these common collaborative patterns, as we believe this provides a good balance between formalization and understandability. In the following figures, two places with labels s_1 and s_2, connected with a dashed arrow, will denote a copy of the original subprocess N'. If multiple copies of the subprocess N' appears on the same figure, we will use superscripts to differentiate them. Places p_1 and p_2 are the entry and exit points of the MapReduce structure.

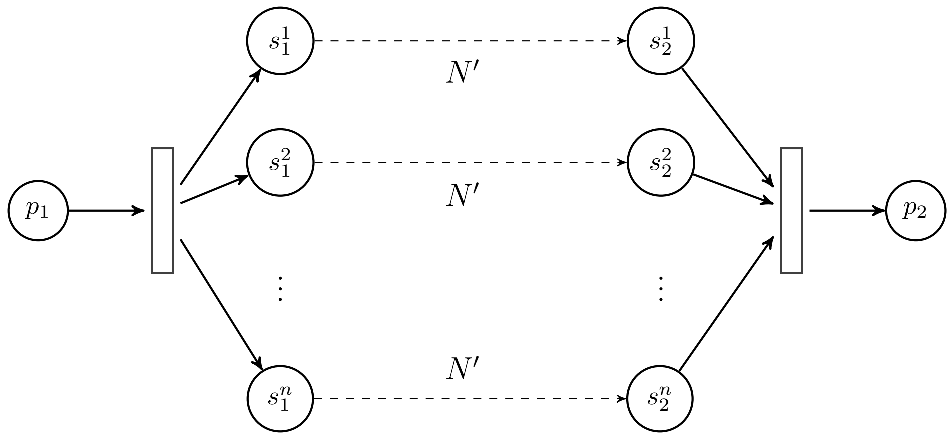


Figure 3.14: This process model is a template for creating N copies of the process model N' connected to places p_1 and p_2. Notice that the number N is fixed in design time and, hence, this transformation is clearly defined in execution time.

Create N copies of the token This collaborative process is the most used strategy in crowdsourced projects. Asking several individuals to perform the same task usually leads, in average, to the right solution. In this simple case, we are asking the process to substitute the MapReduce structure with N copies of the subprocess N' as depicted in Figure 3.14. Notice that N is fixed in design time, so this notation is a simplification for the process designer.

Some crowdsourced projects tweaked the *Create N copies of the token* for enabling the platform to create more copies in case there is not enough consensus between human workers. For modelling such behavior, a combination of the *Create N copies of the token* and the *Iterative process* (explained

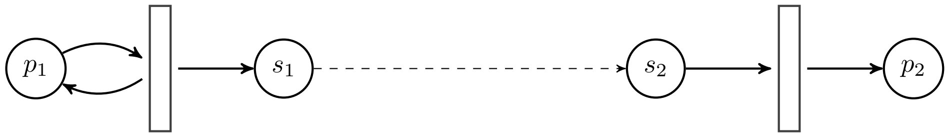


Figure 3.15: Proposed modification for modelling infinite subprocesses. Every time a token is created in s_1, the token in p_1 is not consumed and, hence, the process could have always a token available at s_1.

below) is needed.

Infinite subprocess In this case, we want the process to continuously accepts new entries to the subprocess, a collaborative process interesting for governing crowdsourced contests (as the one depicted in Figure 3.4).

Figure 3.15 depicts the transformation requests to the process model. Each time a token is generated in s_1, the token in p_1 is not consumed, allowing the creation of more tokens in s_1. This transformation could be combined with a deadline, consuming the token at p_1 at some point and, hence, ending the execution of the subprocess. For instance, this could be used to open a combination during a fixed period of time.

For each In several cases, we will need to process information contained in a list. Instead of asking individuals to process the whole list, crowdsourced processes typically reduce the complexity to process elements of the list individually. For doing that, we will ask the subprocess to create as many tokens as elements in the list, and then execute the subprocess for each of them independently.

Figure 3.16 depicts the proposed modification to the process model. We included to intermediate places e_1 and e_2 that are used for storing information as it is computed. In place e_1, the complete list is initially stored and one element is removed from the list every time it is consumed by the transition prior to s_1. This element is stored in a new token in place s_1, that will continue the execution of the subprocess N'. A variable *pending* is also updated in the token of place e_1, indicating the number of elements that have been created in s_1. When the list associated to place e_1 is empty, the token will move to e_2. Now, one token will be consumed to s_2 to be included in the list of token e_2. Notice that every time that a processed element is included in the list, *pending* will be modified in order to let the process know that there is still some token in the subprocess N' pending to be processed. Finally, when all information is processed, *pending* should be zero and the process may continue.

Figure 3.17 depicts the translation to colored Petri nets of the process graphically described in Figure 3.8. This process combines three *Create N copies of the token* with a *For Each* collaboration processes for finding and fixing mistakes in a text. Notice that we expanded the transformations required for these two collaborative processes, but we excluded the worker and deadline management constraints. The figure shows the net modeled using the CPN Tools, which allowed us to simulate the execution of the process and validate that the transformation proposed in Figure 3.16 implements the desired behavior.

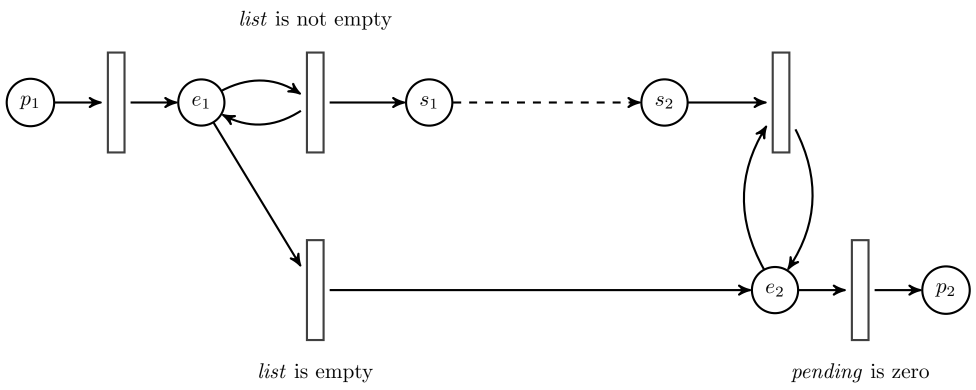


Figure 3.16: Proposed process for processing elements in a list independently. A list of elements is hold in place e_1, and later split in several tokens in s_1. Once the list is empty, it moves to place e_2 and recollects the processed elements stored in s_2.

Iterative processes As in the MapReduce Structure, the LOOP Structure contains a sub-process that will be executed as many times as needed until an *exit condition* is satisfied. Figure 3.18 depicts the transformation needed to model this behaviour in colored Petri Nets. The color in place c starts with a *counter* variable stating the number of times the subprocess has been executed, as well as a list containing all token that goes from s_2 to s_1. All this information could be used to define the exit condition.

Workflow Transformation

Unusual changes in the process continue to be formalized as a map between two process models, but we need to ask to clearly define modifications on

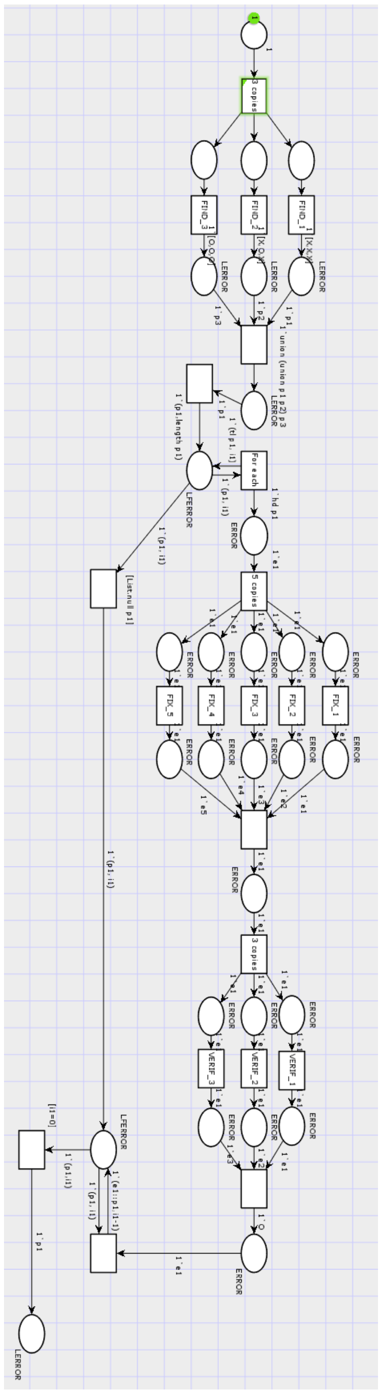


Figure 3.17: Translation to colored Petri nets of the process used by Soylent and specified with CrowdWON in Figure 3.8. Worker and Deadline management transformation have not been performed for ensuring understandability.

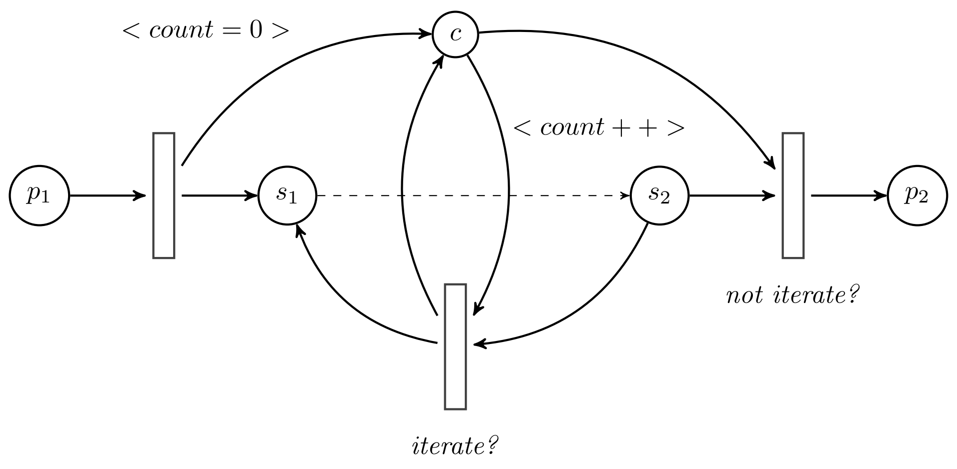


Figure 3.18: Proposed process for modelling iterative processes. A token in place c is created for counting the number of times the subprocess is executed, as well as storing information from previous executions.

data binded to tokens:

Definition 3.11. *A* ***workflow transformation*** *is a map φ between the places of two CrowdWON processes. Notice that the color type of the place p and $\varphi(p)$ may not be compatible, and, hence, some modifications on the color of the token are needed. We will denote the expression that will define the necessary modifications by $\varphi'(p)$.*

When a transformation is requested (by a computer task or the process administrator), every token pointing to a place p will be now pointing to the place $\varphi(p)$. In the case of a token assigned to a place that was not originally in the description of the CrowdWON process model (for instance, a place added for modelling that a worker claimed a task but did not finish it yet), the platform should wait until the token arrives to a place in the domain of φ.

3.6 Implementation of CrowdWON

A collaboration between the research team and the globalization team of CA Technologies lead to a Proof of Concept of an industrial Crowdsourcing platform. Besides, one translator provider reviewed the development of the technology, providing their vision on how translation should be done through crowdsourcing, and evaluating the usefulness of the tool for their company.

In this Proof of Concept, the CrowdWON language was partially implemented in order to let requesters define their own processes and monitor its execution. We did not include a visual designer for CrowdWON processes and, hence, requesters had to manually modify a configuration file for defining their processes. This configuration file was a BPMN file that includes specific elements tailored to support the data type definitions, the MapReduce structure and deadline assurance mechanisms specifics to CrowdWON.

Figure 3.19 depicts the translation process that showcased the Proof of Concept in which CrowdWON was implemented. The translation process starts with the top process model which, after asking the requester the size of the translation tasks and the target languages, calls the middle process model for each target language. We used the sub-process notation for representing the MapReduce structures. Then, an automated script divides the original text in several chunks as accorded by the requester. These chunks are then translated without any verification step. A screenshot of such translation

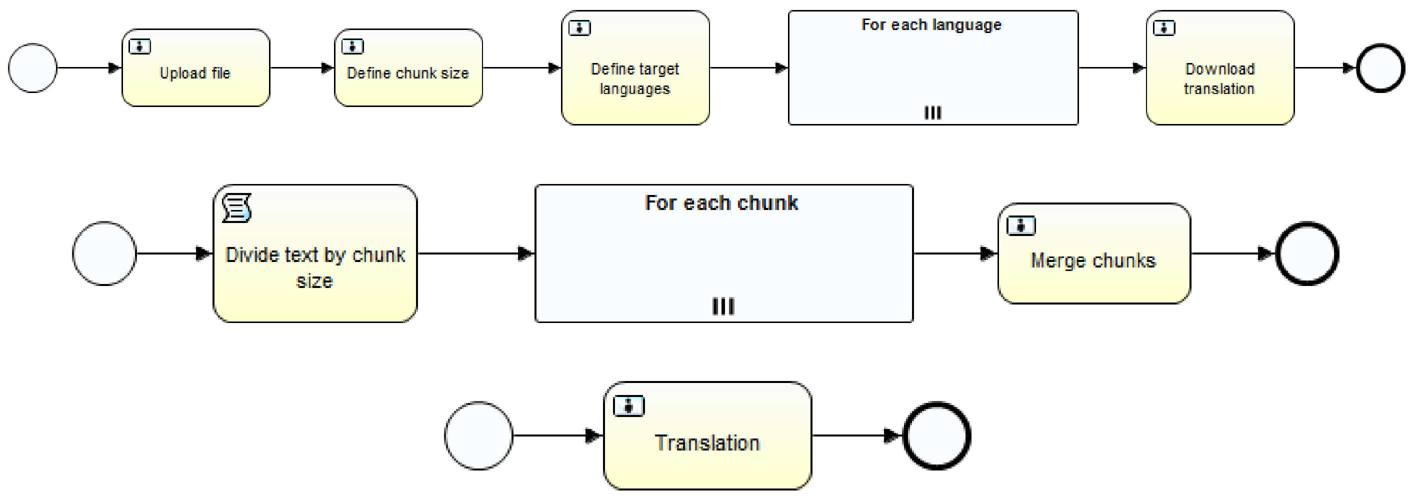


Figure 3.19: Translation process executed during the Proof of Concept in which CrowdWON was implemented. The translation process starts with the top process model which, after asking the requester to set the target languages, calls the middle process model for each language. The original text is divided in several chunks that are executed following the bottom model.

activity is included in Figure 3.21. One of the benefits of using BPMN is that there are several process modelers in the market. Nevertheless, the specifics of CrowdWON are hindered in the underlying XML specification. We are aware that this approach is not suitable for non-technical individuals, but it was sufficient for the scope of this Proof of Concept. Figure 3.20 depicts the underlying procedure for executing one task in a crowdsourcing platform and highlights the benefit of having a modelling solution tailored to this scenario.

Although not having a visual editor, we include a flatten version of the process as a mechanism for visualizing the current status of its execution. Figure 3.23 depicts the flatten visualization of the execution of a translation process. Nested boxes represent subprocesses triggered by a MapReduce structure, whilst parallel boxes shows the concurrency of both subprocesses. Activities that did not start are not depicted in the flatten version of the process model, as a workflow transformation may change the course of the process. In fact, the third activity of the process, *Define target languages*, decides how many subprocesses will be executed next. In the example, the requester choose *Spanish (Spain)* and *Spanish (Mexico)* and, hence, two concurrent subprocesses started afterwards.

Figure 3.22 depicts the final usage and relation of both graphical version of CrowdWON and its formalization using colored Petri Nets. The high-level, graphical version of CrowdWON is used as the interface with the end user. The graphical representation is translated by the platform to a formal process model based on colored Petri nets, which governs the execution of the crowdsourced process. Notice that the connection between the two versions is bilateral, as information contained in the tokens of the low-level version

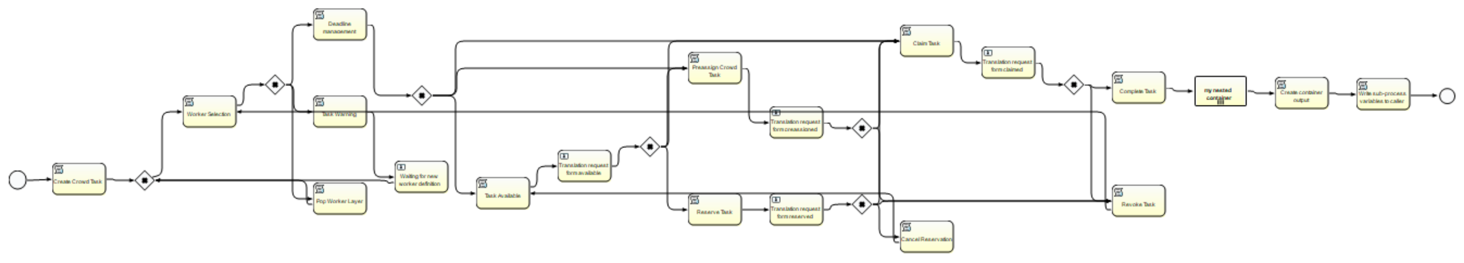
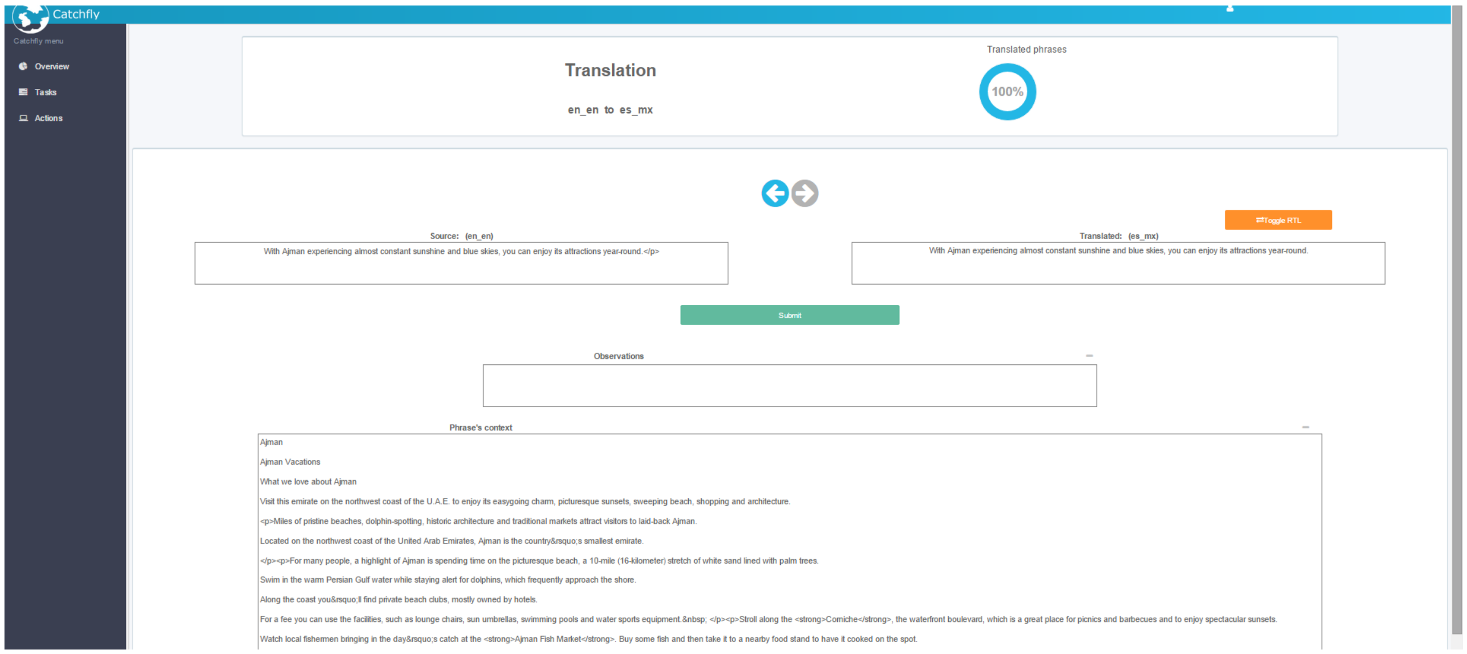


Figure 3.20: A glimpse of the underlying process that explain how a crowdsourced task is managed by the platform, including communication with the crowdsourcing platform, deadline management and task assignment.



Source: (en_en)

With Ajman experiencing almost constant sunshine and blue skies, you can enjoy its attractions year-round.</p>

Translated: (es_mx)

With Ajman experiencing almost constant sunshine and blue skies, you can enjoy its attractions year-round.

Observations

Phrase's context

Ajman

Ajman Vacations

What we love about Ajman

Visit this emirate on the northwest coast of the U.A.E. to enjoy its easygoing charm, picturesque sunsets, sweeping beach, shopping and architecture.

<p>Miles of pristine beaches, dolphin-spotting, historic architecture and traditional markets attract visitors to laid-back Ajman.

Located on the northwest coast of the United Arab Emirates, Ajman is the country&rsquo;s smallest emirate.

</p><p>For many people, a highlight of Ajman is spending time on the picturesque beach, a 10-mile (16-kilometer) stretch of white sand lined with palm trees.

Swim in the warm Persian Gulf water while staying alert for dolphins, which frequently approach the shore.

Along the coast you&rsquo;ll find private beach clubs, mostly owned by hotels.

For a fee you can use the facilities, such as lounge chairs, sun umbrellas, swimming pools and water sports equipment.&nbsp; </p><p>Stroll along the <strong>Corniche</strong>, the waterfront boulevard, which is a great place for picnics and barbecues and to enjoy spectacular sunsets.

Watch local fishermen bringing in the day&rsquo;s catch at the <strong>Ajman Fish Market</strong>. Buy some fish and then take it to a nearby food stand to have it cooked on the spot.

Figure 3.21: Screenshot of the translation task in the process of Figure 3.19.

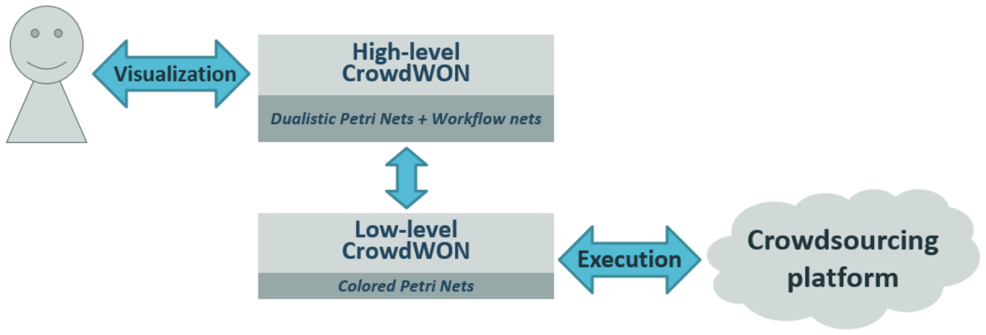


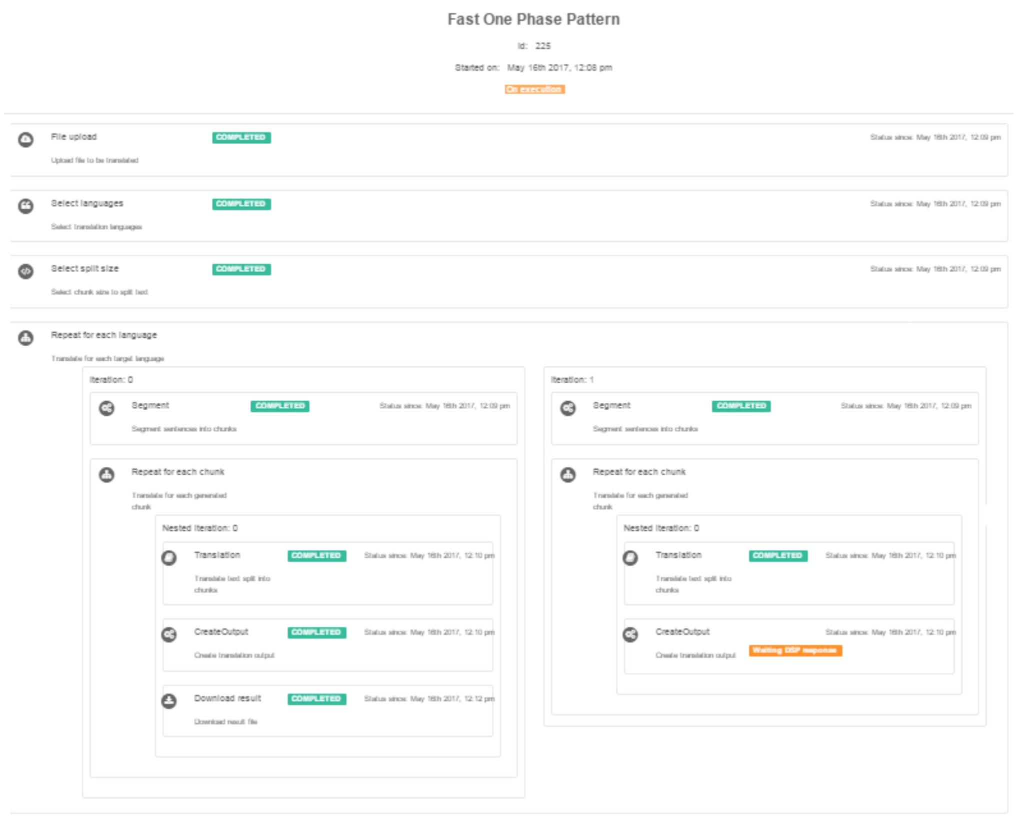
Figure 3.22: Diagram describing the relation between the high-level language (as in Section 3.3) and its low-level version (as in Section 3.5).

can be translated to the high-level version in order to give feedback to the end-user about the execution of the process.

The overall evaluation of CrowdWON by both the translation provider and the globalization department was positive, with a special interest in the definition of worker requirements. In particular, they acknowledged the value of the translation process later defined in Figure 5.2 and that this process was not possible to execute within current crowdsourcing platforms as:

- pre-assignment of tasks to particular individuals has been overlooked but poses several advantages as stated in Chapter 5.
- there is a mechanism for relaxing the process and worker requirements in order to satisfy deadlines.

For deadline management, we had to be less generic than in the original definition of CrowdWON. The original level of generalization was perceived as a bit confusing by newcomers to the language. In particular, we only allowed to



Fast One Phase Pattern

Id: 225

Started on: May 16th 2017, 12:08 pm

On execution

File upload COMPLETED Status since: May 16th 2017, 12:09 pm
Upload file to be translated

Select languages COMPLETED Status since: May 16th 2017, 12:09 pm
Select translation languages

Select split size COMPLETED Status since: May 16th 2017, 12:09 pm
Select chunk size to split text

Repeat for each language
Translate for each target language

Iteration: 0

Segment COMPLETED Status since: May 16th 2017, 12:09 pm
Segment sentences into chunks

Repeat for each chunk
Translate for each generated chunk

Nested Iteration: 0

Translation COMPLETED Status since: May 16th 2017, 12:10 pm
Translate text split into chunks

CreateOutput COMPLETED Status since: May 16th 2017, 12:10 pm
Create translation output

Download result COMPLETED Status since: May 16th 2017, 12:12 pm
Download result file

Iteration: 1

Segment COMPLETED Status since: May 16th 2017, 12:09 pm
Segment sentences into chunks

Repeat for each chunk
Translate for each generated chunk

Nested Iteration: 0

Translation COMPLETED Status since: May 16th 2017, 12:10 pm
Translate text split into chunks

CreateOutput Status since: May 16th 2017, 12:10 pm
Create translation output Waiting DSP response

Figure 3.23: Flatten representation of a translation process for visualizing the execution of the process. Finished tasks are depicted with a green label, whilst on progress or failed activities have an orange label.

define two types of deadlines: skipping a sequence of tasks if not enough time to execute the whole process; and the automatic resignation of crowdsourced tasks if the worker does not reply on time. Besides, we were not able to assess the utility of the workflow transformations as the translator provider was not interested in having radical changes on their translation processes: a mixture of deadline management and adaptive worker requirements satisfies their current needs.

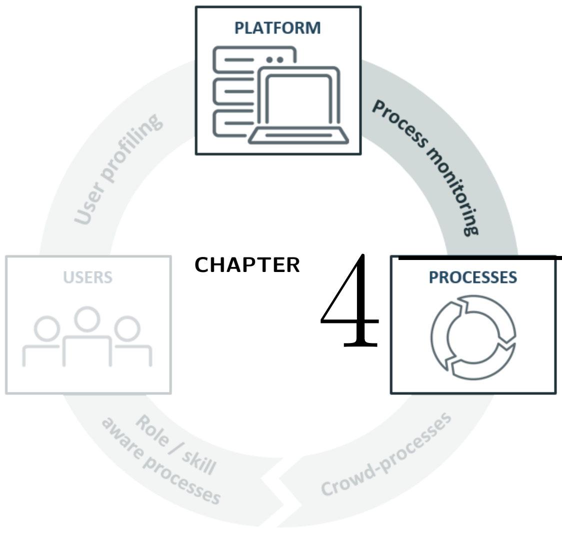
3.7 Discussion

The research on crowdsourcing has been mostly focused on study individual tasks properties, and the application of crowdsourcing to the resolution of real complex problems has been limited to a few examples. However, we have observed a real industrial interest in crowdsourcing for solving complex real world processes. To cover this industrial need, the design of tasks takes a global perspective: deadline management does not only affect particular tasks, but the whole process; Individuals can participate in multiple tasks of the same process, but it may be need to limit those contributions, etc. CrowdWON offers to the industry a graphical language able to represent a possible workflow for complex industrial processes.

As future work, we want to further investigate the best procedure to describe reward mechanisms in our model. At the moment, it would be easy to implement usual financial incentives with a combination of task parameters, computer-executed tasks and the usage of the *ended* function, but we still need to study how our workflow transformations can affect rewards.

It still remains to study if our modelling language can also help in the design of mechanisms to measure the skill acquisition of crowd workers. Our new contribution-based policy in the selection of workers allows for communication between individuals contributing to the process. Therefore, it is possible to give feedback to individuals, so they can improve their contribution. Moreover, this also sets a first step to group-based crowdsourcing. Groups could claim complex tasks, collaboratively create sub-processes to solve the problem and assign sub-tasks to their colleagues or the crowd–if the team does not have expertise in an area.

Another interesting research area is to combine a granulated classification of crowd tasks and process querying in order to recommend similar processes. In combination with a simulation tool, a practitioner may then *what if* scenarios by estimating changes in quality or mean time-to-completion if the proposed modification is performed. For instance, the FIND-FIX-VERIFY pattern has shown more potential in producing higher quality outcomes, and other processes may benefit from such boost on quality by replacing particular activities of their processes by this pattern.



CHAPTER 4

Reducing Event Variability for Improving Process Discovery

During the last Chapter, we described a graphical modelling language for crowdsourced processes. Having those processes is fundamental for supporting optimization and monitoring of the task resolution. Unfortunately, it is often that processes are not detailed enough or defined prior to its resolution. In those cases, a human supervisor needs to analyze and understand the factual work done in each step of the process.

In this chapter, we work towards automatizing the laborious task of monitoring non-defined processes. Assuming that all the steps recorded by the platform have some textual description of the work done, we propose to use novel comparison tools for generating groups of similar activities and, hence, enabling later analytics and insights, such as a process discovery for understanding, monitoring, or formalize the underlying crowd-process.

4.1 Introduction

As Microsoft recently highlighted[1], the industry is moving toward bots that automatically answers simple orders or questions. This recent shift towards conversational bots is primarily thanks to recent developments on Natural Language Processing, which made possible to analyze messages, answer appropriately and, in some simple and predesigned cases, automatize an action. Nevertheless, human experts are still needed to validate complex decisions. But an essential piece is missing for a deeper implementation of conversational bots on the industry: a lack of monitoring tools for conversations.

The lack of monitoring tools for conversations is not only hindering the implementation of fruitful, bot-based conversations, but several business-to-business (or business-to-consumer) services are still provided as a human-to-human conversation that could benefit from such tools. For instance it is known that improving the efficiency of customer support channel lead to low customer churn rates and, hence, to run a more efficient business.

Process modelling has the potential for helping the industry move towards conversational bots, as data-aware process models are a good candidate for modelling conversations in which events represent messages interchanged. Besides, business process modelling languages would be a good visualization for human experts to monitor behaviour of the bot and, eventually, validate automatic actions based on the flow of a conversation in the process model.

One of the most frequent assumptions in the literature of business pro-

[1]`http://blogs.microsoft.com/blog/2016/08/03/progress-in-the-shift-to-conversational-computing/`

cess modelling and process mining is that events are well defined (i.e. the process execution cannot execute events outside a fixed list of events) and all variability of an activity is abstracted as attributes of the event, that do not usually affect the discovery of the process. Nevertheless, this assumption is no longer acceptable on the aforementioned context, in which events are manually defined by humans and, hence, variability is expected.

In this chapter, we investigate the problem of event name variability for process discovery and propose an approach to resolve this problem thorough event log pre-processing. In particular, we introduce an approach for clustering event names based on novel similarity metrics between textual data [75] and, afterwards, create a new refined log by projecting events to the discovered clusters. In Section 4.3, we describe the problem and a general overview of a solution. Then, in Section 4.4, we explain the details of our solution which is later validated in a simple use case and in an industrial scenario during Section 4.5.

4.2 Related work

Different approaches exist in the literature for the problem of discovery and management of process models with a large set of supported activities. For instance, the Fuzzy Miner [102] allows practitioners to choose a level of abstraction for the discovered process model, and the algorithm automatically merges different events into a single, and more abstract, cluster of events. Similar approaches [15,39,40,94] find groups of correlated events and substitute them for the discovery of a more abstract process model. Nevertheless, these approaches are either based on the *directly-follows* relation [39, 102], a temporal correlation [40] or satisfying a particular known pattern [94], or initially unknown patterns [15]. Our approach is not based on the fact that events may have a sequential relation between them or follow a predefined pattern, but that the event name similarity may indicate how similar are two events.

Bag-of-words techniques, i.e. using the frequency of each word in a document, have been largely used for comparing two texts and discover a list of topics in a set of documents. Nevertheless, we have not found any attempt on using these techniques for measuring the similarity of two activities with the purpose of simplifying the complexity of an event log. In fact, the most similar approach we have found in the literature is [10], in which authors consider activity names, and their descriptions, to map events to activities of a known process model.

4.3 Log pre-processing via Event Variability Reduction

In this section, we introduce a new event log pre-processing based on discovering abstract events that compromises a plurality of the original events. Its main objective is to decrease the ratio of distinct events per trace and increase the support of events (i.e. the number of traces in which an event appears). Although performing such pre-processing will reduce the information from individual events, it will enable practitioners to compare different traces and, hence, perform exploratory analysis such as trace clustering and

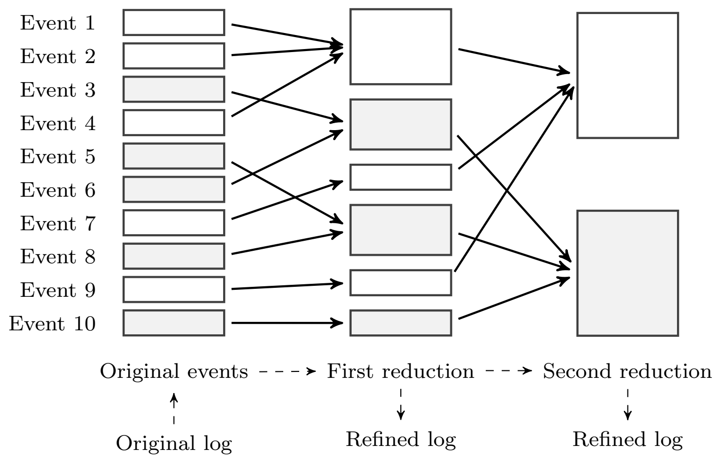


Figure 4.1: Graphical representation of two executions of an event variability reduction method over 10 fictional events. The color of the box depicts the final abstract event.

process discovery. In Section 4.5, we will see an example of a datset that would have been impossible to explore without the use of the Event Variability Reduction technique explained in this Chapter.

Definition 4.1. *Given a log L, a set of events E and a partition of the event set $\{E_i\}_{i \in I}$, then the **Event Variability Reduction** is the event log*

$$L' = \{\langle i_0, i_1, \ldots, i_n \rangle \ / \ e_j \in E_{i_j}\ , \ \ \forall \langle e_0, e_1, \ldots, e_n \rangle \in L\}$$

Generally speaking, the Event Variability Reduction replaces all events in the log L by the unique identifier of the partition in which they belong. Figure 4.1 depicts a graphical example of the application of Event Variability Reduction techniques to an Event log L, which is compromised by 10 distinct events. A first run of the Event Variability Reduction technique reduces the number of distinct events to 6, and generated a refined log by projecting the original events to the new event space as specified by the arrows. In this example, the refined log will start with two executions of an abstract activity $1'$ instead of the original sequence of Events 1 and 2. A second run of the Event Variability Reduction method simplifies even more the event space to only 2 distinct events.

Notice that the Event Variability Reduction is different from other approaches for reducing granularity of events already considered in the literature (see, for instance, [94]). Contrary to these approaches in which the presence of several events in the same trace indicate the execution of a higher-level action, we are removing information from the event space in order to enable the comparison of events that were initially different. For instance, Events 1 and 2 of Figure 4.1 were completely different in the Original Log, but they are considered the same event after the first run of the Event Variability Reduction.

Figure 4.2 depicts an example of how the Event Variability Reduction can be stacked with other BPM tools. In this particular example, which utilizes an event log later used in the evaluation, a process discovery technique is used in combination with the Event Variability Reduction. The high-variability of the original log produced a process model that made impossible understandability of the process model. The evaluation done in Section 4.5 is primarily based on the flow described in this Figure.

Figure 4.3 depicts a more ambitious example with the aim of comparing the behaviour of two users in a digital platform. The activities they perform in the platform are simplified using an *Event Variability Reduction* technique,

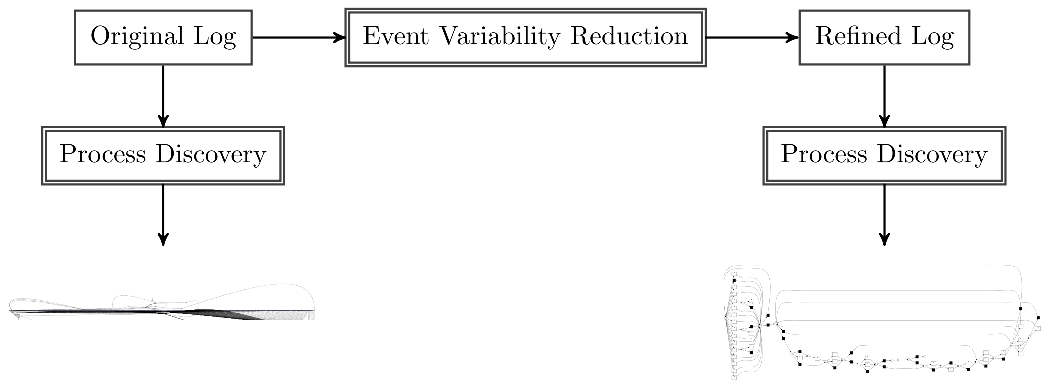


Figure 4.2: Example of how the event variability reduction can be stacked in the usual BPM toolchain as an event log pre-processing tool.

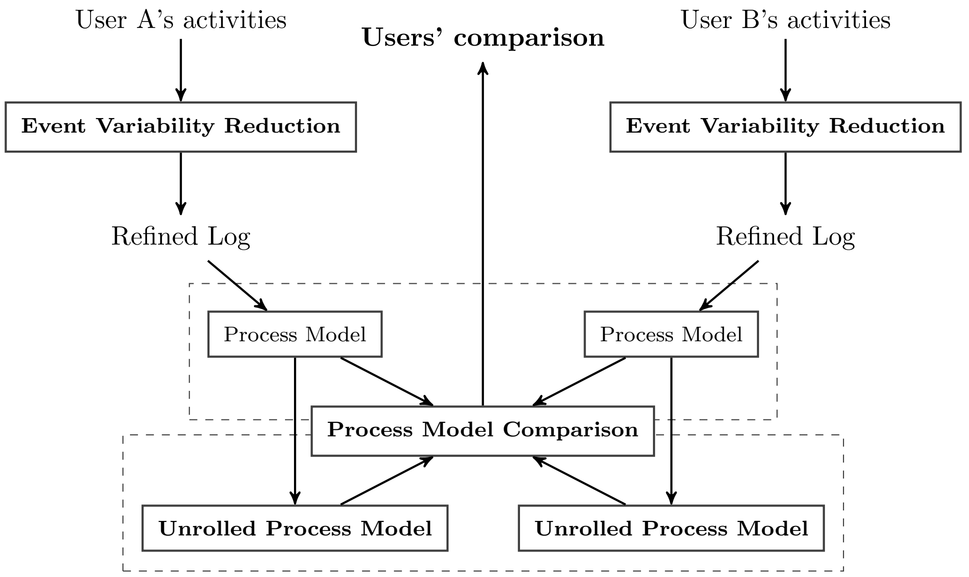


Figure 4.3: Diagram summarizing how 3 of the contributions of this thesis can be stacked for comparing the behaviour of two users in a digital platform.

that will be used for discovering a set of process models that summarize the behaviour of the users. The two discovered process models can be fetched into a *Process Model Comparison* technique for measuring a similarity score between the underlying users. The technique later explained in Chapter 5 may be used for this objective. As we are focusing on modelling the behavior specific to a user, we may want to retrieve a more precise process model. For instance, one could use the *process unrolling* technique later explained in Chapter 6.

The utility of the Event Variability Reduction is based on the quality of the event set partition that we use. In general, it is expected that if two events e_1 and e_2 are projected into the same event e, then both events have a property in common that do not necessarily share with events not projected to e. In the rest of this chapter, one can assume that event names are messages interchanged by a customer and a conversational bot, and, hence, we will assume that the event name is good representative of the action taken.

4.4 Approach

In this section, we propose a particular approach for realizing the Event Variability Reduction. Assuming that the name of the events is a good representative of the action taken, we propose to group together those events that have event names with a similar meaning. We will use a novel technique, summarized in Section 4.4.1, for measuring the similarity of two event names. Contrary to traditional bag of words techniques, such similarity compares the semantics of words and sentences.

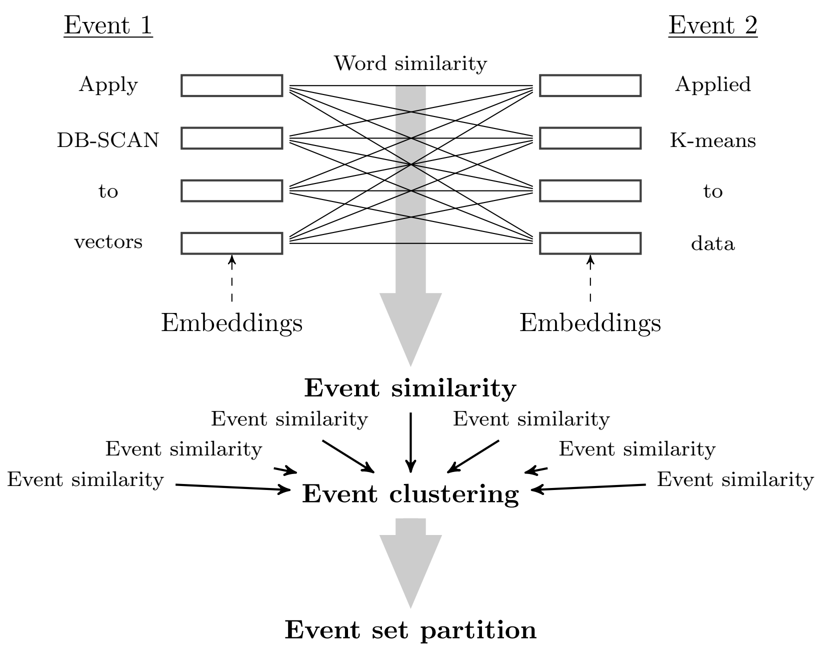


Figure 4.4: Overview of the Event Variability Reduction based on word embeddings. A clustering technique utilizes an embedding-based text similarity for creating groups of events.

Figure 4.4 summarizes our methodology. First, we retrieve all event names from the log and we compute the embeddings of all the words contained in the event name, as explained in Section 4.4.1. These word embeddings allow us to compute a word similarity, that then is later combined for measuring a similarity metric between event names such that events with a similar semantic are very similar according to this metric. Finally, we consider a clustering technique for discovering group of events and we use this as the event set partition of the embedding-based Event Variability Reduction.

We expect this approach to work on the *conversational bot* scenario because all the information of the action taken is encoded in the messages interchanged between the customer and the conversational bot. Nevertheless, this approach can be used in other scenarios. For instance, if the event name does not hold enough information but a textual description for each event is provided, then we can use the same approach with the textual description.

4.4.1 Word Embeddings

An embedding is a map that generates representations of objects difficult to analyze into a well-known space, such as a vector space, allowing further analysis. Notice the resemblance of embedding to hash functions. The later are used to create a representation that is useful for deciding if two complex object are equal with some uncertainty, although unlikely. On the contrary, embeddings have extra properties that allows for studying the original object based on its representation.

Some advances on embeddings for natural language processing have been possible thanks to the recent developments on neural networks. For instance, Word2Vec [75] is a word embedding such that the vector representation of words with similar meaning are, indeed, close in their vector space. The major benefit of using Word2Vec is that the training method is unsupervised: No need for building complex taxonomies, we just need to feed lots of real sentences and the word embedding learns the *meaning* of the word by comparing adjacent words. Moreover, performance with respect to other traditional and unsupervised count-based techniques [11] positions Word2Vec as the perfect candidate for measuring similarity of textual data.

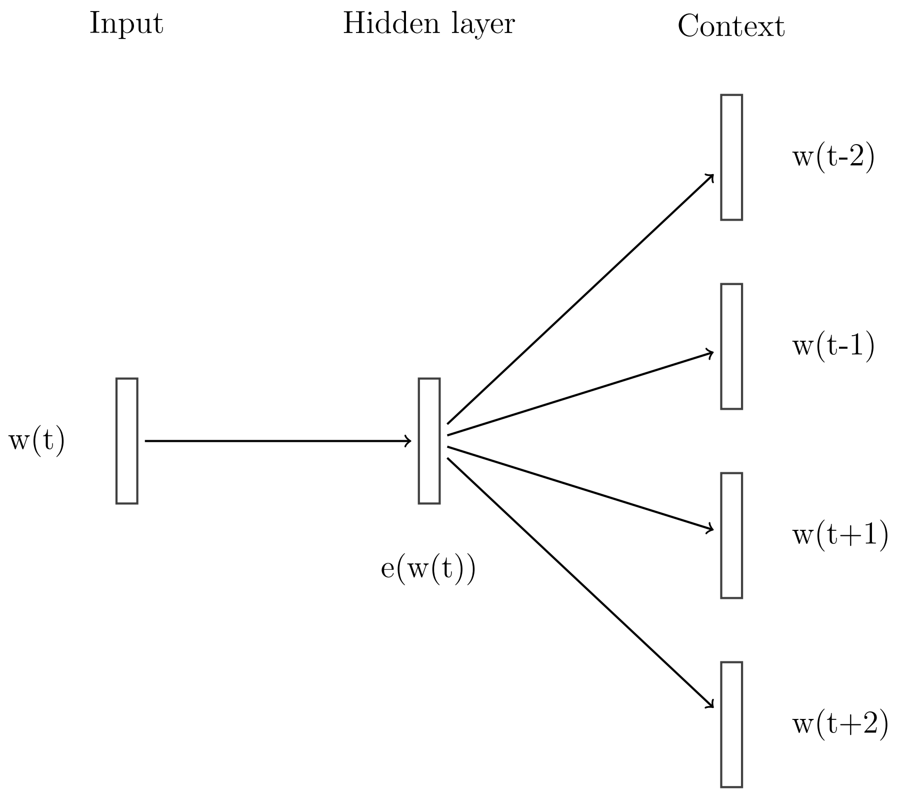


Figure 4.5: Example of the Skip-Gram architecture of the neural network used to train the Word2Vec model [75]. The neural network is designed to predict the 2-window context of the word $w(t)$. I.e. given the word $w(t)$, it computes the embedding $e(w(t))$ that can best approximate the two previous, and two future, words found in training examples.

Technically, the Word2Vec model consists of a fully connected feedforward neural network with 1 word as input, 1 hidden layer and as many word outputs as the size of the context we are considering. Figure 4.5 depicts an example of a neural network used to compute the Word2Vec embedding. The objective of such neural network is to use a word w for predicting its most probable context, i.e. words that are next to w in any sentence.

To train such a neural network, we split the example sentences on buckets of words. For instance, `The quick brown fox jumps over the lazy dog` would be splitted in buckets `The quick (brown) fox jumps`[2], `quick brown (fox) jumps over`, `brown fox (jumps) over the`, and `fox jumps (over) the lazy`, `jumps over (the) lazy dog`. Once the neural network is trained, the embedding vector of a word w is the values of the hidden layer once the neural network is feed with word w. In [11], authors tested different sizes for the hidden layer over tests related to semantic similarity, synonyms, concept classification, and then published[3] their results alongside the embedding to be reused for research purposes. Our results are based on this pre-trained model.

Albeit the simplicity of such definition, one must realize that first we need to do find a one-to-one mapping from words to numerical vectors, so we can use this numerical vector as input for the feed-forward neural network. The simplest choice is to use a one-hot encoder, i.e. the i-th word in the dictionary is represented by a zero vector with an 1 in the i-th position. By using the one-hot-encoder, the size of *words* in the neural network are as big as the

[2]Words inside parenthesis depict the input word of Figure 4.5, which is used to predict the surrounding words.

[3]http://clic.cimec.unitn.it/composes/semantic-vectors.html

number of words in our dictionary. Nevertheless, one can choose the size of the hidden layer, allowing us to have much shorter word representations.

Even though Word2Vec is a perfect candidate for embedding words in a vector space, we are interested in comparing small sentences. There are some variants of Word2Vec in the literature that allows for embedding paragraph or variable-length texts [62]. Nevertheless, such techniques are usually not suitable for short-length texts. In [54], authors introduced a technique based on Word2Vec to compute a similarity between short sentences that shows better performance. Their approach measures the pairwise similarity between words of the two sentences, averaging by the inverse document frequency[4] of words, as follows

$$\text{Similarity}(S_1, S_2) = \sum_{w_1 \in S_1} \text{idf}(w_1) \cdot \frac{(c+1) \cdot \max_{w_2 \in S_2} \text{ Sim } (w_1, w_2)}{\max_{w_2 \in S_2} \text{ Sim } (w_1, w_2) + c \cdot \left(1 + b - b \cdot \frac{|S_2|}{\text{average length}}\right)},$$

where S_1, S_2 are two sentences (S_1 is assumed to be larger than S_2), w_i is a word of the sentence S_i, $\text{Sim}(w_1, w_2)$ is the cosine similarity between the Word2Vec embeddings of w_1 and w_2 and b, c are two regularization constants[5].

[4]We follow the classical approach $idf(w) = \log \frac{\text{Number of documents}}{\text{Occurrences of } w}$, even though other definitions of the inverse document frequency may be used.

[5]We set c to 1.2 and b to 0.75 as proposed by [54]

4.4.2 Event Rediscovery via Document Embedding Clustering

Document Embeddings transform words, paragraphs and/or documents into a point in a vector space. Some embeddings such as Word2Vec transform words with similar semantics into points closer in the space, and hence grouping those points produce groups of similar events. What we propose is to use a clustering technique over event names for discovering groups of similar events to be generalized by an Event Variability Reduction.

Definition 4.2. *Given a log L, with a set of distinct events E, we define an **Embedding-based Event Variability Reduction** as the Event Variability Reduction defined over the partition set obtained after running a clustering technique on E and using the embedding-based similarity metric defined in 4.4.1.*

After applying an embedding-based Event Variability Reduction, the newly discovered event log has reduced the uniqueness of event names by discarding the wording used and, instead, focusing on the semantic of the words used. Notice that, depending on the clustering technique used, one could obtain event logs with different levels of granularity. Besides, by reducing the variability of events and increasing the support of the events, we leverage the utility of other data exploration techniques. In particular, a process discovery technique could be used to have a general overview of the process.

4.5 Evaluation

For evaluating the approach presented in this Chapter, we decided to consider first a dataset in which we know that *event names* hold information about an unknown activity, there are some guidelines on how those *events* should be named and positioned in the trace (i.e. the system process) and results can be easily interpreted. With such dataset, we will perform a preliminary process analysis that would have been impossible without the use of the Event Variability Reduction explained in this Chapter. Afterwards, we apply the techniques to an industrial dataset compromising several textual conversations between technical support engineers and customers. Prior to the technique described in the Chapter, the lack of structure in textual messages did not allow for a mechanism for monitoring the evolution of conversations.

4.5.1 Structure of Documents in Wikipedia

In the context of decentralized human collaboration, some projects rely on guidelines to palliate the variability of human outcomes and communities inside the crowd are created, if its size allows it, to ensure better coherence of the solutions. Wikipedia is a great example of such a complex project, with over 200 guidelines[6], ranging from behavioral recommendations on a discussion to naming rules, and almost 1500 portals[7] helping editors of articles on a specific topic to have a similar style and share knowledge between them.

[6]`https://en.wikipedia.org/wiki/Wikipedia:List_of_guidelines`

[7]`https://en.wikipedia.org/wiki/Portal:Contents/Portals`

We retrieved over 800 articles from Wikipedia[8], selected from the list of featured articles[9] of the *Media*, *Literature and Theater*, *Music biographies*, *Media biographies*, *History biographies* and *Video gaming* categories. From the list of articles we extracted the structure of the document, i.e. sections and subsections of the text. Since these articles were featured on the home-page of Wikipedia, we assume that the contents are consolidated and their respective communities had refined its contents to maintain the same style among their pages.

We set two objectives for this preliminary analysis of the Wikipedia dataset:

- Finding a common structure among articles in the same category.
- Finding similarities in the structure of different articles.

4.5.1.1 Discovery of the *Structural* Process Model

In regards of our first objective, finding a common structure among articles of the same category, the literature of Business Process Management has proposed to use process discovery techniques for finding the *common structure* (i.e. the process model) for a set of Wikipedia articles (i.e. traces in which events are the title of each section). Nevertheless, the variability of event names on this Wikipedia articles is so high that the discovered models resemblance the flower model.

Table 4.1 summarizes the size and variability of our dataset. One may notice the high ratio of distinct events over events seen in the log. This will

[8]8th August 2016

[9]`https://en.wikipedia.org/wiki/Wikipedia:Featured_articles`

be an issue as it difficulties understandability of obtained process models. Besides, this also hints that lots of events are only seen once and, hence, patterns in the structure of the document is more difficult to find. In fact, Figure 4.6 shows that most of the section titles (more than 1000) have been seen only once in the dataset. Moreover, the number of events shared among traces is very low as the number of distinct events in the complete log (1347) is not so different to the sum of distinct events considering the traces of the 6 different datasets individually (1623). Therefore, comparison of traces between categories is almost impossible. Unfortunately, the most repeated section names are not useful for comparing two documents, as those sections are used for referencing external documentation (i.e. *External Links*, *References*) and introducing the article (such as Contents). Nevertheless, after discovering the abstract events we see a complete different picture allowing us to further analyse this dataset. In order to overcome this challenge, we applied the embedded-based event variability reduction technique for discovering a set of 50 abstract events shared among all the Wikipedia articles.

Table 4.2 shows six activities discovered by applying the embedded-based Event Variability Reduction . One may check that Cluster 4 trivially reefers to sections involving *writing*, and Cluster 5 combines sections containing *return to*. Nevertheless, other combinations are less trivial such as the sections included in Cluster 2. Unfortunately, some groups are not as accurate as the aforementioned. For instance, the first cluster seems to contain topics related to *philosophy*, *religion* and *history*. The three topics are certainly related, but a better granularity on such cluster may benefit later uses of the

	Events	**Distinct events**	**Abstract events**	**Traces**
History biography	1029	384	39	96
Media biography	677	157	40	66
Music biography	1257	246	41	120
Literature	1862	373	46	181
Video games	1253	124	36	140
Media	2718	339	46	268
Total	8756	1347	50	871

Table 4.1: Table summarizing the size and variability of our dataset. Abstract events have been found using the methodology explained in Section 4.4.

discovered activities.

The embedded-based Event Variability Reduction replaces all the listed Wikipedia titles in Table 4.2 with the assigned Cluster number. Therefore, one may take the set of titles as the new *event name*. This may hinder the understandability of the discovered abstract activities, as the practitioner needs to take a look into the clustered items, as we have done in the above paragraph, to have an understanding of their relation.

Continuing the log analysis, we run a process discovery method on each of the logs. In particular, we run the Inductive Miner[10]. The process models discovered are depicted in Figures 4.7, 4.8, 4.9, 4.10, 4.11 and 4.12. Most of them have a small subprocess with a flower-like behavior, and an in-depth

[10]We run the infrequent version of the Inductive Miner, with default parameters, on ProM 6.5.1.

Cluster 1	**Cluster 2**	**Cluster 3**
in culture	travels	aftermath of centralia
philosophy	first voyages	history of the manuscripts
re-discovery	privateering expedition	with the five
mythology	journey	the poems
historiography	trans-pacific voyage	in the media
Cluster 4	**Cluster 5**	**Cluster 6**
writing and publishing	return to united states	lineups
writing career	return to france	collaborations
writing style	return to missouri	recording
writing history	return to canada	discography
writing	return to japan	collaborators

Table 4.2: 5 randomly chosen section titles from the first 6 out of 50 discovered clusters from the Wikipedia dataset.

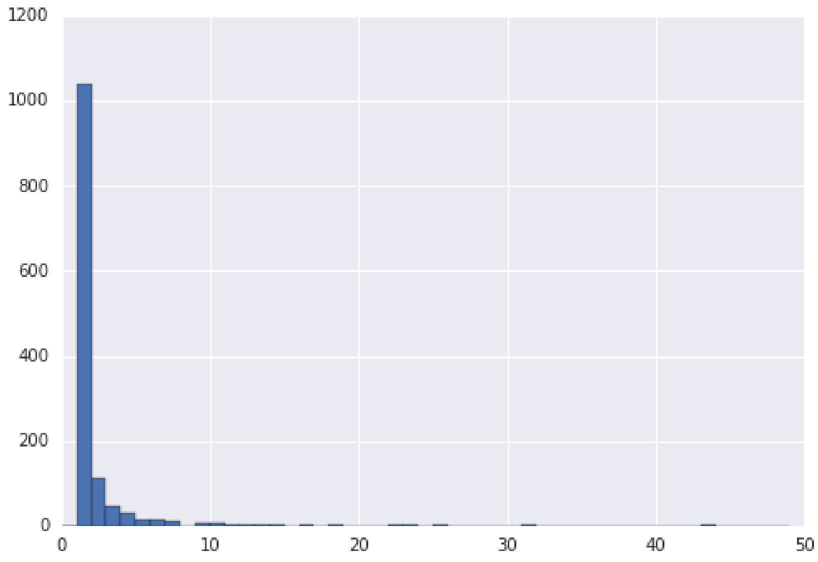


Figure 4.6: Absolute frequency of sentences used in less than 50 Wikipedia articles. Only 35 sentences out of the 1347 sections surpass the threshold.

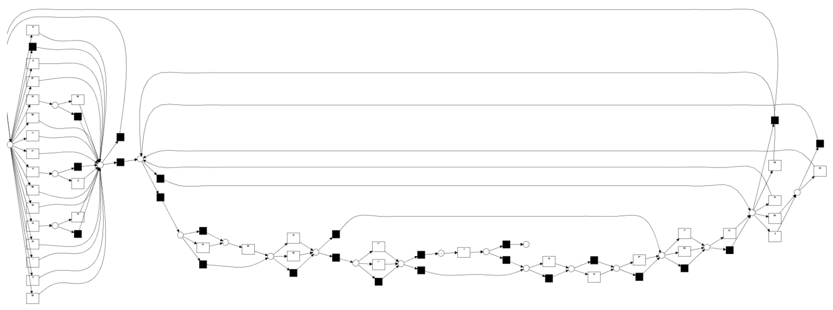


Figure 4.7: Petri net discovered from articles in the History biography category.

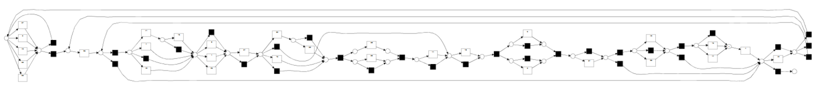


Figure 4.8: Petri net discovered from articles in the Media biography category.

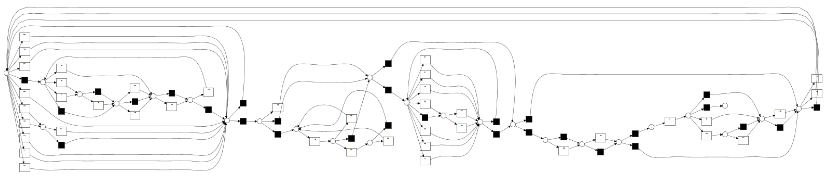


Figure 4.9: Petri net discovered from articles in the Music biography category.

analysis of the traces and abstract events highlighted several section and subsections with similar names that may happen in any ordering. Nevertheless, in general, a more detailed pattern has been found in this dataset thanks to the event abstraction.

4.5.1.2 Comparison of the Structure on Wikipedia Articles

Apart from the discovery of the *structural* process models, we would also like to showcase that the Event Variability Reduction could also help in the comparison of two Wikipedia articles. In particular, we consider conformance checking for getting an intuition of how similar are articles from different categories.

Table 4.3 depicts fitness and precision of the discovered process models with respect to all the abstract event logs. Fitness is, in general, high for the

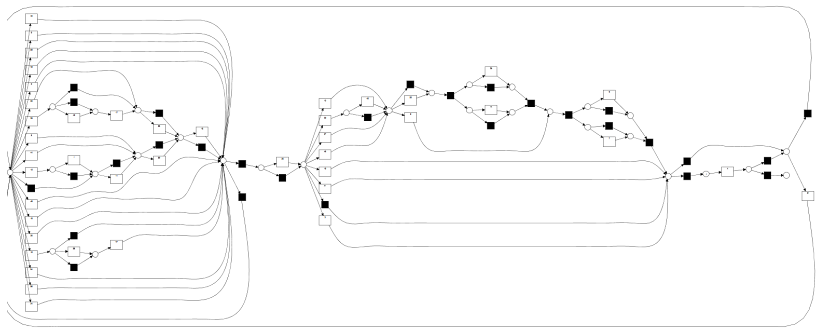


Figure 4.10: Petri net discovered from articles in the Literature category.

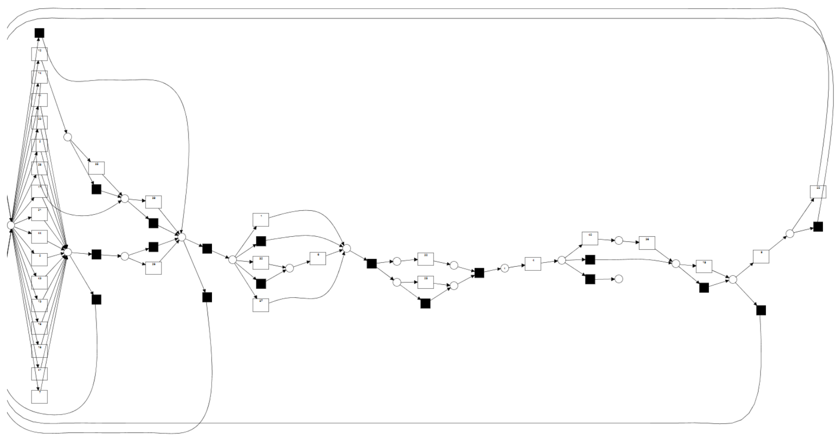


Figure 4.11: Petri net discovered from articles in the Video gaming category.

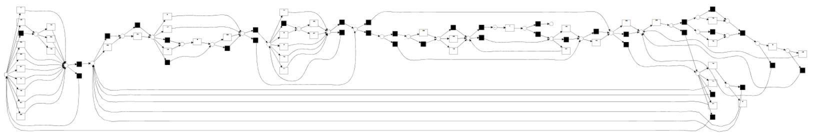


Figure 4.12: Petri net discovered from articles in the Media category.

		History biography	Media biography	Music biography	Literature	Video games	Media
History biography	F	**0.94**	0.79	0.72	0.79	0.65	0.68
	P	0.37	0.37	0.37	0.39	0.41	0.43
Media biography	F	0.73	**0.91**	0.77	0.69	0.59	0.61
	P	0.47	0.42	0.46	0.49	0.50	0.52
Music biography	F	0.88	0.92	**0.97**	0.86	0.69	0.72
	P	0.35	0.34	0.34	0.36	0.38	0.39
Literature	F	0.71	0.66	0.66	**0.81**	0.75	0.80
	P	0.41	0.39	0.45	0.48	0.46	0.48
Video games	F	0.44	0.49	0.55	0.64	**0.87**	0.73
	P	0.66	0.64	0.65	0.64	0.55	0.65
Media	F	0.74	0.75	0.75	0.83	**0.97**	0.93
	P	0.43	0.42	0.44	0.42	0.43	0.46

Table 4.3: Table summarizing quality of the discovered process models. For each row, fitness and precision of a process model is measured with respect to all the logs.

logs used for discovering the process model and slightly lower for the other logs. One may notice that the three *biography* process models have high fitness with the biography logs, but it is significantly lower with respect to *Video gaming* and *Media* categories. The contrary also holds, as fitness of the *Video gaming* and *Media* process models is significantly lower with respect to the three biography logs. On the other hand, the *Literature* category fits fairly well in both groups.

These results were expected, as all *biography* articles should have a similar structure (although talking about different types of artists or personalities) and *videogames* are nowadays produced as most of films and series (as they appear in the *media* category). On the contrary, *literature* articles usually talk about the authors and historical context of the book, and also about its plot (which is very common in *videogames* and *media* articles).

4.5.2 Application of the Event Variability Reduction to Trace Monitoring of Human-driven Processes

Some textual documents such as Tickets in Support systems, or live support chats, evolve over time. For example, tickets consist of a sequence of messages exchanged between a customer and one or more support engineers. The first messages usually provide a first description of the problem. Nevertheless, the content may evolve throughout the chain of messages and derive to other topic as the root-cause of the issue is being discovered. When the conversation between the customer and the support team ends, the ticket is usually enriched with extra information about the conversation outcomes such as the product causing the issue, type of fix needed, solution proposed, time to complete the issue, or satisfaction of the client with the support team. If this final information is known during the conversation, a support engineer would be able to better guide the conversation.

Initial conversations with the Data Analytics Team and the Support Team of CA Technologies highlighted the utility of being able to visualize the evolution of support tickets. Industrial-ready tools for technical support centers usually rely on a fixed set of *troubleshooting stages*, which must be updated by support engineers as tickets evolve. Even though this allow managers to focus on understanding in which phases is possible to reduce the time-to-resolution of support tickets, no further insights on the real status of open tickets can be retrieved. In particular, one interesting question to answer is if a customer is satisfied with the current performance of an open ticket, and how we can improve the satisfaction given its current status.

Customer escalation is the formal mechanism that customers have to warn support engineers, and the company, that the resolution of an issue is not as fast and smooth as they expected. The number of escalations is used as a Key Performance Indicator (KPI) for measuring the quality of support teams, and it is clearly an indicator of customer dissatisfaction and churning.

The objective of this experiment is to discover a model that best describes the usual evolution of topics in textual conversations and use this model to classify ongoing communications taking into account the evolution of the conversation so far. Following the methodology explained in 4.4, topics are automatically discovered by clustering vector representations of the textual messages contained in a support ticket. I.e. we train a document embedding that is able to summarize the meaning of the textual messages into a fixed-length vector, and then we perform a clustering technique to group similar messages. Word embeddings have been already used for sentiment analysis [93], and therefore there is some evidence that embeddings may be also able to highlight the customer dissatisfaction from subtle textual differences. The embedding techniques presented in Section 4.4 are not tailored for measuring the similarity of long texts as typical messages in a Support Center. Nevertheless, adaptations of the original Word2Vec allows for applying this technique for embedding large documents. In particular, we applied the Doc2Vec technique [62], and the similarity of two documents is defined as the cosine similarity of the two document embeddings.

Process Mining techniques allow us to discover a process summarizing the evolution of the newly generated topics. Therefore, one of the immediate benefits of document embeddings is visualization of the current status of the support ticket, and measuring the deviation from the general ticket resolution. Figure 4.13 depicts the discovered usual evolution of a subset of all support tickets in 2015, whilst Figures 4.14 and 4.15 depicts the discovered models for tickets that escalated and those which successfully ended. For these examples, we use the Fuzzy Miner algorithm with default parameter. For clarity of the results, we depicted the results when only 30 activities were discovered with the embedded-based Event Variability Reduction. There is a clear difference in the structure of both process models, in which the escalated model is less structured that the non-escalated process model. This seems to indicate that customer complain of support cases that were not properly handled by the support engineer, but could also be an indicator that the support engineer needed to spent a lot of time to explore the customer issue

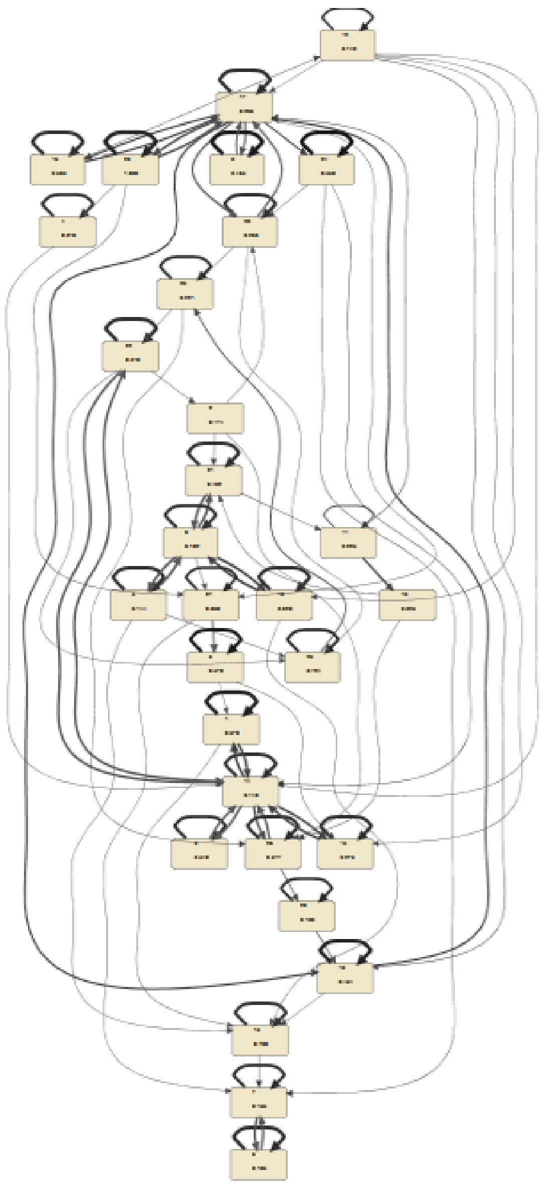


Figure 4.13: Process model describing a subset of technical support cases. 30 activities are depicted, in which each activity represents a cluster of textual messages.

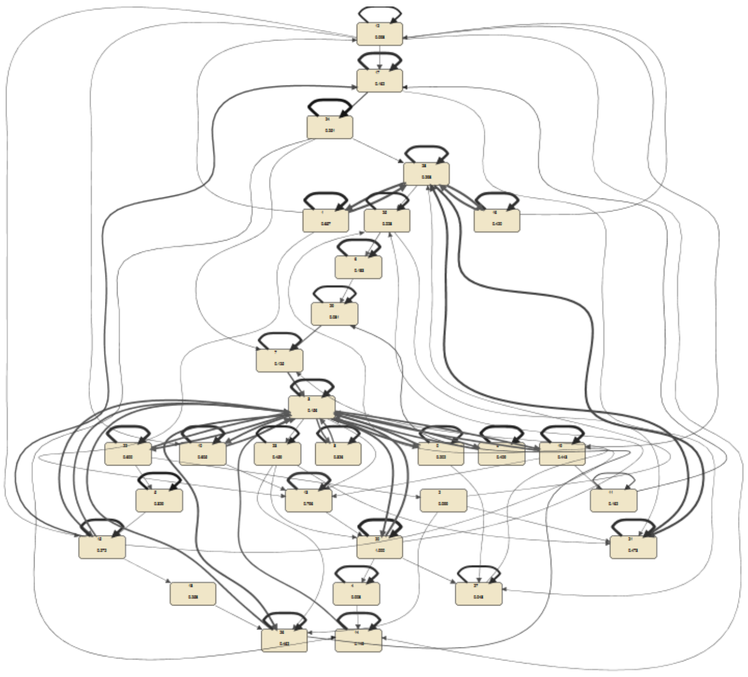


Figure 4.14: Process model describing a subset of all escalated technical support cases. The structure of this process model hints the lack of an structure on how escalated cases are handled by support engineers.

to find its root-cause.

Besides the process models discovered in Figures 4.14 and 4.15, we trained a Hidden Markov Models capable of classifying [25] all traces into the escalated and non-escalated categories. At the initialization step, we used the process models in the two figures as the structure of the hidden nodes. Then, probabilities were tuned during the training of the model for maximizing the accuracy of the classifier. Following this approach, roughly 10% of escalated

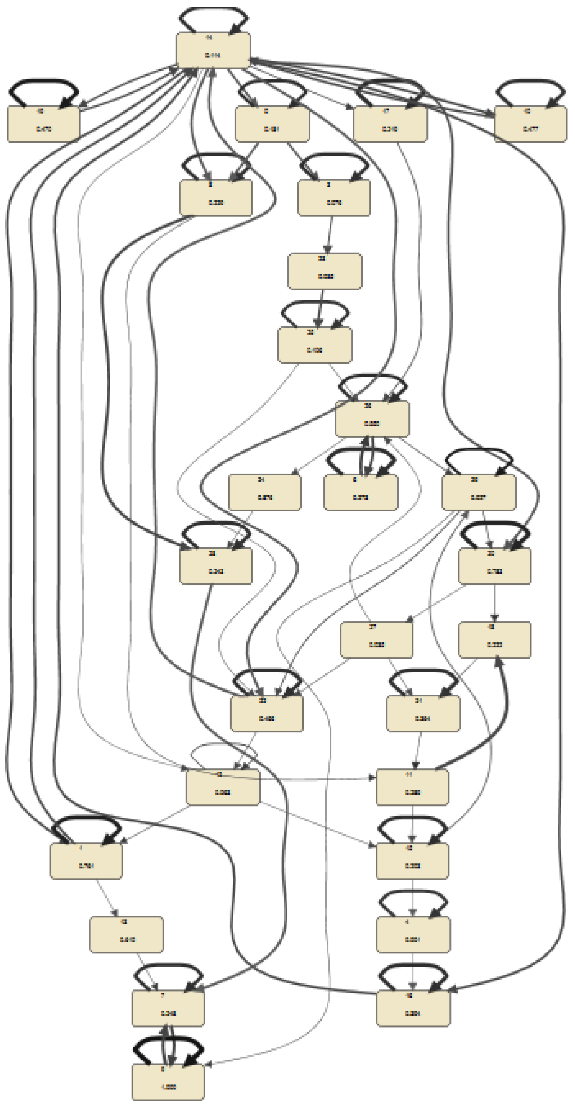


Figure 4.15: Process model describing a subset of all non-escalated technical support cases. This process model is more structured than the process model in Figure 4.14, pointing that deviations from the *normal process* are more common in escalated cases.

cases never seen by the Markov model were properly labeled by the classifier. The high imbalance between the escalated and non-escalated datasets may have caused the low ratio of detected escalations. Nevertheless, it has been acknowledged as a useful tool as it reduces the number of cases that need to be closely monitored for having an impact on customer satisfaction.

4.6 Discussion

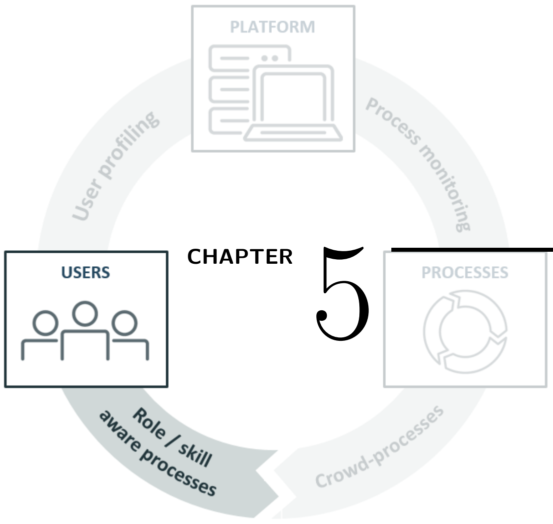
In this chapter, we have developed a method for reducing variability of event names by grouping them according to their similarity. Recent developments on Natural Language Processing allowed us to compute this similarity based on semantic information, instead of traditional bag-of-words techniques or creating a complex ontology.

We have applied this technique on a dataset compromising the structure of articles in Wikipedia. Initially, it was impossible to find any common structure nor finding any similarities between articles because section names were almost never repeated in the dataset. Nevertheless, after discovering the abstract events, common process discovery technique already discovered some patterns on the data and enable us to compare two different articles. Although this use case is very simplistic, the results validate the methodology and motivates further research on this direction.

Removing information from event names is usually seen as a reduction of the data quality and, hence, has been overlooked in the Business Process Management literature. Nevertheless, we have seen that there is an industrial necessity on monitoring processes with highly variable events, such as monitoring conversational bots.

Part II

Worker Profiling and Monitoring



CHAPTER 5

Worker Ranking Determination

We start the second part of this thesis, related to users or workers, by studying a particular crowdsourced process pattern that enables the platform to measure the skill acquisition of workers, and how it was implemented in an industrial prototype for crowdsourced translations. The novelty of such prototype relies on the role of the reviewer, played by skilled individuals on the platform, that acts as reviewers of the translations done by in-training translators. The feedback provided by the reviewers is later reused for deciding if an in-training translator should be promoted to the reviewer role.

This closes the gap between the users and processes in the framework of this thesis. We have already seen in Chapter 3 that the industry is demanding modelling languages that enable them to assign tasks to users with particular roles. Thanks to the outcomes of this chapter, the method for granting such roles is automatically governed by the continuous execution of the crowdsourced process – instead of using preliminary and ad-hoc Qualification tests.

5.1 Introduction

Historically, crowdsourcing was associated to low quality jobs. The participation of a huge amount of geographically distributed workers is at first glance assumed to be an obstacle for quality. In most of the cases, it is very costly and expensive to establish automatic mechanisms to monitor, control and evaluate the quality in crowdsourcing platforms. Due to this, a poor definition of quality is done and the complexity establishing a universally accepted quality measure makes job quality assurance a really complex task. Some examples are:

- How do we measure the degree of innovation of an idea?,
- How do we evaluate the beauty of a proposal?,
- How do we evaluate the writing style of an author?

All these measurements are highly subjective. Nevertheless, many previous proposals are still based on methods to monitor quality based on automatic measures or golden solutions, which might not be realistic in many different scenarios.

In this Chapter, we assume that human beings are required to evaluate the quality of the output of a task, as an indirect mechanism to evaluate the skills of other workers in the crowd [38, 58]. We propose to use the already calculated quality evaluation of past tasks as the input of an aggregation function. The aggregation function fuses them in a single datum able to summarize, rank and determine the profile and skills of individuals. Taking into account such profiles, we can better determine who are the most suitable individuals for a given task and what is the fairest economical reward. Note that, aggregation functions provide us a huge flexibility in the summarization process, allowing us to define rewards and punishments within their weighting vector. In addition to that, with our system it is also possible to establish an automatic promotion system for crowd workers based on their past tasks evaluations. This fact may also increase the workers' commitment for delivering high quality tasks because their quality will determine their future rewards.

5.2 Related work

Quality evaluation methods can be classified in three main families: (i) by direct inspection of the job provider, (ii) automatic, and (iii) methods using the crowd itself as evaluator. Evaluation by the job provider has an inherent scalability problem, since the crowd can produce a large amount of work and the job provider has a finite amount of expert resources to evaluate it. Clearly for most of the tasks, an automatic evaluation is either impossible or can only guarantee a minimum quality, otherwise it would be possible to set up a completely automated solution without human intervention. Venetis and Garcia-Molina propose "Gold Standard Performance" to detect workers' performance before the crowd-sourcing task starts [105]. Workers' characteristics such as demographics or personality traits are related to the quality of their work under specific task conditions [52]. The worker perception of five quality assurance mechanisms is also studied in [89]. In general, it is considered that inaccurate acceptance or rejection may not only affect a specific task in a platform, but may also encourage other fraudsters to misbehave in the platform. For example, Hirth et al. raise "Majority Decision Approach" [42] to judge whether worker's submission is correct in simple tasks, and using "Control Group Approach" method in complicated cases. Crowdsource the quality evaluation of the jobs performed by the crowd to avoid the use of such gold standard units, has been proposed as an alternative. The main idea behind this proposal is that human beings are the best quality evaluation method [38, 58], but this solution has the potential problem of trustworthiness and management of opinion and criteria disparity.

From the industrial perspective, first steps have been done to establish the basis for crowd coordination and create rewarding mechanisms that are based on involving human beings in the evaluation of the quality of other workers through the so-called AV-Units [76]. In [76] the authors propose an approach to deal with these two challenges (quality evaluation and worker motivation), by using a general mechanism to also crowdsource the quality evaluation of a job performed by the crowd and give workers economical rewards based on the quality of their work. To build a trustworthy crowdsourcing system effectively, two essential aspects have to be addressed: mechanisms for worker coordination to guarantee the correct evaluation of quality, and a reliable mechanism to monitor the skills of workers. Is in this latter aspect that aggregation functions play a central role.

5.3 Crowd-based Quality Evaluation.

The general mechanism of [76] that guarantees job quality is based on the idea that human beings are the best quality evaluation method in many situations [38, 58]. Any complex task is subdivided into a series of subtasks called *Action-Verification Units* (AV-Unit). AV-Units establish relationship patterns between the workers of the crowd to help them to provide a higher degree of quality working in a collaboratively manner.

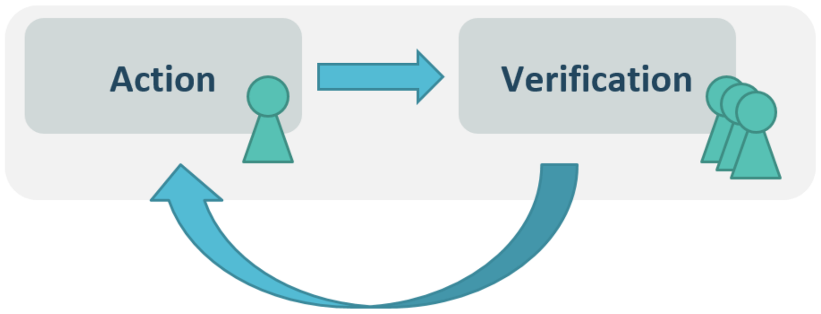


Figure 5.1: Action-Verification Unit (AV-Unit) [76]

Figure 5.1 depicts an AV-Unit, and a formalization as a CrowdWON process model can be found in Figure 5.2. As we observe, an AV-Unit is divided into two phases: Action and Verification. In the *Action* phase a single worker performs a specific action. In the *Verification* phase a set of workers verify the quality of the output generated in the previous Action phase. If the workers in the Verification phase consider that the quality provided is below a certain threshold, they might ask the first worker to repeat or improve the action. This process may be repeated iteratively until the output has reached a certain level of quality or the workers in the Verification phase decide to substitute the initial worker (or the worker is not available anymore). In practice, the Verification phase in the AV-Unit acts as a quality filter barrier, that does not allow to proceed with the process until the quality of each step in this process is approved by a set of human evaluators working collaboratively. Note that when more than one worker participates in the Verification phase, an aggregation function is also used to aggregate the decisions (scores) of each individual worker and produce a final decision (*i.e.*, the job has the required quality or not).

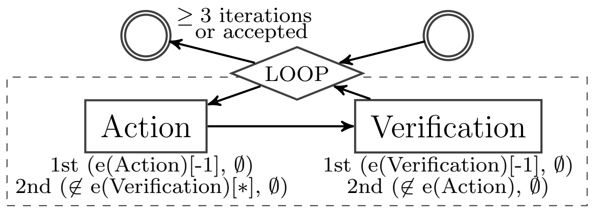

$\geq$ 3 iterations or accepted
LOOP
Action
Verification
1st (e(Action)[-1], $\emptyset$)
2nd ($\notin$ e(Verification)[$*$], $\emptyset$)
1st (e(Verification)[-1], $\emptyset$)
2nd ($\notin$ e(Action), $\emptyset$)

Figure 5.2: An Action Verification unit graphically modelled as a Crowd-WON process model. In this particular instance, only one reviewer is considered for each verification.

5.4 Aggregation Functions

Aggregation functions [97] are numerical functions used for information fusion that combine N numerical values into a single one. These operators are formally described as follows:

Definition 5.1. *Let $X := \{x_1, \ldots, x_N\}$ be a set of information sources, and let $f(x_i)$ be a function that models the value supplied by the i-th information* *source x_i (for the sake of simplicity we often denote $f(x_i)$ by a_i), then a function $\mathbb{C} : \mathbb{R}^N \to \mathbb{R}$ is said to be an aggregation function if it satisfies:*

1. $\inf \mathbb{C}(a_1, \dots, a_N) = -\infty$ *and* $\sup \mathbb{C}(a_1, \dots, a_N) = +\infty$ *(boundary condition)*

2. $\mathbb{C}(a_1, \dots, a_N) \leq \mathbb{C}(a'_1, \dots, a'_N)$ *if* $a_i < a'_i$ *(monotonicity)*

There are several aggregation functions in the literature (see *e.g.* [22, 97] for further review) but we will focus only on idempotent aggregation functions, *i.e.* the aggregated value $\mathbb{C}(a, \dots, a)$ is a. Among them, the most well-known functions are the arithmetic mean (AM) and the weighted mean (WM).

Yager defined in [110] the Ordered Weighted Averaging (OWA) aggregation function as a weighted linear combination of order statistics. In short, it works like a weighted mean after ordering the values a_i. We provide below a definition of the OWA operator using a non-decreasing function, as this is the most useful approach in our context.

Definition 5.2. *Let Q be a non-decreasing function in $[0, 1]$ which satisfies the boundary condition $Q(0) = 0$ and $Q(1) = 1$, then the mapping $OWA_Q : \mathbb{R}^N \to \mathbb{R}$ defined as follows is an OWA operator:*

$$OWA_Q(a_1, \dots, a_N) = \sum_{i=1}^{N} \big(Q(i/N) - Q((i-1)/N)\big) a_{\sigma(i)}$$

where σ is a permutation of $\{1, \dots, N\}$ such that $a_{\sigma(i)} \geq a_{\sigma(i+1)}$.

This operator has several properties. We underline the following ones:

i) For all Q, it holds that:

$$\min_i a_i \leq \mathrm{OWA}_Q(a_1, \ldots, a_N) \leq \max_i a_i.$$

The choice of the function Q allows us to modulate OWA_Q from the minimum to maximum function. For example, when we consider the family of functions $Q_\alpha(x) = x^\alpha$, also called Yager Quantifiers, we have that large positive values of α lead to an OWA near to the minimum and, on the contrary, values of α near to zero lead to an OWA near to the maximum. Besides, when $\mathbf{a} = (a_1, \ldots, a_N)$ is fixed, OWA_{Q_α} is non-decreasing with respect to α.

ii) The OWA operator is symmetric for all Q. That is, the order of the parameters is not relevant for the computation of the output.

OWA operators are generalised by Choquet integrals [98] with respect to fuzzy measures, a family of fuzzy integrals that can be used for information fusion. In short, given a function f that represents the information supplied by the sources in X, the Choquet integral of f represents an aggregated value of those in f. In such integrals, fuzzy measures play the role of weights in the weighted mean.

Recall that a fuzzy measure μ is a set function over X such that the two boundary conditions hold (*i.e.* $\mu(\emptyset) = 0$ and $\mu(X) = 1$) and $\mu(A) \leq \mu(B)$ for every two subsets $A \subseteq B$ of X. A fuzzy measure is symmetric if it only depends on the cardinality of the set.

Formally, the Choquet integral is defined as follows:

Definition 5.3. *Let μ be a fuzzy measure on X; then, the Choquet integral of a function $f : X \to \mathbb{R}^+$ with respect to the fuzzy measure μ is defined by*

$$(C)\int f\,\mathrm{d}\mu = \sum_{i=1}^{N} a_{\sigma(i)}[\mu(A_{\sigma(i)}) - \mu(A_{\sigma(i-1)})]$$

where σ is a permutation such that $a_{\sigma(i)} \geq a_{\sigma(i+1)}$ and $A_{\sigma(i)} = \{x_{\sigma(1)}, \ldots, x_{\sigma(i)}\}$.

The Choquet integral with respect to the fuzzy measure $\mu(A) = Q(|A|/N)$ is precisely the OWA$_Q$ operator. This equivalence shows that OWA weights do not depend on the information sources nor possible relations between them. On the other hand, such independence is not required in the definition of a fuzzy measure and further aggregation functions can be defined with the Choquet integral (see *e.g.* [97] for a definition of the Weighted Ordered Weighted Averaging (WOWA) operator).

When a symmetric fuzzy measure is used, the Choquet integral is symmetric as the OWA operator. This property also holds for other fuzzy integrals. In particular, it also holds for the Sugeno integral [98]. Formally, the Sugeno integral is defined as follows:

Definition 5.4. *Let μ be a fuzzy measure on X; then, the Sugeno integral of a function $f : X \to [0, 1]$ with respect to the fuzzy measure μ is defined by*

$$(S)\int f\,\mathrm{d}\mu = \bigvee_{i=1}^{N} (a_{s(i)} \wedge \mu(A_{s(i)}))$$

where $\vee$ stands for maximum, $\wedge$ stands for minimum, s is a permutation such that $a_{s(i)} \leq a_{s(i+1)}$ and $A_{s(i)} = \{x_{s(i)}, \ldots, x_{\sigma(N)}\}$.

In particular, we can choose the symmetric fuzzy measure $\mu(A) = Q(|A|/N)$ as in the Choquet integral, obtaining an equivalent to the to the OWMax operator defined by Yager in [111].

Definition 5.5. *Let Q be a non-decreasing function in $[0,1]$ such that $Q(0) = 0$ and $Q(1) = 1$, then the mapping $SI_Q : \mathbb{R}^N \to \mathbb{R}$ defined as follows is a Sugeno integral of a function $f : X \to [0,1]$ with respect to the fuzzy measure $\mu(A) = Q(|A|/N)$:*

$$SI_Q(a_i) = \bigvee_{i=1}^{N} (a_{s(i)} \wedge Q(i/N))$$

where s is a permutation such that $a_{s(i)} \geq a_{s(i+1)}$.

The twofold integral [78, 96] is a generalisation for both Choquet and Sugeno integrals. The twofold integral is a fuzzy integral that aggregates a function with respect to two fuzzy measures. The rationale of this generalisation is that the semantics of both measures are different. In particular, the measure in the Choquet integral is seen as a 'probabilistic flavour' measure, and the measure used in the Sugeno integral is seen as a 'fuzzy flavour' measure. We use μ_C to denote the measure that corresponds to the one in the Choquet integral, and μ_S for the one in the Sugeno integral.

Definition 5.6. *Let μ_C and μ_S be two fuzzy measures on X, then the* twofold integral *of a function $f : X \to [0,1]$ with respect to the fuzzy measures μ_S and μ_C, denoted $TI_{\mu_S,\mu_C}(f)$, is defined by:*

$$\sum_{i=1}^{N} \Big(\big(\bigvee_{j=1}^{i} a_{s(j)} \wedge \mu_S(A_{s(j)}) \big) \big(\mu_C(A_{s(i)}) - \mu_C(A_{s(i+1)}) \big) \Big)$$

where s is a permutation such that $0 \leq a_{s(i)} \leq a_{s(i+1)} \leq 1$ and $A_{s(i)} = \{x_{s(i)}, \ldots, x_{s(n)}\}$.

5.5 Using Worker Ranking for Trustworthiness Measuring

The success of AV-Units highly depends on the workers' profile. Involving many workers with low skills in an AV-Unit, might have a negative impact on the final quality. In most of industrial processes, quality standards are high and trusting the individuals in the crowd and their capacity to carry out the different tasks assigned to them is essential. Because of this, the main concern of a crowdsourcing platform is to monitor workers in order to evaluate and update their skills based on the quality of their past tasks.

To cope with this requirement, we propose to use a ranking systems that dynamically modifies the worker skills. Specifically, there are several aspects that might influence such a ranking system:

- **The worker quality is measured from the output of their past jobs**: it is necessary to establish rewarding and penalty measures that modify the ranking of the workers in the crowd. In general, the actions with a higher impact for workers are those performed in the Action phase of an AV-Unit. However, it would be also possible to modify the ranking of workers based on their activity when they are acting in a Verification phase.

- **General behaviour of the workers in the crowd**: other aspects might influence the ranking, *i.e.* worker commitment, career, etc. For instance, a worker might click very fast in order to get solutions quickly and get an economical reward. Although, improper behaviour will lead with high probability to bad quality, taking into account behavioural patterns may help to multiply the positive or negative impact of actions in the corresponding worker's ranking and speeding up the detection of incorrect behaviour.

The main idea behind using ranking systems is that a worker with a higher rank will be more trustworthy than workers with lower ranks. Therefore, it is possible to set a fair payment system where workers quality modulates the economical reward, increasing in this way the workers motivation to deliver high quality outputs in their future tasks. Additionally, ranking systems might allow us to automatically determine the most suitable workers for a given task.

5.5.1 Automatic Worker Ranking Determination

As aforementioned, a natural way to determine the rank of crowd workers is by combining the quality values of their previous works. This can be achieved by using aggregation functions, which were reviewed in Section 5.4. Concretely, for each Action unit of previous AV-Units, denoted by x_i, that has at least one Validation value a_i, it is possible to combine all these quality values a_i in a single aggregated number $\mathbb{C}(a_1, \ldots, a_n)$. Note that, for a worker, this aggregated value cannot be bigger than the maximum quality achieved for any of her previous works nor lesser than the worst.

The choice of the aggregation function will lead to different ways of promoting and demoting crowd workers. For instance, the use of the arithmetic mean (AM) equally considers all previous works, independently on whether a task has been recently delivered or not. In this case, a bad quality task delivered during the training period counts exactly the same as a more recent task when the worker skills are better. To overcome this issue, we can replace the AM by the weighed mean operator (WM), doing this it is possible to give more importance to recent tasks than older ones.

On the other hand, by using OWA operators it is possible to emphasise extreme quality values. For example, using the fuzzy quantifier $Q(x) = x^{\alpha}$ with α close to 0, we are giving more importance to the highest quality tasks of a certain worker, without taking into account when the task was done (near in the past or not). In this setting, we are assuming that future tasks will also have a high quality because the worker finalised in the past (at least one time) a high quality task.

On the contrary, by using OWA operators with the same fuzzy quantifier but with $\alpha = 2$, the bigger the growth of the quantifier is when x values are near to one. Therefore, the associated weight of the low quality tasks will be bigger than the high quality ones. In this setting, we are penalising a lot workers with some low quality tasks. In this model, workers will find more difficulties to improve their rank. Finally, we can reduce the relevance of possible outliers by choosing a fuzzy quantifier $Q(x)$ with slow growth near $x = 0$ and $x = 1$.

In certain industrial scenarios, it may be interesting to consider both dimensions: task quality and when the task was delivered. To do this, it is necessary to use the twofold integral. In this integral, it is possible to include two fuzzy quantifiers, one for delivering time Q_t and another for task quality Q_q.

5.6 Crowd-based Text Translation: A Practical Example

For this example we use a crowdsourcing platform for software and text localization[1] that is being developed by CA Technologies (CA) for translating their technical documentation and marketing materials, applying the aforementioned AV-Units for ensuring task quality. In such platform, crowd workers are divided into three disjoint categories: *newcomers*, *associates* and *seniors*. Figure 5.3 presents a description of each category. Newcomers are workers that use the platform to learn and improve their professional translation and post-edition (reviewing manually the output of an automatic translation) skills. They do not receive any economical reward for their translations and quality in real translations never depends on their work, but they receive feedback from senior translators to improve their skills. The platform also offers them many training examples. The main task of associate workers is to post-edit texts and their economic reward mainly depends on the text length. Finally, senior workers are those that know CA quality standards for translations and have proven very high skills in translation and post-edition. Their main task is to verify the work done by associates and provide them feedback to improve the quality of their translations and, as a consequence, their rank to become seniors. Note that, the economical reward of a senior translator is higher than that of an associate translator. The platform also considers that an associate translator becomes a senior translator if the

[1]similar to text translation [79, 113], but localization takes into account cultural differences between countries with the same first language.

quality of translations is high compared to the quality obtained by senior translators.

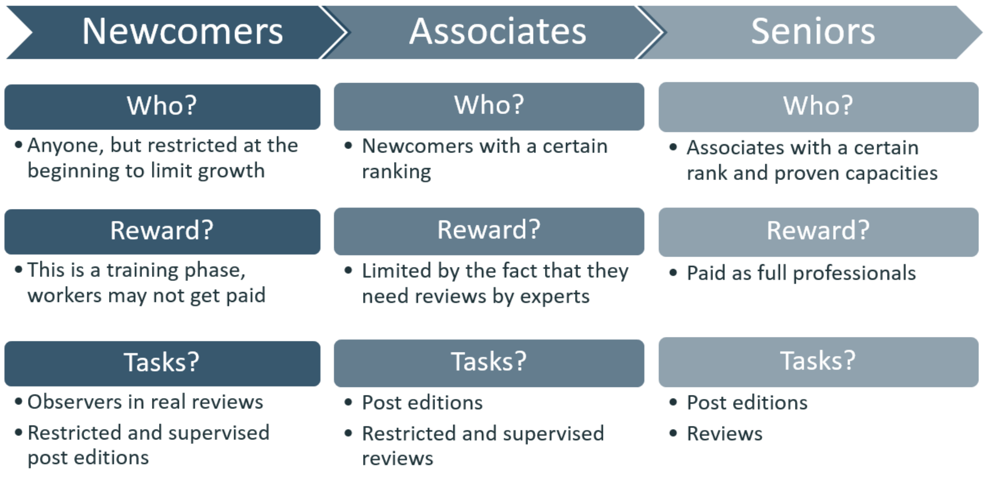

Newcomers
Associates
Seniors
Who?
Anyone, but restricted at the beginning to limit growth
Reward?
This is a training phase, workers may not get paid
Tasks?
Observers in real reviews
Restricted and supervised post editions
Who?
Newcomers with a certain ranking
Reward?
Limited by the fact that they need reviews by experts
Tasks?
Post editions
Restricted and supervised reviews
Who?
Associates with a certain rank and proven capacities
Reward?
Paid as full professionals
Tasks?
Post editions
Reviews

Figure 5.3: Translator categories at the CA Technologies crowdsourcing platform for localization.

5.6.1 TQI, a Quality Measure For Text Translation

Translation Quality Index (TQI) [88] measure is the standard way to evaluate the quality of a professional translation. To compute the TQI, it is necessary that an expert review the translated text to detect the errors and evaluate their severity. Therefore, TQI measure reflects the criterion of such expert. To help experts to decide the severity level of an error, CA Technologies provides the following categories:

Sev1 Linguistic issues that bring the most direct (critical) impact to end users, such as:

- unexpected functional results, which are different from the English product description/statement
- deterring subsequent operations from product execution
- causing product features to fail
- carrying negative legal, political, security, and financial consequences or cultural noncompliance

Sev2 Linguistic issues that bring indirect yet substantial impact, leading the end user to:

- difficulties in understanding functionalities and technologies invented and developed in CA products
- misleading or incorrect interpretation of concepts for CA products

Sev3 Linguistic issues that do not impact the end user operation, yet impact the end user experience such as:

- spending additional time trying to figure out the meaning of descriptions/statements
- inconsistencies when trying to read product materials
- incompatibilities with layout and formatting

Sev4 Linguistic issues with minimal impact to overall end user experience, such as:

- noticeable and tolerable minor linguistic flaws
- layout and formatting errors that are only visible by comparing to the English source

Table 5.1: Text Quality Levels based on TQI ranking.

Level	Criteria
Excellent	$\text{TQI} \geq 90$
Good	$80 \leq \text{TQI} < 90$
Fair	$70 \leq \text{TQI} < 80$
Acceptable	$60 \leq \text{TQI} < 70$
Reject	$\text{TQI} < 60$

Once all the translation errors have been detected and categorised, TQI is calculated as follows,

$$\text{Total} = 6\,a_1 + 2\,a_2 + 1.2\,a_3 + 0.8\,a_4$$
$$\text{TQI} = 100 - 40\,\frac{\text{Total}\,/\,\text{Number of Reviewed Words}}{0.01}$$

where a_i means the number of Sev i issues detected in the text. The final quality levels of a given translation are ranked in five categories, as it is depicted in Table 5.1. In the next section, we use similar quality levels to decide whether a worker is ranked as newcomer, associate or senior.

5.6.2 AV-Units Applied to Text Translations

In this section, we present how to use AV-Units to create a crowd-based platform for text translation. The platform works as follows: Firstly, the source texts go through a rule-based machine translation engine and a first automatic translation is produced. Secondly, the original and the machine-translated texts are sent to external human translators who post-edit the text written in the target language, we call this step as post-edition step (PE), and it corresponds to the action part of the AV-Units methodology. Thirdly, multiple (from one to three) text verifications are performed by CA translators. It is important to note here that verifiers cannot modify text, they can only inform about detected errors. We call this part verification step (VE), and it corresponds to the verification part of the AV-Units. If the translation does not reach the minimum quality level (TQI ≥ 80), a new PE and VE steps are performed. To avoid infinite loops, in this second round verifiers are allowed to modify the text if they think it is required to maintain a TQI quality up to 80. After this process, the text is ready to be published.

5.6.3 Worker Categories and Promotion Mechanisms

As we have introduced before, workers are ranked into three different categories. To determine a worker category, we aggregate the TQI values obtained in the past into an overall TQI value, as it is explained in Subsection 5.5.1. Table 5.2 depicts worker categories from the obtained overall TQI values. The platform assumes that the minimum overall TQI level for being considered an associate translator is equal to 60. Remember that translations with a TQI quality measure below 60 are rejected. For being considered a senior translator, the overall TQI obtained by a worker must be up to 80, since this is the minimum level required for being a CA internal translator.

In the platform, workers are automatically promoted or demoted depending on the overall TQI obtained in their past translations.

Table 5.2: Worker Ranking Categories.

Category	TQI value
Senior	$\mathrm{TQI} \geq 80$
Associate	$60 \leq \mathrm{TQI} < 80$
Beginner	$\mathrm{TQI} < 60$

5.6.4 Numerical examples

Some experiments have been done using the weighted mean (WM), OWA operators and the twofold integral on four different workers (U, I, D, O). We have generated 300 TQI scores for the U, I, D and O workers by the following strategy:

- The worker U has a stable performance, in the sense that she is not learning from the feedback provided by the *Verification* phase. In the simulation, each word has a fixed probability for generating a Severity 1, 2, 3 or 4 error.

- The worker I slightly increases her performance with every translation, simulating that she learns from every *Verification* phase. In the simulation, the probability for generating an error is decreased for every iteration.

- The worker D slightly decreases her performance with every translation, simulating a worker that is losing commitment and motivation.

Her probabilities for generating an error are increased after every iteration.

- Finally, the worker O deliberately increases or decreases her performance in order to achieve a certain status in the platform. In the simulation, the probability for generating an error depends on how close her score is to 80, being significantly higher when her score is above 80. This type of worker simulates a *fraudster* that is willing to get profit from the implemented reputation system.

During this simulation, we aim to study how the selected aggregation functions impact on the evaluation and performance of the four workers.

Two families of weights have been used, defined by the fuzzy quantifiers

$$Q^e_\alpha(x) = x^\alpha \qquad\qquad Q^s_\alpha(x) = \frac{1}{1+e^{\alpha - x}}$$

$$Q^t_\alpha(x) = \begin{cases} 0 & \text{if } x \le \alpha \\ 1 & \text{if } x > \alpha \end{cases}$$

Here, Q^e stands for exponential function, Q^s for the sigmoidal function and Q^t for the threshold function. For these experiments, we considered the following combinations of weights and aggregation operators:

- The weighted mean with weights computed based on $Q^e_{0.5}$, $Q^e_{2.0}$, $Q^s_{0.5}$. By using $Q^e_{2.0}$ we are putting more emphasis on the latest TQI scores of the worker, whilst $Q^e_{0.5}$ considers the first scores of the worker. On the other hand, $Q^s_{0.5}$ reduces the relevance of both latest scores and preliminary results, and averages in-between results.

- For the OWA operator, we chose the functions $Q^e_{0.5}$, $Q^e_{2.0}$ and $Q^s_{0.5}$ to compute the weights. Since we order TQI scores first, $Q^e_{2.0}$ now focus on the highest scores obtained by the worker, whilst $Q^e_{0.5}$ considers the lowest scores obtained. In order to reduce the relevance of TQI outliers, the $Q^s_{0.5}$ could be considered.

- For the Twofold integration we considered the $Q^t_{0.2}$ threshold function, meaning that the top 20% scores are reduced to the immediately score below. For example, for a user with ten scores $s_0 < s_1 < s_2 < \ldots < s_9$, the twofold integration would assign the value s_7 to both s_8 and s_9. After this cut, an OWA operator is used to aggregate the score. For this experiment, we used again the functions $Q^e_{0.5}$, $Q^e_{2.0}$ and $Q^s_{0.5}$.

Aggregated scores for workers *U*, *I* and *D* Figures 5.4, 5.5 and 5.6 depict the aggregated scores obtained by the chosen aggregation functions for the worker U (Figure 5.4), worker I (Figure 5.5) and worker D (Figure 5.6). The general impression on these figures is that all aggregated scores show an stable behavior (although some operators are more prone to recent changes and, hence, variability is slightly high between iterations) fruit of the monotony of the TQI scores. Nevertheless, the OWA operator and Twofold integration (using $Q^e_{0.5}$ for weights) consistently shows a pessimistic view of the worker's performance. This position the two operators as good candidates for being considered when the platform is in a conservative position, and *Senior* position requires major commitment from workers. We have also detected that there are no significant differences between the twofold integration and the OWA operator on the long-term vision of the tool as

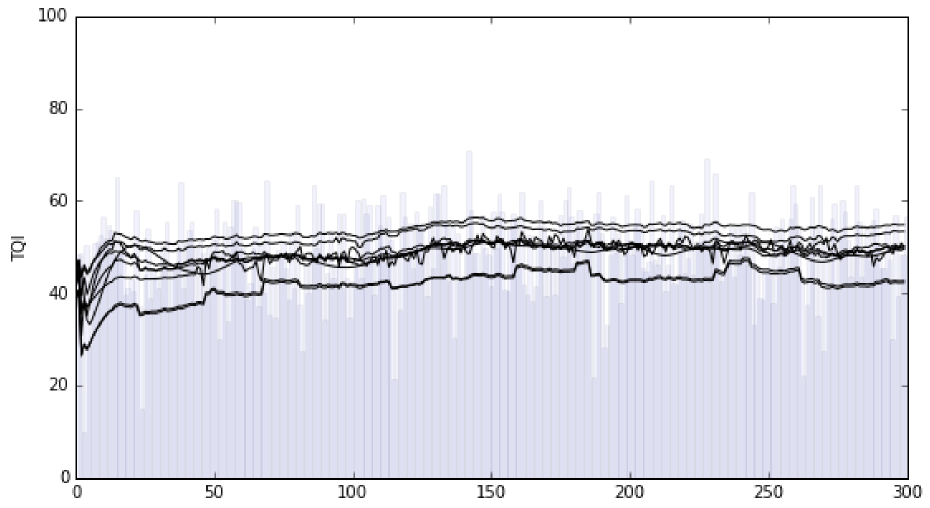


Figure 5.4: Bar plot of the TQI scores obtained by the U worker, and the aggregated scores obtained with the selected operators.

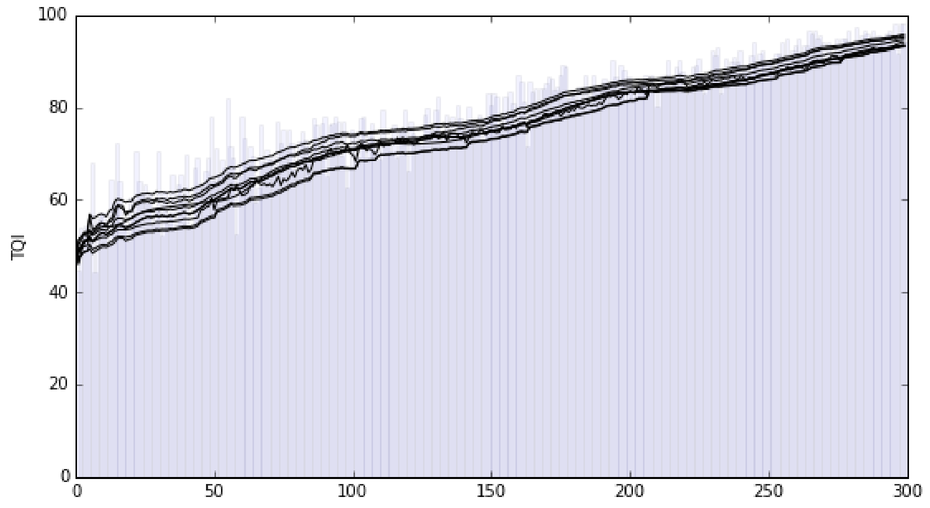


Figure 5.5: Bar plot of the TQI scores obtained by the I worker, and the aggregated scores obtained with the selected operators.

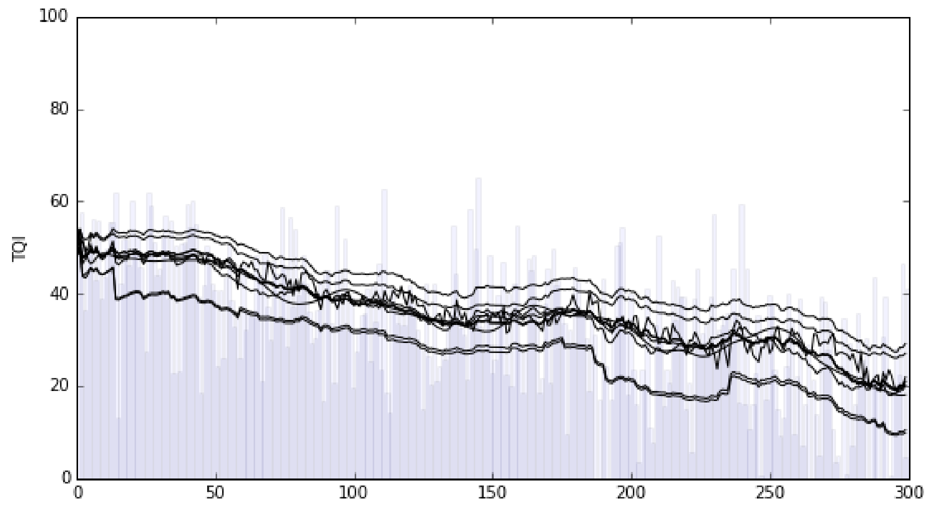


Figure 5.6: Bar plot of the TQI scores obtained by the D worker, and the aggregated scores obtained with the selected operators.

the twofold integration is resilient towards unexpected improvements on the worker's performance (behavior not seen on workers U, I and D).

Aggregated scores for worker *O* Worker O provides a more interesting simulation of the aggregated scores, as the behavior of the worker changes depending on their current score. For the simulations of worker O we have chosen one metric to be provided to the worker so he knows his current status on the platform.

Figure 5.7 depicts the TQI scores obtained when worker O is aware of her own average score. In general, we are obtaining an expected behavior from this user. Her performance ranges from 60 to roughly 90 (with a mean of 77) and all the aggregated scores highlights the periodic behavior of the user.

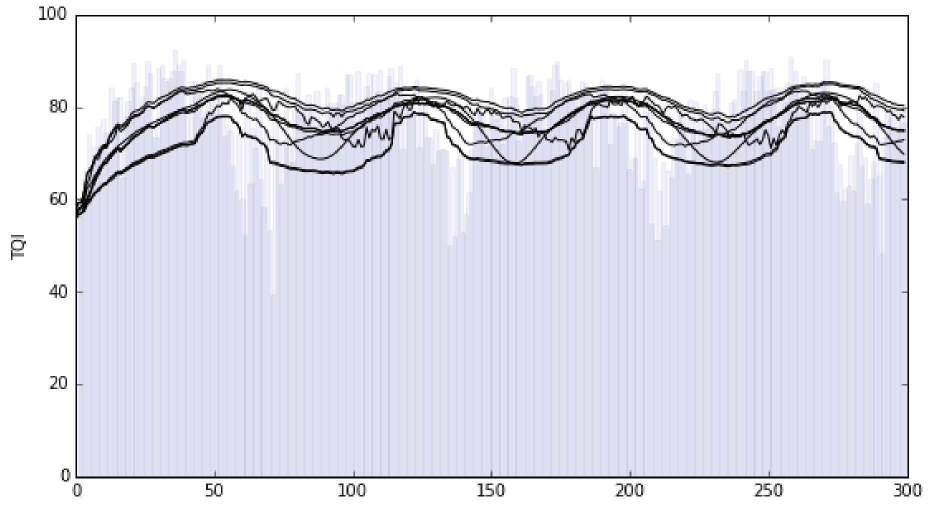


Figure 5.7: Bar plot of the TQI scores obtained by the O worker when he is aware of his average score, and the aggregated scores obtained with the selected operators.

What is interesting is the specifics of some of the aggregated scores. For instance, the OWA operator with $Q^e_{0.5}$ delays the increase of the aggregated score until a significant amount of work justifies such increase. Figure 5.8 depicts such aggregated score. Besides the wider local minimums of such metric, decrease of the aggregated score is not significantly delayed since the real decrease on TQIs. This position this particular OWA operator as a good candidate for measuring the *trust* of the worker, in which lower values of this metric may recommend the platform to put more emphasis on the evaluation of her work. Figure 5.9 compares such OWA operator with two weighted means. None of the weighted means were able to capture the idea

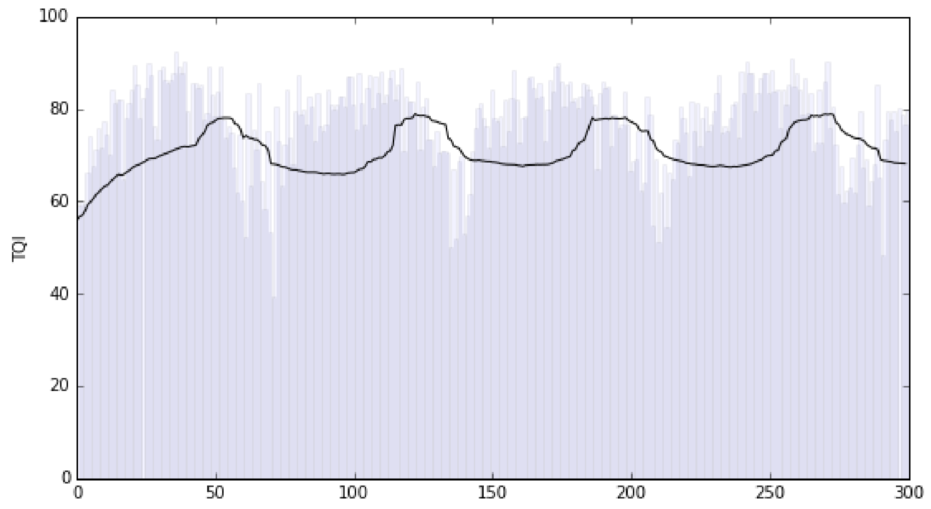


Figure 5.8: Bar plot of the TQI scores obtained by the O worker when he is aware of his average score. The aggregated score obtained with the OWA operator ($Q^e_{0.5}$) is also plotted.

of *trust* is the sense that it is difficult to obtain and easy to lose.

In the previous example, we have seen how the weighted mean ($Q^e_{0.5}$) provides an aggregated score which is resilient to discriminate worker for their latest work and the OWA operator ($Q^e_{0.5}$) provides this idea of trust that the worker must continuously prove their commitment in order to get a good score. In Figure 5.10 we depict how the worker O would have reacted in case that she is aware of such metrics. Notice that, when the weighted metric is considered, the worker learns that she can produce terrible outcomes after obtaining a good status on the platform. Such metric allows her to reduce her performance during large periods, as there is some delay until the metric

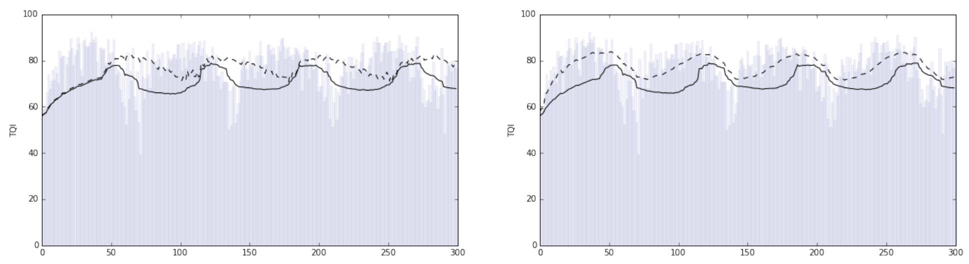


Figure 5.9: Bar plots of the TQI scores obtained by the O worker when he is aware of his average score. On the left plot, the OWA operator of Figure 5.8 is compared with the weighted mean ($Q^e_{0.5}$), whilst the right plot compares it with the weighted mean ($Q^e_{2.0}$).

penalizes such behavior. On the other hand, the *trust*-inspired OWA operator forces the worker to keep producing excellent work for some time until her status increases. Then, as the trust metric is drastically reduced after the first bad results, the periods of inefficiency are shorter.

It would be interesting to consider the effects of these metrics on the motivation of workers. Can we keep workers motivated if they do not see an increase of their rank after several good quality results? In any implementation of a crowdsourcing platform, one probably needs to consider a mixture of such metrics in which a non-trivial and slightly volatile aggregated score is provided to the users but then another internal metric is used for promoting or demoting the workers status.

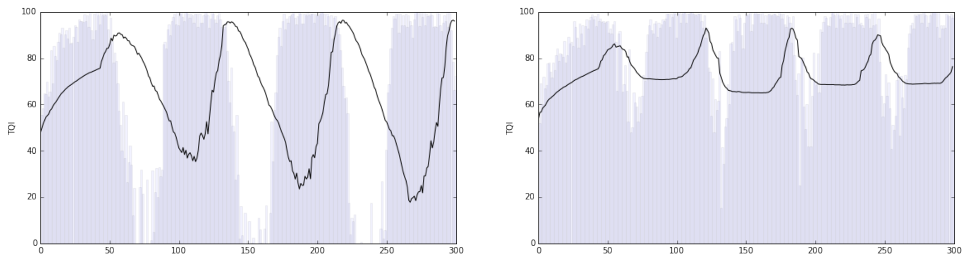
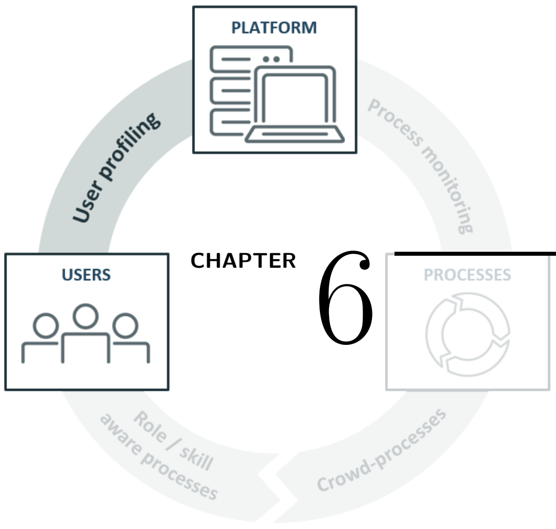


Figure 5.10: Bar plots of the TQI scores obtained by the O worker when he is aware of his score. On the left plot, the worker is aware of the weighted mean ($Q^e_{0.5}$), whilst the OWA operator is considered on the right plot. Both metrics are also plotted in their respective plots.

5.7 Discussion

We have studied the problem of combining past quality task evaluations using several aggregation functions to automatically determine worker category in a crowdsourcing platform holding a large professional community with different professional profiles. We have studied different cases to adjust the platform behaviour modifying the aggregation process (selected function and fuzzy measure). In this way, we can set an automatic management system for worker promotions and demotions. We have also introduced a real crowd-based platform for text translations, describing how our ideas can be deployed in it. It is still open to study how aggregation functions may determine the fairest economical reward for a given text, worker rank and final quality of the current translation.



CHAPTER 6 Evidence-based Worker Behavior Elicitation

In the previous chapter, we described a particular process that enable requesters to ensure overall quality of the problem resolution at the same time that ensures sustainability of the process by self-governing the promotion and demotion of the roles performed by the users. Nevertheless, there is no clear way of extrapolating such mechanism to other processes.

Here we propose to let the platform monitor the actions performed by individuals in order to create a profile of their behavior. We assume that those actions can be though as events that can be later processed by a discovery method, summarizing such actions in the form of a process model. Apart from the fitness of the resulting process models, precision is a key quality metric of these behavioral profiles. Low-precision models are more likely to describe the behavior of several users, reducing the insights obtained by analyzing or comparing process models. In particular, repetition of activities – very often due to the human nature – is one of the key trace characteristic that reduces precision of models discovered with most process mining techniques as we highlight, and palliate, during this chapter.

Later in Chapter 7, we will review how we used these models in order to automatically create groups of users with similar behavior. When applied to an industrial scenario, we have empirically seen that differences in these behavioral models are enough to group together workers with the same organizational role.

6.1 Introduction

Process discovery techniques strive to derive models that are expected to be good under four quality dimensions: fitness, precision, generalization and simplicity [19]. Hence, these are multi-objective techniques that search in a large solution space, where typically not one but many optimal solutions exist. In practice, each discovery technique puts the emphasis in a proper subset of dimensions; for instance, techniques based in the *theory of regions* focus on deriving fitting and precise models, while simplicity and generalization is not always guaranteed. Another example is the recent block-based techniques that recently appeared [63, 64], where structured, fitting, generalized and simple process models are preferred.

The techniques from [63, 64] are the driving force of this work. On the one hand, they are among the few scalable process discovery technique that can derive structured process models. This has made [63, 64] one of the most popular techniques for process discovery nowadays. However, as mentioned in [17], these techniques can sacrifice precision significantly for the sake of deriving a fitting structured model (see the example in Section 6.1.1). The alternative offered in [17] is to use evolutionary techniques, which are far from scalable. Instead, the technique proposed in this chapter represent a fresh look at this problem, amending (when possible) process models derived from the technique in [63, 64] as a simple post-processing step, based on unrolling loops in the model whenever the number of loop iterations found in the event log satisfy certain criteria. Next section illustrates the intuition behind the technique of this chapter.

Although the generation of more precise process models is a cross-domain challenge in the process discovery arena, we explored this problem as an opportunity for improving the description of the user behavior in a, possibly digital, platform. In this thesis, we are envisioning platforms that analyzes all actions performed by their users and generate a process model summarizing the behavior with two objectives: predict future actions of the users, and compare (or classify) users based on their behavior. For the first objective, highly-precise process models may not be the solution as they are usually lacking generalization on their models and, hence, the platform might be overlooking interesting options. Nevertheless, for the second objective, improving the precision lead to process models that express the uniqueness of the user and are less likely to describe others. One may then consider analyzing the event logs directly instead of a process model, but, then, we may fall into too specific characteristics hindering the comparison of different users.

It is not clear at which point it is not worth, or counterproductive, to increase the precision in behalf of generalization of a process model. And we may never be able to answer this question theoretically, as typically this depends on the later usage of the process model. It is important to have a mechanism to measure the utility, or objective, of the process models outside the traditional four quality metrics for process models: fitness, precision, generalization and simplicity. Then, the question would be how improving the precision of a process model affects to my overall objective. In our case, slightly improving precision helps in differentiating the users whilst enabling a meaningful and understandable comparison between them.

6.1.1 Label Splitting as Loop Unrolling to Improve Precision

Consider the model Figure 6.1.a, which was discovered by considering the trace $\sigma = ABCADCBACDACABCADCBACADE$. It is hard to notice that the precision of this model could be improved: Activities A, B and D can be found in any ordering and hence the parallel construct is appropriate, and trace σ hints that the iterative approach might be a good candidate for describing such a process. Nevertheless, a further analysis shows that there is still place for improvement.

In this chapter, we propose to *unroll* iterative parts of a process to check if there are hidden relations between the activities that are hindered by the limitation of only having one single copy of the activity in the model. See Figure 6.1.b for an example of such unrolling. In this particular case, we have chosen to repeat the iterative structure so we are forcing to execute its subprocess twice in each iteration. A replay of trace σ on this new process model highlights that activities B and D were never mutually exclusive. And hence, one could discover that the process model of Figure 6.1.c might be more precise in describing σ.

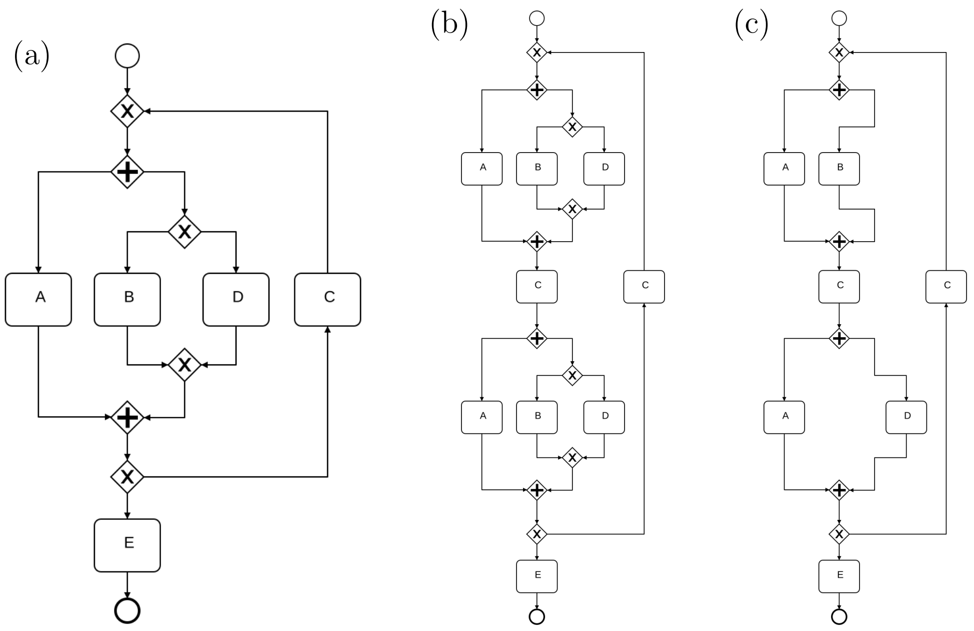


Figure 6.1: A first model accepting traces such as $\sigma = ABCADCBACDA$-$CABCADCBACADE$. The second model is an unrolled version that only accepts executing twice the initial iterative behaviour. Finally, a repair of the model with respect to the trace σ highlighted that the second choice construct could be simplified to a simple sequence.

6.2 Related work

Different approaches exist in the literature for the problem of label splitting in the context of process mining. We will focus here in recent approaches, and will illustrate the different nature of the technique of this chapter with respect to them. The heuristic techniques in [69, 104] rely on a window local search approach to define the duplication of certain candidate activities. This process is done in the model itself ([104]) or as a refinement of the input log ([69]). By focusing on the loops of a process model, the technique of this chapter complements these approaches.

Alternatively, global approaches can be found in [28, 84]. These global methods rely on the use of *unfolding* of the process model ([84]) and a later optimization technique to fold back activities, or search for special states of the underlying state space of the model ([28]), followed by a clustering strategy to merge them heuristically. By relying on complex representations and techniques (unfoldings or state spaces can be exponential on the size of the models), these approaches cannot be applicable for large inputs.

6.3 Definitions and Notation

Although the graphical notation used for representing processes is irrelevant in terms of the results presented in this chapter, the process tree notation (see Section 2.2.1.4 for more details) will be used in order to improve understandability, as some notions used in this Chapter are easily understood in the process tree notation.

Definition 6.1. *An **iterative subprocess** or loop l is the combination of two subprocesses that describe a process that can be repeated. The **forward path** of l (fwd(l)) is the subprocess that must be executed at least once during the execution l. Whereas the **backward path** of l (back(l)) is the subprocess such that its execution enforces the loop to re-execute fwd(l).*

From now on we will consider all process models to be structured. Importantly, structured processes allow us to map particular events in the trace to a subprocess in the process model. Allowing us to define the following notions:

Definition 6.2. *Given a process model N with a loop l and a trace $\sigma \in \mathcal{L}(N)$, we define $E_l(\sigma)$ as the number of times fwd(l) is executed during the execution of σ.*

Definition 6.3. *Let l be a loop of a process model N and σ a trace accepted by N. We define the **projection of** σ **to** $\mathbf{l}$ (denoted by $\sigma|_l$) as the result of keeping the events that are mapped into activities contained in l after a replay of σ in N. Moreover, we define the **projection of** σ **to the exit condition of** $\mathbf{l}$ (denoted by $\sigma|_{Exit(l)}$) as keeping the events that are mapped into activities of N that cannot coexist with the execution of l. In particular, all activities contained in l and any other concurrent activity are erased by this projection.*

Considering again the process model of Figure 6.1.a and trace $\sigma = AB$-$CADCBACDACABCADCBACADE$, we have a loop structure l consisting of: as the forward path, activities A, B and D that can be executed concurrently but B and D are mutually exclusive; and activity C as its backward path.

The forward path was executed $E_l(\sigma) = 8$ times; $\sigma|_l = \sigma - \{E\}$ whereas $\sigma|_{Exit(l)} = E$. Any execution of activity E clearly indicates that any event after event E is not part of the loop.

Definition 6.4. *(Fitness and precision) Process mining techniques aim at extracting from a log L a process model N with the goal to elicit the real unknown process $\mathcal{S}$. By relating the behaviors of L, $\mathcal{L}(N)$ and $\mathcal{S}$, particular concepts can be defined [19]. A process model N* fits *log L if $L \subseteq \mathcal{L}(N)$. A process model is* precise *in describing a log L if $\mathcal{L}(N)\backslash L$ is small.*

Unless stated otherwise, we assume we deal with fitting process models. In case this condition is violated, we assume the process models are first aligned with current techniques to satisfy the fitness condition [2].

6.4 Label Splitting with Loop Unrolling

Most discovery algorithms generate processes in which activities are not duplicated, forcing the algorithm to introduce loops when an activity is consistently occurring multiple times per trace. Unfortunately, this constraint may overgeneralize the resulting process model. Consider, for instance, trace $\sigma = ABCA$. A technique like the ones in [63, 64] will produce an iterative process model even though the trace is not showing so clearly that behavior.

First we will describe the unrolling algorithm for improving the precision of loops that are not included in any other iterative process. The main idea of this algorithm is to create a process model that forces the re-execution of the loop and then filters out unused elements. Finally, we will extend this algorithm for the case of nested loops.

6.4.1 Simple Case: Unrolling of Individual Loops

The first process model of Figure 6.1.a depicts a process describing the log consisting of the trace *ABCADCBACDACABCADCBACADE*. One may notice that, when replaying the log on the process model, the forward path (Activities A, B and D) is executed a multiple of two. Hence, we may force the process model to repeat the loop as in Figure 6.1.b. The unroll of a loop is precisely the process of making explicit this transformation.

Definition 6.5. *A k-unroll of a loop l is the process of substituting the loop for the subprocess defined by a loop structure with:*

- *A sequence of $k-1$ copies of the sequence fwd(l);back(l) as the forward path of the new loop structure,*
- *finishing the aforementioned sequence with another copy of fwd(l);*
- *The back(l) is maintained as the backward path of the new loop structure.*

In Figure 6.1.b, a 2-unroll of the iterative process is performed. In this case, the subprocess containing activities A, B and D is the forward path, whilst activity C is the backward path. And hence, its 2-unroll produces a loop structure with the sequence $AND(A\,OR(B,D))\,C\,AND(A\,OR(B,D))$ as the forward path and maintains C as the backward path.

Proposition 6.6. *Given a process model N describing the log L, a k-unrolling ($k>1$) of a loop l increases the precision of the model.*

Besides, if the process model fits log L and k is a divisor of the greatest common divisor (gcd) of the number of executions per trace of the forward path of l, then the k-unrolled process also fits log L.

Proof. Let l be a loop of N and let N_k be a k-unroll of l with $k > 1$. By construction of the k-unroll, we can ensure that any trace is an element of the language of N such that the forward path of l is executed a multiple of k times. I.e.

$$\mathcal{L}(N_k) = \{\sigma \in \mathcal{L}(N) \mid E_l(\sigma) \text{ is divisible by } k\}$$

We will show that $\mathcal{L}(N_k) \subsetneq \mathcal{L}(N)$ and hence, based on Definition 6.4, N_k improves the precision of process model N. Let σ' be a trace accepted by the process N that visits exactly once the forward path of the loop l, and hence the backward path of l is never visited. Since 1 is not a multiple of k, we can ensure that σ' is not an element of $\mathcal{L}(N_k)$ and hence $\mathcal{L}(N_k) \setminus L$ is a non-trivial subset of $\mathcal{L}(N) \setminus L$ and therefore the precision of N' is bigger than the precision of N.

Besides, let k be a divisor of the greatest common divisor of the number of executions per trace of the forward path of l, N a process model that fits log L and N_k the k-unroll of the process model N. Let σ' be a trace of the log L. Since N fits log L, the trace σ' is in the language of the process model. Moreover, the number of executions of the forward path of l is a multiple of k and hence the trace σ' is also an element of the language of L. Therefore, N_k fits log L. □ □

Once all loops have been unrolled, some activities and structures of the resulting process model may be redundant or unused and can be removed or

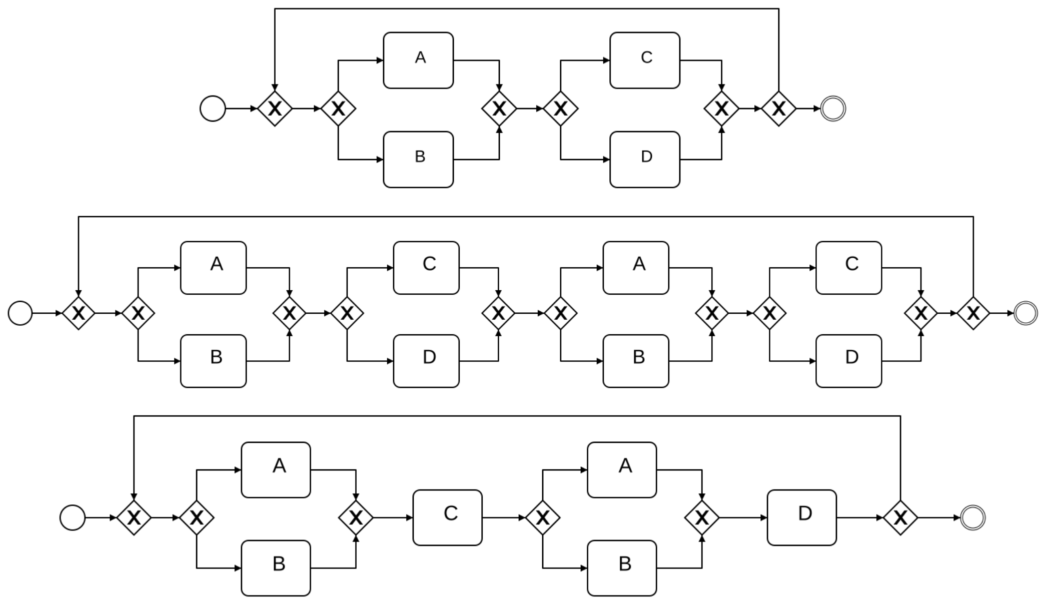


Figure 6.2: A process model describing the traces $ACADBCBD$, $BCBDACBD$ and $BCAD$.

simplified allowing for further improvement on the precision of the process model. The first process model of Figure 6.2 describes traces $ACADBCBD$, $BCBDACBD$ and $BCAD$. Such process may benefit from a 2-unroll as shown in the second process model. Besides, a replay of the three traces highlight that split choices between C and D are unnecessary: starting with C, activities C and D alternate in the execution of the process model. The last process model of Figure 6.2 depicts the process model after pruning unused paths.

6.4.2 General Case: Unrolling of Nested Loops

Structured subprocesses allow process models to have nested loops structures This poses a problem for deciding the number of unrolls, as the number of executions of the forward path per trace may be interleaved across embedded loops. The process model from Figure 6.3 depicts a process with a nested loop that accepts trace *ABBBABBB*. If we follow the count executions of the forward path, then activity B is recommended to be unrolled 6 times, even though it has never been executed 6 times in a row.

Instead of considering the number of executions per trace, we may count the number of consecutive executions of a forward path. In the particular case of trace *ABBBABBB*, the forward path B is consecutively executed 3 times at two different points in the trace, whilst the forward path consisting of activities A and B is consecutively executed 2 times. Definition 6.7 formalises this concept by counting the number of executions on maximal subtraces contained in the loop subprocess.

Definition 6.7. *Let l be a loop structure of the process model N, and σ a trace accepted by N. Then we define the* ***set of continuous executions*** *of the loop l in the trace σ as*

$$CE_l(\sigma) = \left\{ n \;\middle|\; \exists\, \sigma_1, \sigma', \sigma_2 \text{ s.t. } \begin{array}{c} \sigma = \sigma_1 \sigma' \sigma_2 \\ \sigma'|_l \in \mathcal{L}(l) \\ \sigma'|_{Exit(l)} = \emptyset \\ E_l(\sigma'|_l) = n \\ \sigma' \text{ is maximal} \end{array} \right\}$$

Informally, the set $CE_l(\sigma)$ represents a set of numbers, each one denoting continuous executions of l in σ.

The combination of *no exit condition* and maximality of subtrace σ' in Definition 6.7 ensures that we are splitting the trace σ on chunks such that a continuous execution of the loop l is not separated in two different subtraces. Besides, non-consecutive executions of the loop cannot be included in the same group as this would have shown some activities incompatible with the execution of the loop. Notice that activities that are executed concurrently alongside the iterative subprocess l might be included in the subtrace σ', but they are removed during the projection to the iterative subprocess l and they are not part of the *exit condition*.

Consider trace $\sigma = ABBBABBB$ and the smaller loop, or B-loop, of process model 6.3. The exit condition of the B-loop is activity A, since any execution of that particular activity clearly shows that the execution is happening outside the B-loop. Hence, we may split σ in two instances of $\sigma' = BBB$. Notice that we cannot extend it since then we would include an exit condition, and σ' is accepted by the B-loop. And therefore, $CE_{B-loop}(\sigma) = 3$.

Similarly to the non-nested case, the language accepted by an unrolled process model can be described as a refinement on the language accepted by the original process model as depicted in Proposition 6.8.

Proposition 6.8. *Let N be a process model, and l a loop subprocess of N. Let N_k be any k-unroll of l. Then*

$$\mathcal{L}(N_k) = \{\sigma \in \mathcal{L}(N) \,|\, \forall n \in CE_l(\sigma), n \textit{ is divisible by } k\}$$

Proof. The definition of the k-unroll of loop l ensures that any execution of the loop l executed a multiply of k times the forward path of l and hence any maximal subtrace $\sigma' \subseteq \sigma$ such that $\sigma'|_l \in \mathcal{L}(l)$, $\sigma'|_{Exit(l)} = \emptyset$ must satisfy that k divides $E_l(\sigma')$.

On the other hand, let σ be a trace in $\mathcal{L}(N)$ such that all continuous executions $CE_l(\sigma)$ are divisible by k. Then, σ is also an element of the language $\mathcal{L}(N_k)$. Suppose not, then the trace σ violates any behavioural relation between a set of activities or the iterative process must finish before repeating k times the forward path. Both cases are not possible. The former violates the fact that $\sigma \in \mathcal{L}(N)$ and the latter violates the fact that $CE_l(\sigma)$ only contains multiples of k. $\square$

Proposition 6.9 is a direct consequence of Proposition 6.8, due to the hard constraint that k divides all $n \in CE(t, l)$.

Proposition 6.9 (Generalization of Proposition 6.6). *Given a process model N describing the log L, a k-unrolling ($k > 1$) of a loop l increases the precision of the process model. Besides, if the process model fits log L and k is a divisor of $CL(l, t)$ for all $t \in \mathcal{L}(N)$, then the k-unrolled process model also fits log L.*

Revisiting the example of trace *ABBBABBB* and the process model of Figure 6.3, which contains a nested loop, a replay of the trace contemplates that the big loop is executed 2 times and the smaller loop is executed 3 times on each execution. Hence, we could perform a 3-unroll on the latter and a 2-unroll on the former. Doing so, we discover the second process model of Figure 6.3, and a second replay highlights the possibility of removing the unnecessary loop structures as illustrated by Figure 6.3.

Proposition 6.9 establishes a necessary condition over k in order to main-

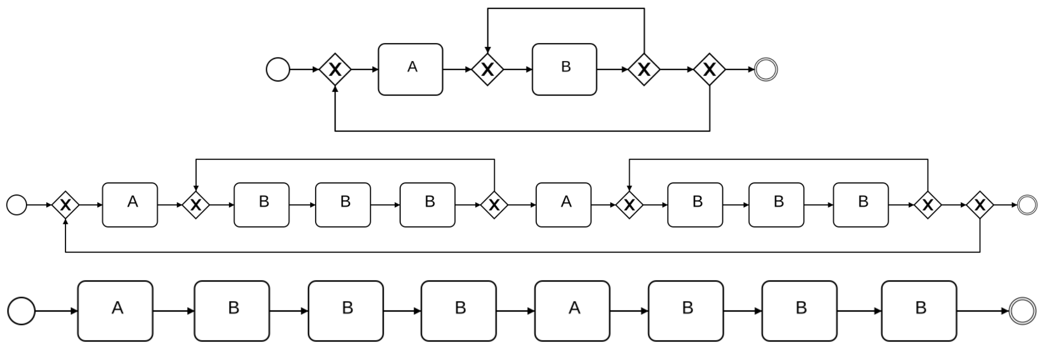


Figure 6.3: Three process models describing the trace *ABBBABBB*.

tain the fitness of the k-unroll. Still, practitioners may sacrifice replayability of some traces by choosing a different k. See Table 6.1 for a simple log in which not choosing the *right* k could be a better option. Proposition 6.9 would recommend to choose $k = 1$, as the greatest common divisor of 3 and 4 is 1. Nevertheless, there is only one trace t with 3 continuous executions of the iterative subprocess. By choosing $k = 2$, the resulting model would accept all traces except that special trace t, and, hence, the k-unroll improves the precision of the process model while replayability of the log is not significantly impacted. Still, we believe that the choice of k in this case should be analyzed by the practitioner in order to avoid a loss of significant behavior.

6.5 Evaluation

Using an existing dataset compromising 15 small logs [104] whose source processes are well-known and reproduce behavior commonly found in real-life scenario, we plan to evaluate the gain on precision in comparison with a state

Continuous executions	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**≥ 7**
Absolute frequency	30	0	0	1	45	0	12	0

Table 6.1: Example of the number of continuous executions of a subprocess l in a log L. Besides the number of executions, this table also counts the number of traces with such number of executions. For instance, only one trace has 3 consecutive executions of the loop l.

of the art technique for label splitting [28]. The behavioral characteristics of the considered process models are summarized in Table 6.2. Afterwards, we discuss some experiments with pairs of process models and event logs in which the optimal k provided by Proposition 6.9 is 1, and hence setting a bigger k provides a more precise process model but at the cost of losing fitness.

Table 6.3 contains the precision obtained with the process model discovered with Inductive Miner (IM), the precision obtained after unrolling these process models and precision obtained by PNSimpl [28]. We have used the alignment-based precision metric [1] for this evaluation. 7 out of 15 processes do not show any improvements with our technique since it was not possible to perform an unroll without losing fitness. For the BPI Challenge 2012 log, it was also not possible to perform an unrolling without losing fitness. None of the iterative subprocesses was repeated a multiple of k times for any k. Nevertheless, for this dataset we followed another strategy: We choose k in order to minimize the loss in fitness. In this particular case, after performing

Model	**Activities**	**Sequential**	**Choice**	**Concurrency**	**Self loops**	**Loops without backward**	**Loops**
alpha	11	x	x	x			x
betaSimpl	13	x	x	x	x		
Fig5p1AND	5	x		x			x
Fig5p1OR	5	x	x				x
Fig5p19	8	x	x	x	x		
Fig6p9	7	x	x	x	x		x
Fig6p10	9	x	x	x		x	
Fig6p25	21	x	x			x	x
Fig6p33	9	x	x				x
Fig6p34	10	x	x	x		x	
Fig6p38	12	x		x	x		
Fig6p39	7	x	x	x		x	
Fig6p42	14	x	x	x		x	
RelProc	16	x	x			x	x

Table 6.2: Summary of the characteristics of the process models considered in the evaluation. *Self loops* reefer to activities that can be executed again immediately after its execution. *Loops without backward* are those loops compromising more than one activity such that their backward path is empty, whilst *Loops* consider those loops with a non-trivial backward path.

	Inductive Miner (IM)					**IM + Unrolling**					**PNSimpl**
Model	P	T	τ	Precision	Fitness	P	T	τ	Precision	Fitness	Precision
alpha	11	17	6	0.6750	1.0	12	19	3	**0.7386**	1.0	0.70
betaSimpl	14	21	8	0.6216	1.0	14	15	2	0.9130	1.0	**0.93**
Fig5p1AND	9	8	3	0.8333	1.0	10	8	2	**1.0000**	1.0	**1.00**
Fig5p1OR	5	6	1	0.7000	1.0	6	6	0	**1.0000**	1.0	**1.00**
Fig5p19	9	14	6	0.6746	1.0	Not applicable					**0.85**
Fig6p9	10	15	8	0.6667	1.0	Not applicable					**0.83**
Fig6p10	15	24	13	0.6282	1.0	Not applicable					**0.76**
Fig6p25	22	35	14	0.7629	1.0	25	36	13	**0.8467**	1.0	**0.84**
Fig6p31	6	10	1	0.6250	1.0	7	10	0	0.90	1.0	**1.00**
Fig6p33	7	11	1	0.6667	1.0	8	11	0	0.90	1.0	**1.00**
Fig6p34	17	24	12	0.5785	1.0	Not applicable					**0.93**
Fig6p38	13	11	4	0.6212	1.0	13	10	2	**0.77**	1.0	0.65
Fig6p39	12	12	5	**0.8986**	1.0	Not applicable					**0.89**
Fig6p42	7	18	4	0.2284	1.0	Not applicable					**0.74**
RelProc	21	28	12	0.7143	1.0	Not applicable					**0.73**

Table 6.3: Comparison of the precision in selected process models discovered with Inductive Miner and then Unrolled. The simplicity of the processes is also depicted with the number of places P, transitions T and silent activities τ in the discovered Petri Nets. For some cases the unroll is not possible without sacrificing fitness.

a 2-unrolling on the activity *Calling to add missing information to the application*, 9% of the traces cannot be replayed by the unrolled process model, with a minimal impact on fitness, but its precision increases 5%.

Results indicate that precision gain is similar with both techniques, provided that unrolling is possible. Nevertheless, both approaches treat the initial process model differently: Our approach enhances the expressive power of the initial process, whilst PNSimpl rediscover the process after each label split and, hence, the final process might be significantly different. In terms of complexity, the technique of this chapter may be a light alternative for methods like [28], which require to iteratively apply agglomerative clustering for special sets in the state-space representation of the event log.

Non-optimal choices of the k-Unrolling After such experiment, we wanted to test if the usage of non-optimal k gives value to the k-unrolled process model. We considered the BPI Challenge 2012 dataset as first example. This real-life log contains events describing the application process for a personal loan, or overdraft, within a global financing organization. From all these events, we have only selected the events starting with W as they show actions performed by workers of the organization, which are more prone to have repeated activities. For this dataset we followed another strategy: We choose $k > 1$ in order to minimize the loss in fitness. After performing a 2-unrolling on the activity *Calling to add missing information to the application*, 9% of the traces cannot be replayed by the unrolled process model, with a minimal impact on fitness, but its precision increases 5%. Besides, this results implies that the financing organization of the BPI Challenge 2012 usually had to call customers an even number of times for getting the necessary information. Why are odd repetitions too sparse?

Besides those cases in which a very small number of traces do not sat-

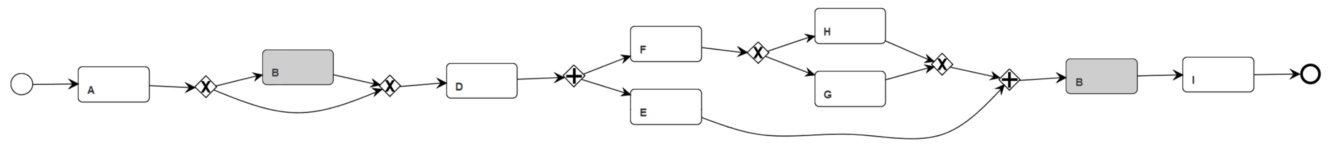


Figure 6.4: Example of a process model in which an activity, with grey background, may be executed twice. The first execution is completely optional, whilst the second execution is mandatory.

isfy the multiplicity by $k > 1$, we believe that one meaningful case in which we have to manually set k is when at least one repetition of the activity is optional. For measuring the utility in such a case, we manually designed a process model in which one activity is optionally repeated with some activities in-between. See Figure 6.4 for the example we used in this small experiment. We used such process model to generate a log in which 50% of the traces only executed the activity once, and it is executed twice in the rest of the event log. Figure 6.5 depicts the process model discovered with the Inductive Miner. In this model, the grey activity is depicted as a self-loop and performing a 2-unrolling has a large impact on the replayability of the event log, even though it will more close to the original behavior. Figure 6.6 depicts the 2-unrolling of the process model after a model repair has been performed. One may notice that the second execution of the grey activity is now optional, thanks to the process model repair.

What we have seen in the repaired 2-unrolling of the process model in Figures 6.5 and 6.6 is that the discovered model has some hidden non-trivial relationship between activities. For instance, we have seen that some activities between the two executions of the grey activity have been duplicated in the process model and have both been depicted as optional. Nevertheless, skipping one of these blocks of activities depends on the number of executions of the grey activity. Besides, we have also seen that concurrent activities with any of the two grey activities produce process models in which the same concurrency covers the execution of both repetitions. In the future, we will need to consider a *repairing* method that places these concurrent activities next to their respective activity repetition. These two insights are consistent with

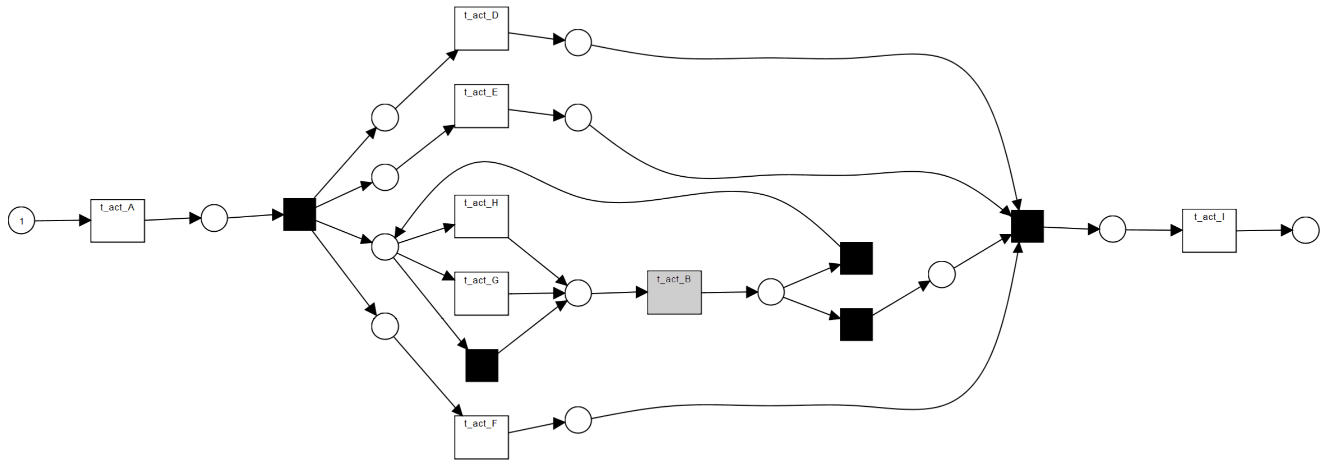


Figure 6.5: Example of a process model in which an activity, with grey background, may be executed twice. This process model has been discovered using the Inductive Miner over an event log generated by the process model in Figure 6.4.

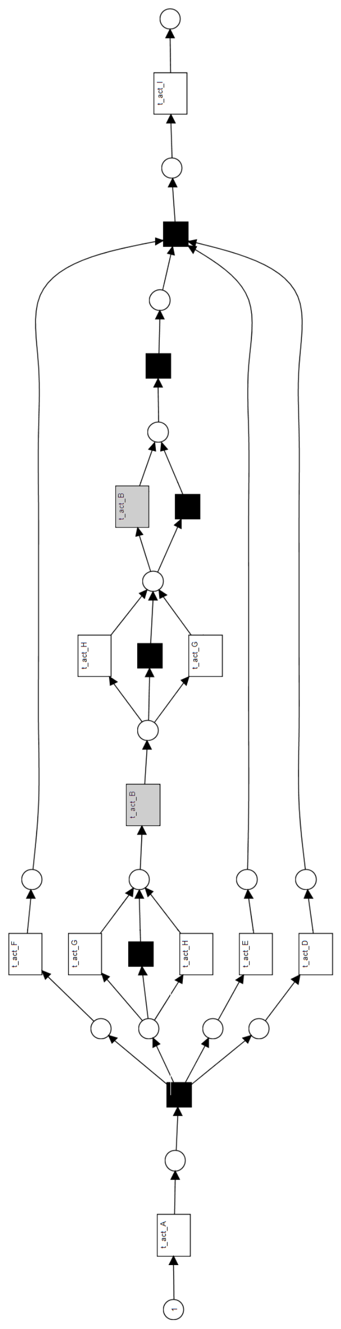


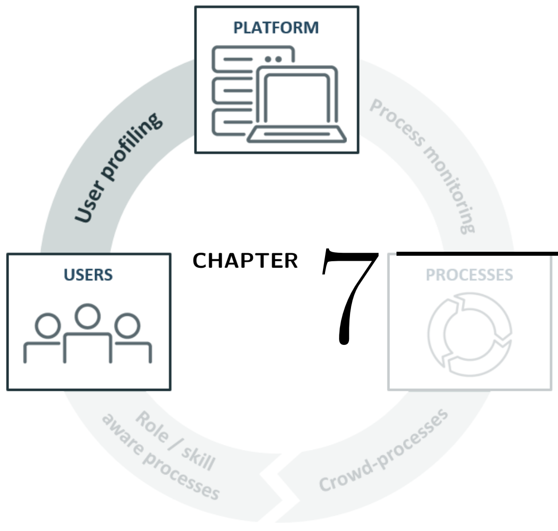
Figure 6.6: Process model discovered by repairing the 2-unrolling of the Process Model depicted in Figure 6.5.

the rest of the process models of this evaluation, as well as in other small examples we have been testing while preparing this thesis.

6.6 Conclusion

In this chapter, we presented a method for improving the precision of structural subprocesses based on explicitly repeating iterative subprocesses and pruning unused constructs and activities. We have shown that this approach is applicable to simulations of real-life processes, and also it is applicable to real-life scenarios.

The presented approach is the first step on considering the unrolling of iterative processes. Results in Table 6.3 show several examples of how unrolling improve the precision of the process models, with minimal impact on their complexity. Nevertheless, bigger process models might be more difficult to understand and, hence, it remains to conduct expert reviews on readability and understandability of process models after unrolling. Besides, we have experienced on some datasets that some iterative processes can be explained as a few iterations are used for initialization, and then the real loop starts. We would also like to study how the k-unroll operation affects the precision of the process model for a particular precision metric. In particular, is it possible to establish a lower bound on the increase of the precision?



CHAPTER 7 Process Model Comparison Based on Cophenetic Distance

Continuing the idea of considering process models as representation and summary of the user's behavior in the platform, we describe a new similarity metric between process models in this Chapter. Being able to compare between process models describing human behavior would enable us to consider finding groups of humans that work on a similar manner, or detecting those individuals whose behavior significantly differs from their peers. In particular, we have applied these techniques with an industrial dataset compromising several workers with access to a source code repository. It turns out that their role in the organization is partially seen in how they access such source code repository.

This Chapter closes the monitoring capabilities of the platform with respect to workers. We have seen that some processes demand a more sophisticated user categorization, or profiling, in order to achieve the desired level of quality. We have proposed a small pattern to assess the acquisition of skills, which may be used for defining and sustaining the just mentioned categorization. In those cases in which is not possible to use that particular pattern, the solution presented in this chapter may be used to measure how similar is a particular worker to a selected set of workers which have already shown the desired property (either quality or skills).

7.1 Introduction

Nowadays process models are ubiquitous objects in companies and organizations. They are becoming precious for representing unambiguous and detailed descriptions of real processes. On the one hand, *BPMS* platforms, which allow designing, deploying and managing the processes in organizations, are based on process models. On the other hand, evidence-based process models (i.e., process models with a high alignment with respect to the underlying real process) can be used to analyze the process formally, e.g., detecting inconsistencies or performance problems that may hamper the correct and optimal execution of the process. Furthermore, the existence of environments for creating, managing and querying *process model collections* enable the hierarchical and cross-organizational analysis, with process models as atomic objects.

A core technique necessary in many of the aforementioned situations is the automated comparison of process models. Due to its importance, this problem has received significant attention in the BPM field, which can be split into *structural techniques* based on graph-edit distance [30, 31, 32, 112], and *behavioural techniques* that focus on the execution semantics or behavioural relations of the corresponding models [7, 29, 83, 107, 108]. Intuitively, structural techniques are fast but inaccurate (in terms of the differences found), whereas pure behavioural techniques are complex (both in computation time and memory usage) but accurate.

In this section we propose a novel method to compare process models[1]. The technique is based on a recent algorithm [23] from the field of *computational phylogenetics*, where the objects to compare are labeled trees showing the inferred evolutionary relationships among various biological species. We adapt the algorithm to the BPM context, thus using *process trees* [18] as notation. Our proposed similarity metric sits halfway between pure structural similarity methods (inheriting their low complexity features), and behavioural similarity metrics (capable of providing similar behavioural information). Moreover, the performance of our approach allows us to consider this metric for large process models.

Motivating Example

Let us use a real-life example to motivate the contributions of this chapter. A product manager decides to monitor all accesses to an SVN repository. Apache Subversion (SVN) is a software versioning and revision control system. Software developers use SVN software to collaborate in the maintenance and development of software, by monitoring changes in files such as source code, web pages and documentation. All accesses to a SVN repository are done through HTTP/S, as specified in the WebDAV/DeltaV protocol. It

[1]We assume the problem of dealing with real activity labels, e.g., when the name of an activity in the models does not perfectly match, is resolved prior to the techniques of this chapter.

turns out [92] that those read and write requests over HTTP/S can be translated to human-friendly SVN commands such as *svn update* or *svn commit*. Continuing the work done by Li Sun et.al. [92], we model the behaviour of developers by using process discovery techniques. In particular, we discovered Petri Nets with the default settings of the Inductive Miner Plugin (ProM 6.5.1). Our goal is to measure the differences between those models, inducing a behavioural distance between individuals. This way, intruder attacks to the repository can be detected globally by analyzing process behaviour that is clearly separated from the rest.

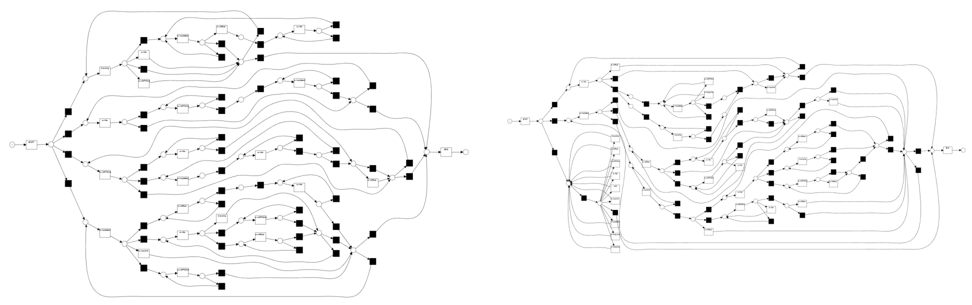


Figure 7.1: Two process models describing how two users access an SVN repository.

Figure 7.1 depicts the access behaviour of two users to the same repository, using a block-structured process discovery algorithm[2]. In our preliminary study, the process model of an average user shows lots of concurrency, duplicate activities and iterative behaviour[3]. Existing behavioural compari-

[2]We used *Discover a Process Tree using Inductive Miner (ProM 6.5)* and then converted them to Petri Nets.

[3]The most common sequence of commands in the dataset is *svn -options*, *svn update*, *svn -options* indicating they use an IDE that overwrites the SVN options just to perform an update and then returns to its previous status.

son techniques struggle when dealing with such models. Either they fall short in describing duplicate activities and loops [107], or the underlying technique does not scale in the presence of concurrent process branches [7].

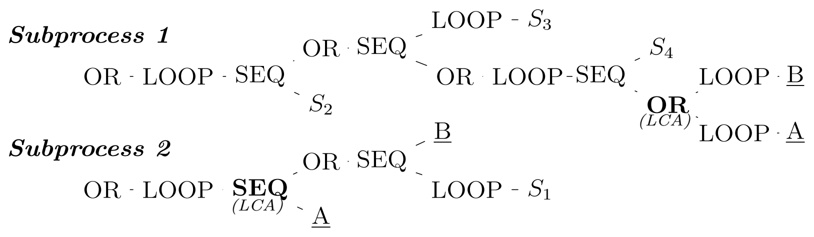


Figure 7.2: Extract of the tree representation of the two processes in Figure 7.1. Only the subtrees related to two common activities A and B are represented, and their least common ancestors are depicted in bold. Activities S_i are unrelated to A and B.

The approach presented in this chapter evaluates the difference of the two minimum subtrees containing a selected pair of activities, and extends the comparison to all possible pairs. Analysis over such subtrees is expected to be more simple and efficient, while still capable of comparing both the structure and behaviour of the two processes. See Figure 7.2 for an example, which focuses on activities A and B in both models. One can check that the difference between the depth of the two activities is an approximation to the graph distance between those two models. For instance, in Figure 7.2, depths of A are 11 in the first subprocess and 4 in the second subprocess, whilst depths of B are 11 and 6. The difference of their depths sum 12, implying that 8 nodes must be removed and 2 extra edges are needed in order to transform one model into the other. Besides, and more importantly, one can see that the common ancestors of activities A and B in Figure 7.2 model two different behaviours: On the first subprocess, activities A and B are mutually exclusive; On the second, A is executed after activity B. Notice that the depth of this common ancestor also highlights how long it takes to make the *behavioural decision* of how activities A and B relate to each other. Therefore, by incorporating these notions into the distance function, we would be able to not only measure structural differences but also highlight differences in the behaviour of two process models. For instance, one could obtain the sentence: *Activities A and B in Figure 7.2 are mutually exclusive in the first subprocess, but activity A always occurs after B in the second subprocess. Besides, the behavioural decision in the first subprocess is done* 6 *steps after the decision is taken in the second subprocess.*

7.2 Related Work

The state of the art techniques for comparing process models can be split into structural and behavioural. In the former, process models are considered as labeled graphs and the comparison is regarded as edit operations over their edges, nodes or both. In contrast, behavioural techniques focus at the comparison of the execution semantics of the compared models.

If we focus on the particular case of business process models, structural comparison techniques based on graph edit operations have been defined [30, 31, 32, 112]. Although the graph edit distance technique is NP-complete, the aforementioned techniques use heuristics that can cope with the inherent complexity of the problem in practice. Also, structural techniques based on *similarity flooding* represent an alternative to these methods [71].

Analogously to structural techniques, behavioural ones exist for the case of business process models. The first ones considered the comparison of the automaton underlying the models [29], although the techniques proposed in [29] are not complete. *behavioural profiles* is another way of comparing the behaviour of process models. A behavioural profile of a process is represented as an $n \times n$ matrix, where n is the number of tasks in the process [107, 108]. Each cell in the matrix contains one of three possible ordering relations, plus an additional co-occurrence relation in case of causal behavioural profiles. Unfortunately, this way of representing a process model falls short in describing several constructs, like skipping/duplicate activities and loops. These problems also affect a similar formalism that recently appeared [83].

The recent work in [7] represents a complete and fresh approach to the problem of behavioural model comparison. It regards the problem of diagnosing differences between process model as the analysis of the synchronous product of the corresponding underlying *event structures*, a partial-order representation of the behaviour. Interestingly, it can describe differences using natural language templates that are more accessible to the general audience. Although both canonical and reduced representations of event structures can be used, these optimized representations only help to alleviate partially the complexity of the approach, as it is shown in the experiments of this paper. of dealing with the behavioural representations in terms of event structures.

7.3 Background

The metric defined on this Chapter is heavily based on the structureness of a process model, and its process tree representation. Apart from the essential notations and definitions of Section 2.2.1.4, here we define the cophenetic vectors, which hold a key property for comparing trees that will be later used in Section 7.4.

7.3.1 Cophenetic Vectors

The **least common ancestor** (LCA) of a pair of nodes u and v of a rooted tree T, denoted by $[u,v]_T$, is the unique common ancestor of them that is a descendant of every other common ancestor. The definition of the Cophenetic vector is based on the discrepancies on the depth of the LCA of every pair of activities.

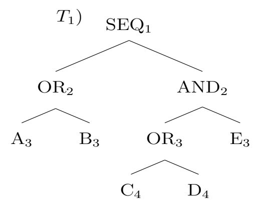
Definition 7.1 ([86])**.** *Let S be the set of labels of a weighted labeled rooted tree T. For every pair of different labels i, j, their Cophenetic value is*

$$\varphi_T(i,j) = \delta_T([u,v]_T) \qquad u,v \textit{ have labels } i,j$$

To simplify notation, we denote the depth of a node with label i by $\varphi_T(i,i)$, and $\varphi_T(i,j) = 0$ if either i or j are not activities of the process tree T.

Definition 7.2. *Let T be a weighted rooted tree, and S the set of activity labels of the tree T, its **Cophenetic vector** is*

$$\varphi(T) = (\varphi_T(i,j))_{i,j\in S}$$



	A	B	C	D	E
A	3	2	1	1	1
B		3	1	1	1
C			4	**3**	**2**
D				**4**	2
E					**3**

	A	B	C	D	E
A	3	2	1	1	1
B		3	1	1	1
C			4	**2**	**3**
D				**3**	2
E					**4**

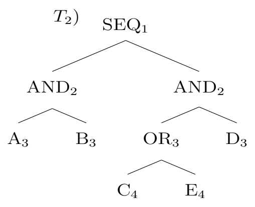


Figure 7.3: Example of process trees and their Cophenetic vector (in matrix representation), assuming the depth of the root is 1. For simplicity, we included node's depth as a subscript of the label. For instance, the LCA of activities C and E in T_1 is the AND gateway that is one children of the root and, hence, its Cophenetic value is 2.

In an already fifty years old paper [86], Sokal and Rohlf proposed the use of the cophenetic values to compare dendrograms. Authors in [23] show that cophenetic values can also be applied to uniquely project labelled trees into a multidimensional vector space, allowing them to define a distance on labelled trees as Theorem 7.3 states.

Theorem 7.3 ([23]). *Two weighted labeled trees without elementary nodes, unlabeled leaves nor repeated labels are equal if, and only if, they share the same Cophenetic vector.*

Cophenetic vectors are not enough for determining process tree similarity: for instance, in Figure 7.3 if the *OR* and *AND* labels of the left tree are interchanged, the Cophenetic vectors of both trees are equal whilst the behaviour represented is different. Besides, constraints in Theorem 7.3 do not allow models with multiple silent transitions. Next section shows how to transform process trees in order to overcome this limitation.

7.4 The Cophenetic Distance between Deterministic Process Trees

As we have seen, Cophenetic values unequivocally represents weighted labeled rooted trees. As it is well known, this allows to induce distance metrics in the set of labeled trees. Let *dist* be any distance between two points in a vector space, we define

$$d(T, T') = dist(\varphi(T), \varphi(T'))$$

as the distance between two trees. For instance, by using the L^1-norm we get

$$d_1(T, T') = \sum_{i,j \in S} |\varphi_T(i,j) - \varphi_{T'}(i,j)|$$

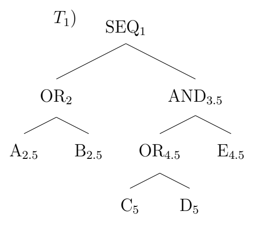
The Cophenetic values were originally conceived to measure structural differences between the leaves of two dendrograms, but we can extend its use to deterministic process trees thanks to Theorem 7.3. This result allow us to modify the depth of each node in order to model the path of gateways we are tracing from the root to activities (the leaves of the tree). In Definition 7.4 we propose a depth function to overcome the following weaknesses of the original Cophenetic distance over labelled trees: (1) ensures that non-common activities increase the distance between two models; (2) depth of activities in a sequential order increase in the same sequential order, modeling the complexity of the blocks already seen by the process; (3) allows for silent transitions; and (4) differentiates two processes with the same structure but modeling different gateways at the root.

Definition 7.4. *Let T be a deterministic process tree. We define the depth function δ'_T as follows:*

1. *Root node has depth* 1.
2. *Iterate over all nodes in a pre-order traversal.*
3. *The depth of all nodes is* 1 *plus the depth of its parent, except*
   - (a) *If the parent is an OR clause, increase* 0.5 *instead of* 1.
   - (b) *If the activity is silent, increase* 0.25 *the depth of the parent and any other sibling. Afterwards, remove the silent activity.*
   - (c) *If the parent is the start of a LOOP, increase also by the maximum depth of the underlying tree.*
   - (d) *If the parent is a SEQ gateway, consider the depth of deepest visited children of the node's siblings instead of the parent.*
4. *Any remaining elementary node will be removed, and its parent and children will be directly connected.*

For the sake of simplicity, $trf(T)$ will denote the combination of the tree T with the aforementioned depth function δ_T.

Figure 7.4 depicts the transformation of the two processes in Figure 7.3. With the aforementioned depth function, Cophenetic values now highlight, for example, differences in the two activities A and B due to the behavioural change of their parent node. This transformation allow us to overcome the limitations of Theorem 7.3, since silent transitions are allowed, but also by



	A	B	C	D	E
A	**2.5**	2	1	1	1
B		**2.5**	1	1	1
C			**5**	**4.5**	**3.5**
D				5	**3.5**
E					**4.5**

	A	B	C	D	E
A	**3**	2	1	1	1
B		**3**	1	1	1
C			**5.5**	**4**	**5**
D				5	**4**
E					**5.5**

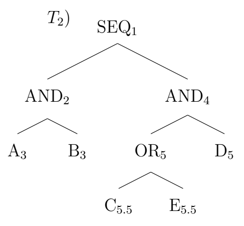


Figure 7.4: Transformation of the process trees in Figure 7.3. For the sake of simplicity, we included node's depth as a subscript of the label. For instance, depth of the *AND* gateway in T_1 is 3.5 because its parent represents a sequence and the maximum depth of the previous processed branch is 2.5

ordering children of sequential gateways. As we state in Theorem 7.5, this transformation uniquely represents deterministic process trees.

Theorem 7.5. *Let T and T' be two deterministic process trees. If $trf(T)$ and $trf(T')$ share the same Cophenetic vector, then T and T' are the same process tree.*

This theorem shows that Theorem 7.3 is also applicable to the new depth definition, and therefore useful for checking equality of two process trees and measuring differences between the models. The proof of this theorem is based on the observation that the Cophenetic values of any subtree are highly related to the Cophenetic values of the complete tree, as Lemma 7.6 shows. Details of the proof of this lemma are omitted, but it is a direct consequence of the pre-order traversal approach of Definition 7.4.

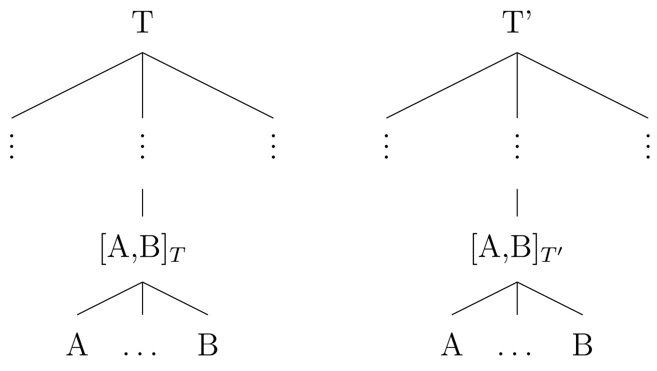
Lemma 7.6. *Let T be a weighted rooted tree, and S a subtree of T. Then* *the Cophenetic vector of S satisfies that*

$$\varphi_S(i,j) = \varphi_T(i,j) - \delta'_T(\textit{root of } S) + 1$$

Theorem 7.5. Let's proof this by induction.

- For processes with 1 or 2 activities, one can list all possible deterministic process trees and check that no two processes share the same transformed tree.

- For processes with $n > 2$ activities, we will show that every strict subtree[4] of T is equal to another subtree of T'. Let VT be a strict subtree of T. Suppose A and B are two activities such that $[A,B]_T$ is the root of VT. Activities A and B are also included in the deterministic process tree T', and $[A,B]_{T'}$ is the root of a certain subtree VT'.



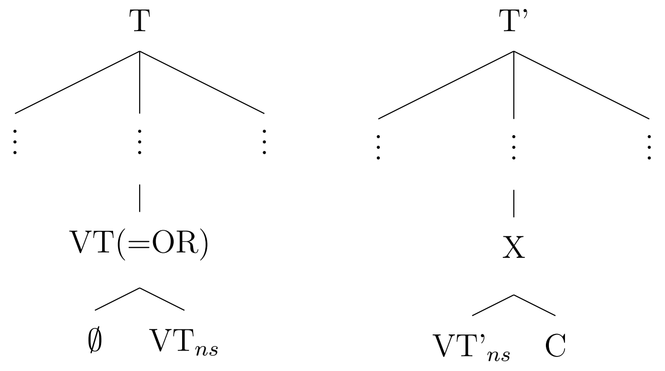
Lemma 7.6 ensures that

$$\begin{aligned}\varphi_{VT}(i,j) &= \varphi_T(i,j) - \delta'_T([A,B]_T) + 1\\ &= \varphi_{T'}(i,j) - \delta'_T([A,B]_{T'}) + 1 = \varphi_{VT'}(i,j)\end{aligned}$$

[4]Here a strict subtree of T is any subtree that does not contain the root of T

where the second equality holds since $trf(T) = trf(T')$ and Theorem 7.3. And therefore, VT and VT' share the same Cophenetic vector and its size is smaller than T and T'. By induction, we can say that both process trees are equal.

There is one case where there are no two activities A and B such that $[A, B]_T$ is the root of VT: The root of VT is an OR-clause, and one children is a silent transition. In this particular case, we can work with the non-silent children VT_{ns}. The combination of two consecutive *OR* conditions is not possible in a valid deterministic process tree, and therefore VT_{ns} falls under the proved assumption. Hence, there is a subtree VT'_{ns} of T' that is equal to VT_{ns}.



VT_{ns} and VT'_{ns} share the same activities, and VT_{ns} is a strict subtree of T. Therefore, VT'_{ns} is also a strict subtree of T'. Let X be its parent node. We will show that X is in fact an *OR* condition, and it only has another silent branch. Let's assume there exists an activity C under X but not included in VT'_{ns}. There are two options:

– X is the root of T'. In that case, we can replace the subtrees VT_{ns}

and VT'_{ns} by a mock activity C'. We reduced the problem to the 2 activities case, already solved. In that case, we share the same Cophenetic value but the two process trees are different (T' does not have a silent transition). We arrived to this contradiction by assuming that C exists.

– X is not the root of T'. In that case, the subtree VX induced by the node X is a strict subtree of T' and X is not and OR condition. By applying the previous reasoning, there is a subtree W of T that is equal to VX and includes VT_{ns}. Notice that, in that case, the only possibility is that C is a silent transition.

This shows that any subtree of T is equal to a certain subtree of T'. By applying this result to all the direct children of the root of T one can see that T and T' are indeed equal.

□

7.4.1 Behavioural Information Captured by Cophenetic Values

The syntax of process trees allow us to easily check the **direct causality** of two activities in the model: one simply needs to check the behaviour explained by their LCA. Co-occurrence of activities is described by an AND gateway, whilst OR internal nodes induce conflict between their underlying activities. Notice that this causal relation is a property for the minimum subtree containing the pair of activities. For instance, if the two activities are inside a bigger loop structure, we would not be able to retrieve this information due to the loop gateway being some levels above the LCA.

To provide a more global information than the local direct causality, depths given by Definition 7.4 can be used. They summarize the behavioural situation of the given node. See, for instance, the processes of Figure 7.4. Depth of activity D could be seen as the sum of the *blocks* found from the root to the node.

$$\begin{aligned}
\delta_{T_1}(D) &= 1(root) + 2.5(Seq) + 1(And) + 0.5(Or) \\
\delta_{T_2}(D) &= 1(root) + 3(Seq) + 1(And) \\
\delta_{T_1}(D) - \delta_{T_2}(D) &= (1-1)(root) + (2.5-3)(Seq) + (1-1)(And) + (0.5)(Or) \\
&= -0.5(Seq) + 0.5(Or) \qquad (7.1)
\end{aligned}$$

Notice that by considering the difference of the two depths, i.e. the value considered by the Cophenetic distance, we start highlighting where are the differences, and the type of changes committed, of the behaviour up to activity D.

When comparing pairs of activities, the cophenetic distance does not only consider the depth of the two activities but also the LCA. Following the previous example, let's compare activity D with C:

$$\begin{aligned}
\delta_{T_1}(C) - \delta_{T_2}(C) &= (1-1)(root) + (2.5-3)(Seq) \qquad (7.2) \\
&\quad + (1-1)(And) + (0.5-0.5)(Or) \\
&= -0.5(Seq) \qquad (7.3) \\
\delta_{T_1}([C,D]_{T_1}) - \delta_{T_2}([C,D]_{T_2}) &= (1-1)(root) + (2.5-3)(Seq) + (1-0)(And) \\
&= -0.5(Seq) + 1(And) \qquad (7.4)
\end{aligned}$$

The Cophenetic value of C stores the differences on the previous block in the sequence, as it did with Activity D. Besides, the Cophenetic value of activities C and D captures again the difference in the sequence and also an AND gateway. Hence, the pair of activities C and D are a step closer to the end in one of the two process models. But more interesting properties could be extracted by measuring the difference of such Cophenetic values: Whilst the cophenetic value $\delta_{T_1}([C,D]) - \delta_{T_2}([C,D])$ gives an idea of the difference of the two processes up to the LCA $[C,D]$, these two new values provides the same differential analysis on the paths from the ancestor to the activities. In this example, $(2)-(3)=1$ indicates that the position of C with respect to their common ancestor differ in the insertion of an AND gateway; whilst in the case of activity D, $(1)-(2)=0.5$ recognizes that an OR gateway has been added, or replaced by an *AND*, in one of the models.

This example shows the potential of the LCA, and the Cophenetic values, to generate more understandable and user-friendly comparison tools between process trees. Definition 7.7 shows two possible sentences we could build thanks to this information.

Definition 7.7. *A set of human-readable differences can be generated using the Cophenetic values.*

- *Given a pair of activities A and B such that they differ in the behaviour explained by their LCA. We could say that*

  "In the first model, Activities A and B are (in sequential order / co-occurrent / conflict). Whilst they are (in sequential order / co-occurrent / conflict) in the second model. Besides, the position of

this behavioural decision differ in $\delta_{T_1}([A,B]) - \delta_{T_2}([A,B])$ units."

- *Given a pair of activities A and B showing the same causality but $\delta_{T_1}([A,B]) - \delta_{T_2}([A,B]) \neq 0$. We could say that*

  "Activities A and B show the same causality, but the position of this behavioural decision differ in $\delta_{T_1}([A,B]) - \delta_{T_2}([A,B])$ units."

In this section a formal guarantee for process tree equality based on Cophenetic distance has been presented, which restricts process trees to be deterministic. Next section lifts this restriction deriving an approximate metric based on the existence of a matching between the two process trees.

7.5 Distance between Indeterministic Process Trees

Only a small fraction of the process models generated from the human interaction with the source code repository are deterministic Process Trees. In the general case, each SVN command is executed several times during a developer day of work, and usually in different contexts producing processes with several duplicated activities. Unfortunately, the Cophenetic distance definition does not easily extend to such a kind of process. Figure 7.5 depicts an example of two indeterministic process trees where one activity, A, is duplicated. The Cophenetic distance cannot be used as it is was previously defined. First, the left model has two options for the depth of activity A. And more importantly, when computing the LCA of A and C, the results

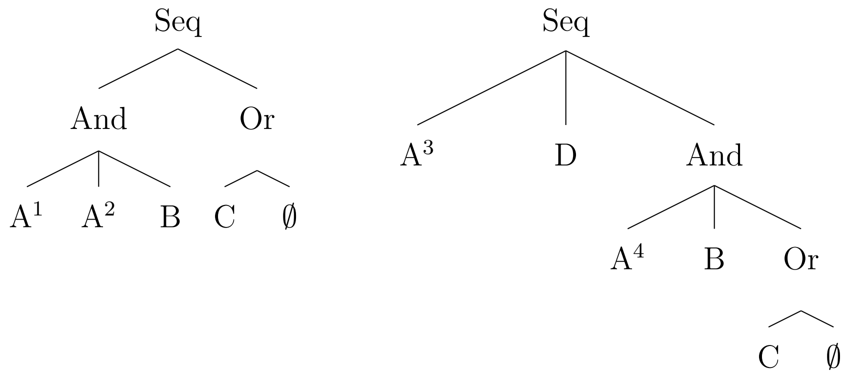


Figure 7.5: Example of two indeterministic process trees. Activities A are indexed for the sake of simplicity, but all of them are indistinguishable.

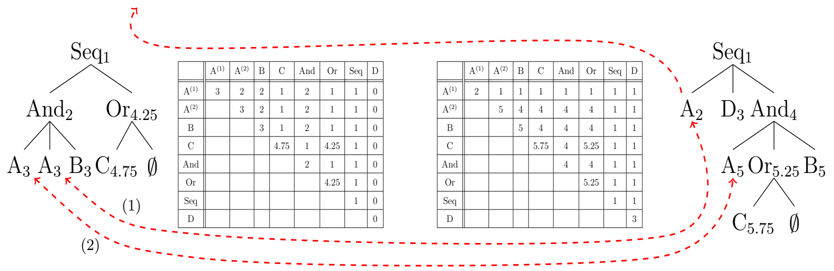
depend on which copy of activity A we chose. For instance, the LCA of A^3 and C is the root, but the AND gateway w.r.t. A^4. Nevertheless, we can still approximate an upper bound similarity metric between indeterministic process trees. In this section, we present a technique that can still be applicable when (some of) the input process trees are indeterministic.

Notice that two process trees T_1 and T_2 are equal if there exists a relabeling of both process trees such that each new label replaces the same label in both models, the resulting process trees are deterministic and their cophenetic distance is zero. Such a relabeling could also be seen as a matching between the activity nodes of both process trees. We could tackle the challenge of extending the Cophenetic distance by making use of such a matching: instead of considering pairs of activities (uniquely represented in a deterministic process tree), this similarity metric compares two pairs of matched nodes. The aforementioned ambiguities among repetitions of an activity are removed by considering these matches.

Definition 7.8. *Let ω be a matching between the nodes of T_1 and the nodes of T_2, we define their* ***matching Cophenetic distance*** *over ω as*

$$d\omega_{\varphi} = \sum_{(i_1,i_2)\in\omega} \sum_{(j_1,j_2)\in\omega} |\varphi_{T_1}(i_1,j_1) - \varphi_{T_2}(i_2,j_2)|$$

Notice that the nodes i_1, i_2, j_1, j_2 in Definition 7.8 are not necessarily representing activities in the model. Such a distance considers all nodes as labeled. The quality of such a similarity metric depends on the quality of the matching ω. On top of that, the utility of the measurement decreases if activity labels are not preserved by the matching.



	$A^{(1)}$	$A^{(2)}$	B	C	And	Or	Seq	D
$A^{(1)}$	3	2	2	1	2	1	1	0
$A^{(2)}$		3	2	1	2	1	1	0
B			3	1	2	1	1	0
C				4.75	1	4.25	1	0
And					2	1	1	0
Or						4.25	1	0
Seq							1	0
D								0

	$A^{(1)}$	$A^{(2)}$	B	C	And	Or	Seq	D
$A^{(1)}$	2	1	1	1	1	1	1	1
$A^{(2)}$		5	4	4	4	4	1	1
B			5	4	4	4	1	1
C				5.75	4	5.25	1	1
And					4	4	1	1
Or						5.25	1	1
Seq							1	1
D								3

Figure 7.6: Example of two indeterministic process trees and a matching Cophenetic distance (represented as a matrix) with respect to a certain node matching. All nodes are matched to their respective nodes with the same label, except activities A (discontinued lines (1) and (2) depict how they are paired) and activity D that does not have a representative node in the first tree. Subscripts depict the depth of the nodes.

Figure 7.6 depicts an example of such a matching Cophenetic distance of the models of Figure 7.5. From the set of all the possible matching, we choose to pair activities with the same label and, for activities A and D, we considered the pairs depicted by discontinued lines. Notice that activity D does not have a matched node in the first tree. In the middle of the Figure, one can find the Cophenetic vectors of both process trees. When considering Activity D, we treat this case as is if the activity does not exist in the first model. This example shows how the matching Cophenetic distance is computed for a specific node matching, but we could iterate over all matchings and get the minimum value possible.

Definition 7.9. *We define the **minimum matching Cophenetic distance** as*

$$d_{\min\varphi}(T_1, T_2) = \min_{\omega} d\omega_{\varphi}(T_1, T_2)$$

where the matching ω preserves activity's labels.

Although the matching Cophenetic distance is a quadratic-time algorithm [23], once we have chosen a particular matching ω, it is still computationally infeasible to compute this distance for each candidate ω in the *minimum matching Cophenetic distance*. In a practical scenario in which the size of the process trees made it impossible to test all possible matchings, one would be able to bound this ideal distance with an approximate node matching. Although current matching algorithms [8,61,74] focus on preserving the structure of the graph, they could be used to approximate this matching due to the structural approach of the Cophenetic distance.

We have chosen the Flexible Tree Matching algorithm (FTM) [61] for estimating the minimum matching Cophenetic distance with indeterministic process trees. The FTM finds the minimum-cost matching that takes into account the cost of relabeling a node, removing or adding a node, and breaking structural relations between nodes (such as direct descendants and siblings). Notice the resemblance to the definition of the Graph Edit Distance (GED): The cost of the matching resulting from FTM is an approximation of the GED, but assessing also the cost of not having the same neighbors. Tuning these costs allows us to focus on mapping nodes with the same activity (we set to 1, the maximum value, the cost of relabeling) and diminishes the relevance of structural differences. The following proposition establishes also a complexity bound:

Proposition 7.10. *The FTM needs at least $O(M \cdot N^3 \log N^2)$ operations to approximate the matching between two process trees T_1 and T_2. Where M is the number of iterations needed by the algorithm (i.e. the expected quality of the results) and N is the total number of nodes in T_1 and T_2.*

Proof. The Flexible Tree Matching iterates M times over a randomly generated matching, to retrieve the find the best possible matching. In each iteration, the algorithm needs to recompute the N pair of matches. A weighted bipartite N^2 graph is considered, where weights represent the cost of adding such a pair to the matching. To get the best outcomes from this choice, the algorithm sort all the edges and randomly chooses one of the costless edges. Hence, each iteration of the Flexible Tree Marching needs $O(N^3 \log N^2)$ operations, plus the complexity of computing the cost of each pair of nodes (which may involve traversing the whole matching depending on the implementation). □

In summary, extending the technique of this chapter to indeterministic process trees requires to first compute a matching and then compute the

Cophenetic distance over this matching. This comes with an increase of the complexity due to the need to compute a matching, a step that dominates the complexity of the whole approach. In the next section, we evaluate the proposed method on various types of benchmarks.

7.6 Evaluation

We divided the evaluation of our similarity metric in three experiments: First we consider a small set of synthetic process models to position our metric with respect to already established comparison tools. Secondly we check that the results given by our approach are consistent with two other metrics in a set of real process models. Finally, we stress the Cophenetic distance with large process models to assess its scalability.

Qualitative Comparison. Figure 7.7 depicts eight models extracted from [12]. These models were used in [12] to evaluate different similarity metrics. All models are deterministic, and share the same activity set except process model V_3. Table 7.1 depicts the similarity given by the Cophenetic distance and state-of-the-art process models distances. In order to compute the Cophenetic distance, all models have been represented as process trees. Notice that the inclusive gateway of model V_3 cannot be translated to a deterministic process tree (because only exclusive ORs are accepted), but it was translated to an AND gateway with all internal branches being completely optional. The Cophenetic distance differentiates models V_0 and V_2, but considers V_0 more similar to V_5 than V_4. Discrepancies shown in Table 7.1 highlights the lack of a clear definition of *similarity*. Overall, the Cophenetic distance offers a different view for the comparison with respect to the other metrics.

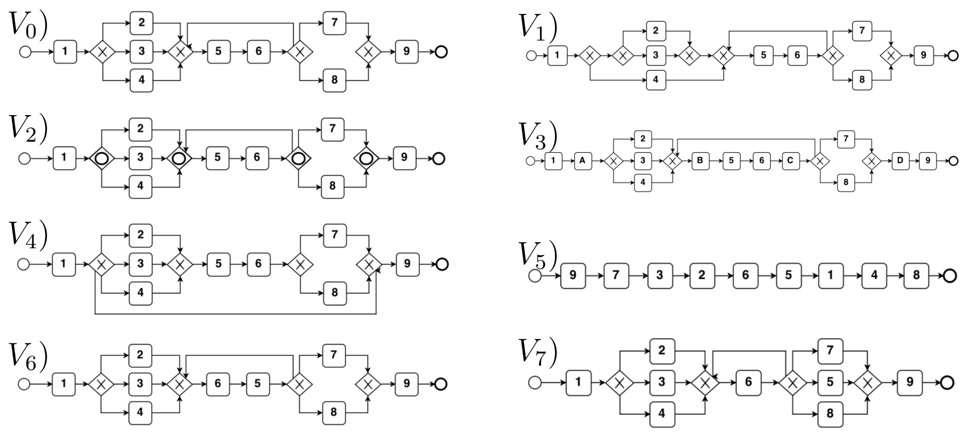


Figure 7.7: 8 process models extracted from [12]. Process models $V_1, \ldots, V_7$ are variants from the same process model V_0.

Correlation with two other metrics. We gathered 700 pairs of deterministic process models from the SAP Reference Book [24] to compare our approach to two other established process model similarity metrics in a real scenario. We have chosen the traditional graph edit distance as a representative of a structural comparison tool; and, for the behavioural part, we have chosen the Event Structures technique [7]. Figure 7.8 depicts the comparison of the three metrics. The X and Y coordinates of a point depicts the distance given by two comparison tools, and the color represents the density of pairs in such a situation. I.e., the less dark blue a point is, the more pairs of models satisfying this relation. One can check, for instance, that most of the models differ at 10 units by the behavioural technique and the graph edit distance. Histograms show that the measurements given by the three

$\mathbf{V_0}$ **compared to**	$\mathbf{V_1}$	V_2	V_3	V_4	V_5	V_6	V_7
Cophenetic Distance	█	□	□	░	▒	▓	▒
Percentage of Common Nodes and Edges	█	█	□	▒	□	░	▒
Node- and Link-Based Similarity	▒	█	□	░	█	▒	□
Graph Edit Distance	█	█	□	▒	□	░	▒
Label Similarity and Graph Edit Distance	□	█	░	▓	□	▒	▒
Number of High-Level Change Operations	█	░	□	▒	□	▓	▓
Comparing PMs Represented as Trees	█	█	□	░	□	▒	▒
Comparing Dependency Graphs	█	▓	□	▒	□	▒	░
Causal Behavioural Profiles	█	▒	□	▒	□	▓	░
Event Structures	█	□	▓	▒	□	▒	░
Longest Common Subsequence of Traces	█	▒	░	█	□	▓	▒
Similarity Based on Traces	█	▓	□	▒	□	▒	░

Table 7.1: Similarity of model V_0 to the rest of models from Figure 7.7 with respect to several similarity metrics. Similar models are depicted by darker cells. Values were extracted from [12], except for the Cophenetic and Event Structures [7].

metrics are correlated[5]. It is not clear that the same factual differences are measured by the three metrics, but the scores obtained are aligned with the two other established metrics.

Scalability of the Cophenetic distance. We also study the differences in performance over large process models. We considered 7 additional pairs

[5]In all three cases, the Pearson correlation coefficient is above 0.85 with a p-value, for testing non-correlation, below 10^{-12}.

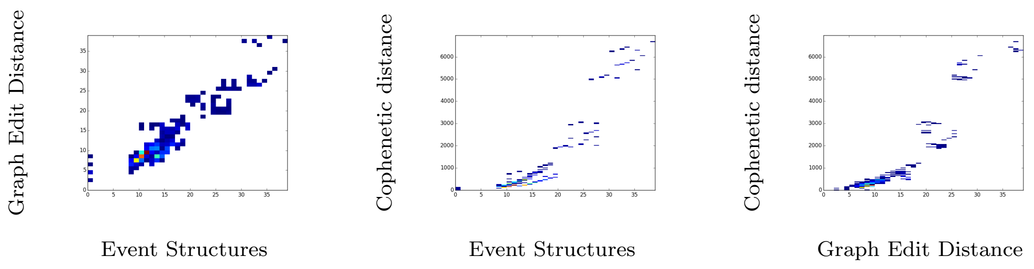


Figure 7.8: A set of two-dimensional histograms comparing the results of the three comparison tools in the SAP dataset.

of process models and run the three comparison tools on each pair. The size of the processes, presence of concurrent blocks and loops varies among the models to test the applicability of the three tools. Table 7.2 depicts the size of such models, and the time needed to measure the differences[6]. The Graph Edit Distance wins all the tests, the tree structure made this algorithm work way faster than usual. The complexity of the other two tools increases significantly with respect the number of activities, although the growth rate in the Cophenetic distance is considerably smaller. Notice the second pair of models, in which concurrency is present, make the behavioural tool run out of memory. Besides, we decided to stop the behavioural tool after 12 hours in all tested process models with more than 100 activities, even with deterministic process models in which the cophenetic distance showed significantly smaller times. This analysis allow us to recommend the Cophenetic distance over

[6]We ran all tests in a virtual machine running Linux elementary OS for ensuring compatibility will all implementations, with 1 CPU and 4 gigabytes of RAM reserved from the host machine. The host used a 4-core Intel $i7-4600$ CPU running at 2.1 GHz with 8 gigabytes of RAM.

other behavioural approaches to analyze big process models [7].

Table 7.2: Time spent in computing the distance between a few selected process models. The table shows the number of activities in each process model, the distance given by the Cophenetic metric and the other two selected comparison tools, and the time used.

Size	Deterministic	Concurrency	Realistic	d_{φ}	Time	d_{ES}	Time (Event Structures)	d_{GED}	Time (GED)
25	No	No	No	0	1.71 s	0	1.63 s	0	0.03 s
30	Yes	Yes	No	7250	0.005 s	Run out of memory		40	0.009 s
50	No	No	No	3713	54.37 s	9	90.54 s	93	0.12 s
60	No	No	Yes [8]	190	322.48 s.	$>$ 12 hours		167	0.19 s
100	No	No	No	16615	467.23 s	$>$ 12 hours		184	0.42 s
100	Yes	No	No	452299	0.57 s	$>$ 12 hours		189	0.14 s
200	Yes	No	No	2441571	2.28 s	$>$ 12 hours		371	0.53 s

7.7 Application to the Comparison of User's behavior

In this section, we explore the idea of compare the behavior of several users using the Cophenetic distance explained in this section. First, we describe the provided behavioural data of 200 individuals and their organizational roles in the company. Process discovery is used to summarize the behavioural data from each individual, and dissimilarities between those processes are meant

[7]Remember that the scale of metrics d_{φ}, d_{ES} and d_{GED} is different, a fact that explains the differences on the absolute values provided in each one.

[8]We discover these processes by analyzing the accesses of two developers to an internal source code repository. Figure 7.1 depicts an example of a pair of such type of processes.

to measure differences in the behaviour of the individuals. Then, we test some clustering approaches with three different similarity metrics to measure how good they approximate the original organizational rules.

7.7.1 Context

Apache Subversion (SVN) is a software versioning and revision control system. Software developers use SVN software to collaborate in the maintenance and development of software, by monitoring changes in files such as source code, web pages and documentation. All accesses to a SVN repository are done through HTTP/S, as specified in the WebDAV/DeltaV protocol. It turns out [92] that those read and write requests over HTTP/S can be translated to human-friendly SVN commands such as *svn update* or *svn commit*.

Continuing the work done by Li Sun et.al. [92], we model the behaviour of developers by using process discovery techniques. First, SVN commands are retrieved from the system and considered as events of a system that represents the developer, and then a trace is defined as all commands executed during a complete business day.

This industrial dataset contains all the accesses of more than 200 individuals to one repository of CA Technologies in production for three years. After pruning users with few accesses to the repository, 83 individuals were kept in the study and their organizational roles were retrieved at the end of the monitoring phase. In particular, 37 *Forward Engineering*, 19 *Quality Assurance Engineers*, 16 *Sustaining Engineer*, 5 *Support*, 2 *Services*, 1 *SWAT Engineer*, 1 *Infrastructure*, 1 *Technical Writers*. The following list summarizes the responsibilities for each role.

- *Forward Engineers* (R1) are in charge of the implementation of new features.
- *Quality Assurance Engineers* (R2) plan, run and design use cases or tests.
- *Sustaining Engineers* (R3) are in charge of solving defects, as well as ensuring that software successfully passes all tests.
- *SWAT engineers* (R4) are in charge of implementing custom integrations.
- *Support* (R5), *Services* (R6) and *Infrastructure Engineers* (R7) interact with internal and external customers with respect to defect detection and solution, software installation and configuration, and maintenance of the infrastructure of Software as a Service solutions provided by the company. *Support Engineers* might push some quick fixes into products.
- *Technical Writers* (R8) collaborate with Forward, Sustaining and Quality Assurance Engineers for creating helpful Knowledge Base and User Guides. Technical Writers are asked to use the source code repository to maintain different versions of the documentation.

Among all the engineers, and fairly distributed among roles, 9 individuals are Managers of a team. Besides, one agent is labeled as a bot, although the purpose of such bot is unknown to the authors of this chapter. Notice that one has the possibility of advancing in their career and change to another department, and, therefore, some individuals might have been *misclassified* as their latest role. Infrastructure and Service Engineers are not supposed to access the repository in their usual pipeline and, therefore, might have been promoted during the project. Nevertheless, clustering may help us to deduce their original roles in the organization.

During the rest of this section we plan to answer the following questions in regard of this scenario:

- How good is clustering of process models for approximating the original role of the individuals?
- Which is the expected role of the bot? And what about the role of other anomalies?

7.7.2 Homogeneity of Roles in Process-based Clustering

In order to measure the quality of the clustering, we will use the **purity** as the metric for measuring the homogeneity of the discovered groups. Let $C = \{C_1, \ldots, C_m\}$ be a clustering of the process models, and $R_i(C_j)$ be the number of individual in cluster C_j with the role R_i, then the purity is defined as

$$\text{Purity}(C, R) = \frac{1}{\text{Number of processes}} \sum_{j \in C} \max_{R_i} R_i(C_j)$$

In other words, the purity computes accuracy as if we label all individuals inside a group with the most popular role inside it. In particular, very heterogeneous groups of individuals will lead to a poor purity.

Figure 7.9 depicts the purity obtained from the SVN-repository dataset with respect to the number of clusters using hierarchical clustering. The

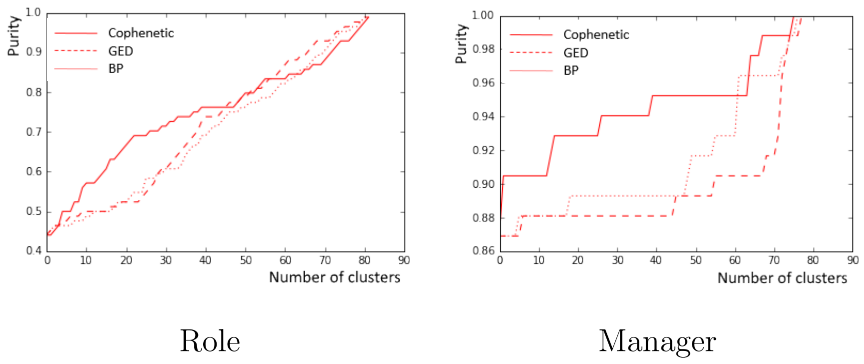


Figure 7.9: The solid line depicts the evolution in the purity of the Cophenetic-based hierarchical clustering as the number of clusters increase, whilst the dashed (resp. dotted) line depicts the graph edit distance (resp. Behavioural Profiles). Two different experiments were performed for detecting individuals' role and their status as managers.

X-axis represents the number of clusters considered, and its respective purity is represented by the Y-axis. The solid line depicts the purity of the clustering obtained based on the Cophenetic distance. One may notice that the three techniques provide similar purity metrics for the first clusters, but then the Cophenetic distance provides significantly better results starting from 5 discovered groups until they converge again at 40 groups. In fact, the range between 5 and 40 groups is the most representative usage of clustering techniques as it is close to the real number of roles and one starts to discover groups of 1 individual when setting larger number of clusters. On the right graphic, only the status as a Manager has been considered. Due to the low number of Managers in the dataset, these results denote the ability of efficiently filter out abnormal behaviour.

	Cophenetic		**GED**		**Behavioral Profiles**	
	Precision	Recall	Precision	Recall	Precision	Recall
Manager	0.71	0.55	No individual was labelled			
Forward Engineer	0.64	0.76	0.46	0.97	0.45	1.00
Sustaining Engineer	0.00	0.00	0.57	0.26	0.00	0.00
Quality Assurance	0.34	0.57	No indiv.		1.00	0.05
Support	0.50	0.40	No individual was labelled			
Others	No individual was labelled					

Table 7.3: Precision and Recall for each of the roles in the organization by considering a hierarchical clustering with 6 groups. In some cases, none of the groups had enough representation of a role.

Table 7.3 summarizes the precision and recall of the hierarchical clustering of Figure 7.9 when we set the number of clusters to 6. Again, we consider the classification as if the predicted label of all elements inside a cluster is the role of the more popular role inside the cluster. These results show again the capabilities of the Cophenetic distance to highlight the *Manager* role and provides, in general, better results for all roles, except the *Sustaining Engineer*. On the other hand, GED and Behavioral Profiles tend to group all individuals in a very big, and heterogeneous, cluster and, therefore, we obtain those roles with high recall but low precision (and those with high precision but very low recall).

As for the bad performance with respect to Sustaining Engineers, notice that responsibilities of the Sustaining ($R2$), Quality Assurance ($R3$), SWAT ($R4$) and Support engineers ($R5$) are all related to defects and bug fixing, and, therefore, they may share some common behaviour and practices. Besides, the number of Sustaining Engineers is slightly below the number of Quality Assurance Engineers, and, hence, it is more likely to label users as Quality Assurance Engineers in case of grouping them together.Table 7.4 summarizes the precision and recall for a clustering of 6 groups. Notice that precision and recall of the Forward Engineer category are not significantly affected in the case of the cophenetic distance, indicating the existence of groups with a strong presence of Forward Engineers. On the other hand, precision and recall are very affected in both GED and Behavioral Profiles cases. The results provided by the GED are an indication of one or more small groups of Forward Engineers (perfect precision, but low recall), and a big group in which half of the developers have a role in $R2345$ and the rest are Forward Engineers or other minor roles. As for behavioral profiles, results are slightly worse than the cophenetic distance, but still incapable of detecting the group of Managers.

We have run the same experiments using DBSCAN [35] as the clustering method. The key benefit of DBSCAN is that the number of clusters is not fixed prior to the clustering, as it defines clusters as groups of individuals that are densely together[9]. Unfortunately, results are significantly worse than the provided by the hierarchical clustering – with purity not surpassing 0.5 across several hyperparameter of the DBSCAN algorithm.

[9]I.e. for every process model in the cluster, there must be at least k process models at distance less or equal than d. Both k and d are manually fixed.

	Cophenetic		**GED**		**Behavioral Profiles**	
	Precision	Recall	Precision	Recall	Precision	Recall
Manager	0.71	0.55	No individual was labelled			
Forward Engineer (R1)	0.60	0.78	1.00	0.05	0.65	0.35
R2345	0.72	0.63	0.50	1.00	0.56	0.85
Others	No individual was labelled					

Table 7.4: Precision and Recall for each of the roles in the organization by considering a hierarchical clustering with 6 groups after merging *Sustaining*, *Quality Assurance*, SWAT and *Support* Engineers into a unique role *R2345*. In some cases, none of the groups had enough representation of a role.

7.7.3 Inducing the (real) Role of Outliers

Some role *anomalies* were present in the dataset. For instance, two individuals were classified as Service Engineers ($R6$) although accessing to the source code repository is not part of their responsibilities. As we have already mentioned, the role data was obtained during the finalization of the project and, hence, the worker may have changed from one department to another. In this case, one service engineer ($R6$) is more close to Quality Assurance Engineers ($R3$), and the other is close to a group of Forward Engineers ($R1$). The three distances are consistent with these results. With respect to the Infrastructure Engineer ($R7$), the cophenetic distance and behavioural profiles map this user close to Sustaining Engineers ($R2$) whilst the graph edit distance relate her to Forward engineers ($R1$). Finally, with respect to the agent labeled as a BOT, the cophenetic and the graph edit distance group it with other Quality Assurance Engineers ($R3$). This might be a hint that the bot is indeed an automatic testing system.

7.8 Discussion

In this chapter we have adapted Cophenetic vectors from computational phylogenetics area to be able to automatically compare process models. The state of the art techniques for comparing process models can be split into structural and behavioural. In the former, process models are considered as labeled graphs and the comparison is regarded as edit operations over their edges, nodes or both. In contrast, behavioural techniques focus at the comparison of the execution semantics of the compared models.

We have shown that our proposed distance fits between the structured and behavioral worlds, as a metric defined on differences in the structure of the process tree notation may algo contain information related to the behavioural similarity. Besides, albeit behavioural techniques are typically computationally demanding, the structural-but-behavioural intermediate approach has shown excellent results on time computation and scalability, allowing BPM practitioners to efficiently measure the behavioural similarity between process models.

Next steps would focus on extending behavioural differences from the difference of Cophenetic values. There is also room to improve the utility and efficiency of the comparison of indeterministic process trees. The presented approximated matching is computed without taking into account the Cophenetic distance itself, but there might be a better matching algorithm that exploits the properties of the Cophenetic values.

CHAPTER 8 Conclusions

The industry is facing a transition towards decentralization of their workforce, which might be compromised by a combination of hired workers, computers and one-time contributors due to advances in Crowdsourcing technologies. Hence, future business process management techniques must support the execution and monitoring of activities, processes and individuals in a distributed collaborative scenario.

For this thesis, we initially set two objectives aiming to advance towards a broader concept of Crowdsourcing. First, a close to market and exploitation-oriented objective for supporting crowdsourcing methodologies in industrial scenarios. Then, a more scientific-oriented objective in which we aim towards the creation of a *normal behavior* profile of users in a digital platform. Following, one can find how this thesis contributed towards both objectives. Then, we set the limitations of our contributions and we set lines of research to continue the work done in this thesis.

Advances on supporting crowdsourcing in industrial scenarios

The first difference that we noticed between current crowdsourcing technologies and industry is the lack of a notion of processes. Business processes are fundamental for the industry, as they guide workers on how (and when) they should execute and allow entrepreneurs to monitor and improve their business. If crowdsourcing is going to substitute the current outsourcing model, then processes should be a first citizen in future crowdsourcing platforms. For tackling such technological gap, we proposed CrowdWON in Chapter 3. CrowdWON is not only a graphical modelling language for crowdsourced processes, with the inherent benefits on the dissemination of processes and best practices, but we also proposed a variant capable of define and automatize the execution of the processes. Such variant was fruit of a Proof of Concept we built in collaboration with a translation provider, in which some of their processes were tested. Besides, Chapter 5 shows how we leveraged knowledge from Crowdsourcing and industrial translation for defining a worker ranking mechanism that slightly advances on current reputation systems implemented in Crowdsourcing platforms.

Apart from the collaboration with a translation provider, we explored other internal processes that had the potential to be crowdsourced. We detected that technical support could be partially outsourced, or, at least, it has the potential to allow workers to participate remotely in the resolution of customer problems. Unfortunately, there is no generic procedure explaining how support engineer should perform in their day-by-day. The lack of such best practices hinders the training of new support engineers, which need to develop their own *procedure* by experiencing lots of interactions with customers. We saw a clear link with process mining techniques, which would analyze the actions taken by the support engineers and discover the underlying procedure. But, actions in such scenario are textual descriptions of the work done and not elements from a set of actions defined by management. Our embedding-based Event Variability Reduction technique, as proposed in Chapter 4, is capable of comparing the similarity of the textual descriptions and merge all those actions that produced similar outcomes. Thanks to that, we were able to discover a generic procedure that we applied for predicting the customer satisfaction. The results shows that this technique has the potential to be used for the monitoring of crowdsourced activities.

Advances on the discovery of the *normal* user behavior

During this thesis we assumed and explored that process mining techniques could be used for generating a profile of the user's behavior. Digital platform have the potential of registering all actions performed by their users and, then, induce their behavior. We opened this line of research inspired by the initial results of Rzeszotarski et.al. [87] in which they discovered that users' behavior (considering scrolling, mouse movements, clicks, keystrokes, delays) was correlated with quality.

When discovering the profile of the user, we expect to achieve a certain level of precision in order to ensure that the process model uniquely represents the user. We considered in Chapter 6 a new addition to the label splitting literature. Contrary to state of the art techniques, our label splitting technique does not completely discard the original process model but enhances it by studying the behavior of loops. In fact, iterative subprocesses are one of the major threats to the precision of a process model and, still, one of the most common behaviors seen in human-computer interaction processes.

Afterwards we explore in Chapter 7 the idea of comparing two process models and, hence, the respective users described by the process models. We have applied this metric to an industrial dataset compromising the accesses of software developers to a source code repository and we were able to see that the functional role was reflected on the behavior explained by the process models.

None of the contributions of this thesis proposed a specific mechanism for measuring the *normality* score that we set as an objective of this thesis. We assume that conformance checking techniques could be used to measure how far (and, hence, abnormal) is a trace towards the profile of the user. Besides, the comparison of user profiles would allow crowdsourcing platforms to create a profile of the *community* by considering events from all his users (or a subset of them, such as those efficient workers known by the platform administrators) and then compare individual process models towards the community model.

8.1 Limitations

The evaluation of industrial exploitation outcomes of this thesis is rather limited to the small set of stakeholders involved in their conception. For instance, the evaluations of CrowdWON and the Event Variability Reduction are largely based on the feedback provided by the representatives of the translation provider and CA Technologies during the assessment of the

Proof of Concepts. Besides, our assessment of CrowdWON may have been biased by the fact that usual crowdsourced processes are simple and short. It remains to run a more in-depth evaluation by broadening the scope to more use case and conducting surveys on usability and utility with experts in crowdsourcing and process modelling.

We also need to explore if some of the other assumptions done during this thesis are valid. For instance, we assumed that the comparison of event names is enough to decide if the two actions should also be considered similar. In the case of an IT technical support system, this assumption seems to be true as the support engineers are asked to input a description of the action performed.

The major limitation of this thesis is the lack of experiences on the user profiling and worker ranking in real-life crowdsourcing platforms. The evaluation of the worker ranking has been done by simulating a set of expected behaviors in a crowdsourcing platform, including a rather suspicious fraudulent user. Although this shows some consistency for the metric, we are not able to prove the consistency in a scenario in which the users may intelligently adapt to the implemented reputation and quality assurance mechanisms. As for the *normality* metric, this means that we were not able to obtain a real dataset in which we could evaluate a user's deviations from her process model. Instead, we proved our outcomes utility in related scenarios, showcasing the promising future of such research challenge.

8.2 Future work

In this thesis, we explored the idea of using process discovery techniques for automatically generating a profile of the users of a digital platform. Initial results on a series of industrial datasets shows that this is a promising research line, but its applicability on other contexts is yet to be explored.

During the development of this thesis, user profiles were used for grouping users with a similar behavior. Organizations can benefit from such system, as this would allow practitioners to have a general idea of their users by visualizing a few processes. It would be interesting to consider process discovery for the categorization of users. I.e. discover as many process models as user categories, and then classify users depending on the fitness of the event log with respect to each process model. For doing that, we need an efficient discovery technique that ensures high fitness and precision with respect to users in the group it is representing, but low fitness with others. This will help systems to classify new users within the first interaction with the platform, and organizations would be able to target features, or adapt the UI for improving the user experience, to selected users based on their behavior. All these use cases are yet to be explored.

Another interesting technical challenge was arisen from the development of this thesis. Conformance checking is usually performed by finding the best alignment, which, in practice, assumes that the link between events and activities is a binary relation. I.e. an event is an instance of an activity or they are completely unrelated. In particular, current conformance checking methods allows for setting a cost for replacing an event for another, but in practice there is no mechanism for computing such costs. Nevertheless, the embedding-based similarity we have reviewed in Chapter 4 could be used for defining for tackling this issue.

It would be interesting to create a simulation of crowdsourced processes based on a taxonomy of crowdsourced tasks, a modelling language as Crowd-WON and a pool of workers modeled based on real user profiles. With such simulator, time-to-completition and quality of crowdsourced processes could be predicted during design time.

List of Figures

List of Tables

Bibliography

[1] A. Adriansyah, J. Munoz-Gama, J. Carmona, B. F. Dongen, and W. M. Aalst. Measuring precision of modeled behavior. *Inf. Syst. E-bus. Manag.*, 13(1):37–67, February 2015.

[2] Arya Adriansyah. *Aligning observed and modeled behavior*. PhD thesis, Technische Universiteit Eindhoven, 2014.

[3] Salman Ahmad, Alexis Battle, Zahan Malkani, and Sepander Kamvar. The jabberwocky programming environment for structured social computing. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, pages 53–64, New York, NY, USA, 2011. ACM.

[4] Vamshi Ambati, Stephan Vogel, and Jaime Carbonell. Collaborative workflow for crowdsourcing translation. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, CSCW '12, pages 1191–1194, New York, NY, USA, 2012. ACM.

[5] Hazleen Aris. Current state of crowdsourcing taxonomy research: A systematic review. In *Proceedings of the 6th International Conference*

on Computing and Informatics, ICOCI 2017, 2017.

[6] Hazleen Aris and Marina Md Din. Crowdsourcing evolution: Towards a taxonomy of crowdsourcing initiatives. In *Pervasive Computing and Communication Workshops (PerCom Workshops), 2016 IEEE International Conference on*, pages 1–6. IEEE, 2016.

[7] Abel Armas-Cervantes, Paolo Baldan, Dumas Marlon, and Luciano García-Bañuelos. Behavioral comparison of process models based on canonically reduced event structures. *Business Process Management: 12th International Conference, BPM 2014, Haifa, Israel, September 7-11, 2014. Proceedings*, pages 267–282, 2014.

[8] Vikraman Arvind, Johannes Kobler, Sebastian Kuhnert, and Yadu Vasudev. Approximate graph isomorphism. In *Proceedings of the 37th International Symposium, MFCS 2012*, 2012.

[9] Bahadir Ismail Aydin, Yavuz Selim Yilmaz, Yaliang Li, Qi Li, Jing Gao, and Murat Demirbas. Crowdsourcing for multiple-choice question answering. In *Proceedings of the Twenty-Sixth Annual Conference on Innovative Applications of Artificial Intelligence*, 2014.

[10] Thomas Baier, Jan Mendling, and Mathias Weske. Bridging abstraction layers in process mining. *Information Systems*, 46:123–139, 2014.

[11] Marco Baroni, Georgiana Dinu, and Germán Kruszewski. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of Association for Computational Linguistics (ACL)*, volume 1, 2014.

[12] Michael Becker and Ralf Laue. A comparative survey of business process similarity measures. *Comput. Ind.*, 63(2):148–167, February 2012.

[13] T. S. Behrend, D. J. Sharek, A. W. Meade, and E. N Wiebe. The viability of crowdsourcing for survey research. *Behavior research methods*, 43(3), 2011.

[14] Michael S. Bernstein, Greg Little, Robert C. Miller, Björn Hartmann, Mark S. Ackerman, David R. Karger, David Crowell, and Katrina Panovich. Soylent: A word processor with a crowd inside. In *Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology*, UIST '10, pages 313–322, New York, NY, USA, 2010. ACM.

[15] RP Jagadeesh Chandra Bose and Wil MP van der Aalst. Abstractions in process mining: A taxonomy of patterns. In *International Conference on Business Process Management*, pages 159–175. Springer, 2009.

[16] Daren C. Brabham. Crowdsourcing: A model for leveraging online communities. *The Participatory Cultures Handbook*, 2012.

[17] Joos C. A. M. Buijs. *Flexible Evolutionary Algorithms for Mining Structured Process Models*. PhD thesis, Technische Universiteit Eindhoven, 2014.

[18] Joos C. A. M. Buijs, Boudewijn F. van Dongen, and Wil M. P. van der Aalst. A genetic algorithm for discovering process trees. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2012, Brisbane, Australia, June 10-15, 2012*, pages 1–8, 2012.

[19] Joos C. A. M. Buijs, Boudewijn F. van Dongen, and Wil M. P. van der Aalst. Quality dimensions in process discovery: The importance of fitness, precision, generalization and simplicity. *Int. J. Cooperative Inf. Syst.*, 23(1), 2014.

[20] Joos CAM Buijs, Boudewijn F Van Dongen, and Wil MP van Der Aalst. On the role of fitness, precision, generalization and simplicity in process discovery. In *OTM Confederated International Conferences "On the Move to Meaningful Internet Systems"*, pages 305–322. Springer, 2012.

[21] Chris Callison-Burch and Mark Dredze. Creating speech and language data with amazon's mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, CSLDAMT '10, pages 1–12, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[22] T. Calvo, G. Mayor, and R. Mesiar. *Aggregation Operators*. Physica-Verlag, 2002.

[23] Gabriel Cardona, Arnau Mir, Francesc Rosselló, Lucía Rotger, and David Sánchez. Cophenetic metrics for phylogenetic trees, after sokal and rohlf. *BMC Bioinformatics*, 14(1):1–13, 2013.

[24] Thomas Curran, Gerhard Keller, and Andrew Ladd. *SAP R/3 business blueprint: understanding the business process reference model*. 1997.

[25] Gil Aires Da Silva and Diogo R Ferreira. Applying hidden markov models to process mining. *Sistemas e Tecnologias de Informação. AISTI/FEUP/UPF*, 2009.

[26] Nilesh Dalvi, Anirban Dasgupta, Ravi Kumar, and Vibhor Rastogi. Aggregating crowdsourced binary ratings. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 285–294, Republic and Canton of Geneva, Switzerland, 2013. International World Wide Web Conferences Steering Committee.

[27] E.P. Dawis, J.F. Dawis, and Wei Pin Koo. Architecture of computer-based systems using dualistic petri nets. In *In proceeding of 2001 IEEE International Conference on Systems, Man, and Cybernetics*, 2001.

[28] Javier de San Pedro and Jordi Cortadella. Discovering duplicate tasks in transition systems for the simplification of process models. In *Business Process Management - 14th International Conference, BPM 2016, Rio de Janeiro, Brazil, September 18-22, 2016. Proceedings*, pages 108–124, 2016.

[29] Remco M. Dijkman. Diagnosing differences between business process models. In *BPM 2008, Milan, Italy, September 2-4*, pages 261–277, 2008.

[30] Remco M. Dijkman, Marlon Dumas, and Luciano García-Bañuelos. Graph matching algorithms for business process model similarity search. In *BPM 2009, Ulm, Germany, September 8-10*, pages 48–63, 2009.

[31] Remco M. Dijkman, Marlon Dumas, Luciano García-Bañuelos, and Reina Käärik. Aligning business process models. In *EDOC 2009, 1-4 September 2009, Auckland, New Zealand*, pages 45–53, 2009.

[32] Remco M. Dijkman, Marlon Dumas, Boudewijn F. van Dongen, Reina Käärik, and Jan Mendling. Similarity of business process models: Metrics and evaluation. *Inf. Syst.*, 36(2):498–516, 2011.

[33] Schenk Eric and Guittard Claude. Towards a characterization of crowdsourcing practices. *Journal of Innovation Economics Management*, 7(1):93–107, 2011.

[34] Enrique Estellés-Arolas and Fernando González-Ladrón-De-Guevara. Towards an integrated crowdsourcing definition. *Journal of Information Science*, 2012.

[35] Martin Ester, Hans-Peter Kriegel, Jorg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press, 1996.

[36] Ju Fan, Meiyu Lu, Beng Chin Ooi, Wang-Chiew Tan, and Meihui Zhang. A hybrid machine-crowdsourcing system for matching web tables. In *Data Engineering (ICDE), 2014 IEEE 30th International Conference on*, pages 976–987. IEEE, 2014.

[37] Antonio Foncubierta Rodríguez and Henning Müller. Ground truth generation in medical imaging: A crowdsourcing-based iterative approach. In *Proceedings of the ACM Multimedia 2012 Workshop on*

Crowdsourcing for Multimedia, CrowdMM '12, pages 9–14, New York, NY, USA, 2012. ACM.

[38] Q. Gao and S. Vogel. Consensus versus expertise: a case study of word alignment with mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, CSLDAMT '10, pages 30–34, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[39] Christian W Günther, Anne Rozinat, and Wil MP Van Der Aalst. Activity mining by global trace segmentation. In *International Conference on Business Process Management*, pages 128–139. Springer, 2009.

[40] Christian W Günther and Wil MP van der Aalst. *Mining activity clusters from low-level event logs*. Beta, Research School for Operations Management and Logistics, 2006.

[41] Umair Ul Hassan, Sean O'Riain, and Edward Curry. Effects of expertise assessment on the quality of task routing in human computation. In *2nd International Workshop on Social Media for Crowdsourcing and Human Computation*, pages 1–10, 2013.

[42] M. Hirth, T. Hossfeld, and P. Tran-Gia. Cost-optimal validation mechanisms and cheat-detection for crowdsourcing platforms. In *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2011 Fifth International Conference on*, pages 316–321, 2011.

[43] M. Hirth, T. Hossfeld, P. Tran-gia, Matthias Hirth, Tobias HoÃŸfeld, Phuoc Tran-gia, M. Hirth, T. Hossfeld, P. Tran-gia, Matthias Hirth,

and Tobias HoÃŸfeld. âœanalyzing costs and accuracy of validation mechanisms for crowdsourcing platforms.â, 2012.

[44] Chien-Ju Ho and Jennifer Wortman Vaughan. Online task assignment in crowdsourcing markets. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, AAAI'12, pages 45–51. AAAI Press, 2012.

[45] J. Howe. Wired 14.06: The Rise of Crowdsourcing, 2006.

[46] Kazushi Ikeda and Michael S. Bernstein. Pay it backward: Per-task payments on crowdsourcing platforms reduce productivity. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 4111–4121, New York, NY, USA, 2016. ACM.

[47] Panagiotis G. Ipeirotis, Foster Provost, Victor S. Sheng, and Jing Wang. Repeated labeling using multiple noisy labelers. *Data Min. Knowl. Discov.*, 28(2):402–441, March 2014.

[48] Panagiotis G. Ipeirotis, Foster Provost, and Jing Wang. Quality management on amazon mechanical turk. In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, HCOMP '10, pages 64–67, New York, NY, USA, 2010. ACM.

[49] Srikanth Jagabathula, Lakshminarayanan Subramanian, and Ashwin Venkataraman. Reputation-based worker filtering in crowdsourcing. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, NIPS'14, pages 2492–2500, Cambridge, MA, USA, 2014. MIT Press.

[50] Kurt Jensen and Lars M. Kristensen. *Coloured Petri Nets: Modelling and Validation of Concurrent Systems*. Springer Publishing Company, Incorporated, 1st edition, 2009.

[51] Nicolas Kaufmann, Thimo Schulze, and Daniel Veit. More than fun and money. worker motivation in crowdsourcingâ“a study on mechanical turk. In *Americas Conference on Information Systems (AMCIS)*, 2011.

[52] G. Kazai, J. Kamps, and N. Milic-Frayling. The face of quality in crowdsourcing relevance labels: Demographics, personality and labeling accuracy. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 2583–2586, New York, NY, USA, 2012. ACM.

[53] Gabriella Kazai and Imed Zitouni. Quality management in crowdsourcing using gold judges behavior. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, WSDM '16, pages 267–276, New York, NY, USA, 2016. ACM.

[54] Tom Kenter and Maarten de Rijke. Short text similarity with word embeddings. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 1411–1420, New York, NY, USA, 2015. ACM.

[55] J. Kim, P. Nguyen, S. Weir, P. Guo, R. Milller, and K. Gajos. Crowdsourcing step-by-step information extraction to enhance existing how-to videos. In *CHI 2014*, 2014.

[56] A. Kittur, B. Smus, S. Khamkar, and R.E. Kraut. Crowdforge: crowdsourcing complex work. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, pages 43–52, New York, NY, USA, 2011. ACM.

[57] Aniket Kittur, Susheel Khamkar, Paul André, and Robert Kraut. Crowdweaver: Visually managing complex crowd work. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, CSCW '12, pages 1033–1036, New York, NY, USA, 2012. ACM.

[58] D. Kontokostas, A. Zaveri, S. Auer, and J. Lehmann. Triplecheckmate: A tool for crowdsourcing the quality assessment of linked data. In *Knowledge Engineering and the Semantic Web*, volume 394 of *Communications in Computer and Information Science*, pages 265–272, 2013.

[59] Pavel Kucherbaev, Florian Daniel, Stefano Tranquillini, and Maurizio Marchese. Crowdsourcing processes: A survey of approaches and opportunities. *IEEE Internet Computing*, 20(2):50–56, March 2016.

[60] Anand Kulkarni, Matthew Can, and Bjï¿rn Hartmann. Collaboratively crowdsourcing workflows with turkomatic. In *In Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work (CSCW '12)*, pages 1003–1012, New York, NY, USA, 2012. ACM.

[61] Ranjitha Kumar, Jerry O. Talton, Salman Ahmad, Tim Roughgarden, and Scott R. Klemmer. Flexible tree matching. In *Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI 2011)*, 2011.

[62] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pages 1188–1196, 2014.

[63] Sander J. J. Leemans, Dirk Fahland, and Wil M. P. van der Aalst. Discovering block-structured process models from event logs - A constructive approach. In *Application and Theory of Petri Nets and Concurrency - 34th International Conference, PETRI NETS 2013, Milan, Italy, June 24-28, 2013. Proceedings*, pages 311–329, 2013.

[64] Sander J. J. Leemans, Dirk Fahland, and Wil M. P. van der Aalst. Discovering block-structured process models from incomplete event logs. In *Application and Theory of Petri Nets and Concurrency - 35th International Conference, PETRI NETS 2014, Tunis, Tunisia, June 23-27, 2014. Proceedings*, pages 91–110, 2014.

[65] Leib Litman, Jonathan Robinson, and Cheskie Rosenzweig. The relationship between motivation, monetary compensation, and data quality among us- and india-based workers on mechanical turk. *Behavior Research Methods*, 47(2):519–528, 2015.

[66] G. Little, L.B. Chilton, M. Goldman, and R.C. Miller. Exploring iterative and parallel human computation processes. In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, HCOMP'10, pages 68–76, New York, NY, USA, 2010. ACM.

[67] Greg Little, Lydia B. Chilton, Max Goldman, and Robert C. Miller. Turkit: Human computation algorithms on mechanical turk. In *Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology*, UIST '10, pages 57–66, New York, NY, USA, 2010. ACM.

[68] Christoph Lofi and Kinda El Maarry. Design patterns for hybrid algorithmic-crowdsourcing workflows. In *16th IEEE Conf. on Business Informatics (CBI)*, Geneva, Switzerland, 07/2014 2014.

[69] Xixi Lu, Dirk Fahland, Frank J. H. M. van den Biggelaar, and Wil M. P. van der Aalst. Handling duplicated tasks in process discovery by refining event labels. In *Business Process Management - 14th International Conference, BPM 2016, Rio de Janeiro, Brazil, September 18-22, 2016. Proceedings*, pages 90–107, 2016.

[70] Nuno Luz, Nuno Silva, and Paulo Novais. A survey of task-oriented crowdsourcing. *Artificial Intelligence Review*, 44(2):187–213, 2015.

[71] Therani Madhusudan, J. Leon Zhao, and Byron Marshall. A case-based reasoning framework for workflow model management. *Data Knowl. Eng.*, 50(1):87–115, 2004.

[72] Andrew Mao, Ariel D. Procaccia, and Yiling Chen. Better human computation through principled voting. 2013.

[73] Winter Mason and Duncan J. Watts. Financial incentives and the "performance of crowds". In *Proceedings of the ACM SIGKDD Workshop*

on Human Computation, HCOMP '09, pages 77–85, New York, NY, USA, 2009. ACM.

[74] Adria Alcala Mena and Francesc Rosselló. Ternary graph isomorphism in polynomial time, after luks. *CoRR*, abs/1209.0871, 2012.

[75] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546, 2013.

[76] V. Muntés-Mulero, P. Paladini, J. Manzoor, A. Gritti, J.L. Larriba-Pey, and F. Mijnhardt. Crowdsourcing for industrial problems. In *Citizen Sensor Networks*, volume 7685 of *Lecture Notes in Artificial Intelligence*, pages 6–18. Springer, 2013.

[77] Victor Muntés-Mulero, Patricia Paladini, Jawad Manzoor, Andrea Gritti, Josep-Lluís Larriba-Pey, and Frederik Mijnhardt. Crowdsourcing for industrial problems. In Jordi Nin and Daniel Villatoro, editors, *Citizen in Sensor Networks*, volume 7685 of *Lecture Notes in Computer Science*, pages 6–18. Springer Berlin Heidelberg, 2013.

[78] Y. Narukawa and V. Torra. Twofold integral and multi-step choquet integral. *Kybernetika*, 40(1):39–50, 2004.

[79] M. Negri, L. Bentivogli, Y. Mehdad, D. Giampiccolo, and A. Marchetti. Divide and conquer: crowdsourcing the creation of cross-lingual textual entailment corpora. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 670–679,

Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[80] Stefanie Nowak and Stefan Rüger. How reliable are annotations via crowdsourcing: A study about inter-annotator agreement for multi-label image annotation. In *Proceedings of the International Conference on Multimedia Information Retrieval*, MIR '10, pages 557–566, New York, NY, USA, 2010. ACM.

[81] David Oleson, Alexander Sorokin, Greg Laughlin, Vaughn Hester, John Le, and Lukas Biewald. Programmatic gold: Targeted and scalable quality assurance in crowdsourcing. In *Proceedings of the 11th AAAI Conference on Human Computation*, AAAIWS'11-11, pages 43–48. AAAI Press, 2011.

[82] Carl Adam Petri. *Communication with automata*. PhD thesis, Universitï¿t Hamburg, 1966.

[83] Artem Polyvyanyy, Matthias Weidlich, Raffaele Conforti, Marcello La Rosa, and Arthur H. M. ter Hofstede. The 4c spectrum of fundamental behavioral relations for concurrent systems. In *PETRI NETS 2014, Tunis, Tunisia, June 23-27*, pages 210–232, 2014.

[84] Hernán Ponce de León, César Rodríguez, Josep Carmona, Keijo Heljanko, and Stefan Haar. Unfolding-based process discovery. In *Automated Technology for Verification and Analysis - 13th International Symposium, ATVA 2015, Shanghai, China, October 12-15, 2015, Proceedings*, pages 31–47, 2015.

[85] Cyrus Rashtchian, Peter Young, Micah Hodosh, and Julia Hockenmaier. Collecting image annotations using amazon's mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, CSLDAMT '10, pages 139–147, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[86] F. James Rohlf Robert R. Sokal. The comparison of dendrograms by objective methods. *Taxon*, 11(2):33–40, 1962.

[87] Jeffrey Rzeszotarski and Aniket Kittur. Crowdscape: Interactively visualizing user behavior and output. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, UIST '12, pages 55–62, New York, NY, USA, 2012. ACM.

[88] R. Schiaffino and F. Zearo. Developing and using a translation quality index. *Multilingual*, July/August 2006.

[89] T. Schulze, D. Nordheimer, and M. Schader. Worker perception of quality assurance mechanisms in crowdsourcing and human computation markets. In *19th Americas Conference on Information Systems 2013 : AMCIS 2013 Proceedings*, Atlanta, Ga., 2013. AISeL.

[90] Victor S. Sheng, Foster Provost, and Panagiotis G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 614–622, New York, NY, USA, 2008. ACM.

[91] Chong Sun, Narasimhan Rampalli, Frank Yang, and AnHai Doan. Chimera: Large-scale classification using machine learning, rules, and crowdsourcing. *PVLDB*, 7(13):1529–1540, 2014.

[92] Li Sun, Serdar Boztas, Kathy Horadam, Asha Rao, and Steven Versteeg. Analysis of user behaviour in accessing a source code repository. Technical report, RMIT University and CA Technologies, 2013.

[93] Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. Learning sentiment-specific word embedding for twitter sentiment classification. In *ACL (1)*, pages 1555–1565, 2014.

[94] Niek Tax, Natalia Sidorova, Reinder Haakma, and Wil M. P. van der Aalst. Event abstraction for process mining using supervised learning techniques. *CoRR*, abs/1606.07283, 2016.

[95] P. S. Thiagarajan. Elementary net systems. In *Advances in Petri Nets 1986, Part I on Petri Nets: Central Models and Their Properties*, pages 26–59, London, UK, UK, 1987. Springer-Verlag.

[96] V. Torra. Twofold integral: A choquet integral and sugeno integral generalization. *Butlletí de l'Associació Catalana d'Intel·ligència Artificial*, 29:13–19 (in Catalan). Preliminary version: IIIA Research Report TR–2003–08 (in English)., 2003.

[97] V. Torra and Y. Narukawa. *Modeling decisions: Information Fusion and Aggregation Operators*. Springer, 2007.

[98] M. Sugeno V. Torra, Y. Narukawa. *Non-Additive Measures*. Springer, 2014.

[99] W. M. P. van der Aalst. The application of petri nets to workflow management. *Journal of Circuits, Systems and Computers*, 8(1):21–66, 1998.

[100] Wil Van der Aalst, Arya Adriansyah, and Boudewijn van Dongen. Replaying history on process models for conformance checking and performance analysis. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(2):182–192, 2012.

[101] Wil M. P. van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes.* Springer Publishing Company, Incorporated, 1st edition, 2011.

[102] Wil M. P. van der Aalst and Christian W. Günther. Finding structure in unstructured processes: The case for process mining. In *ACSD*, pages 3–12. IEEE Computer Society, 2007.

[103] Wil M. P. van der Aalst, Michael Rosemann, and Marlon Dumas. Deadline-based escalation in process-aware information systems. *Decis. Support Syst.*, 43(2):492–511, March 2007.

[104] Borja Vázquez-Barreiros, Manuel Mucientes, and Manuel Lama. Mining duplicate tasks from discovered processes. In *Proceedings of the ATAED Workshop*, pages 78–82, 2015.

[105] P. Venetis and H. Garcia-Molina. Quality control for comparison microtasks. In *Proceedings of the First International Workshop on Crowdsourcing and Data Mining*, CrowdKDD '12, pages 15–21, New York, NY, USA, 2012. ACM.

[106] Luis von Ahn and Laura Dabbish. Labeling images with a computer game. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, pages 319–326, New York, NY, USA, 2004. ACM.

[107] Matthias Weidlich, Jan Mendling, and Mathias Weske. Efficient consistency measurement based on behavioral profiles of process models. *IEEE Tr. Soft. Eng.*, 37(3):410–429, 2011.

[108] Matthias Weidlich, Artem Polyvyanyy, Jan Mendling, and Mathias Weske. Causal behavioural profiles - efficient computation, applications, and evaluation. *Fundam. Inform.*, 113(3-4):399–435, 2011.

[109] Mathias Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer Publishing Company, Incorporated, 1st edition, 2010.

[110] R. Yager. On ordered weighted averaging aggregation operators in multi-criteria decision making. *IEEE Transactions on System, Man, and Cybernetics*, 18:183–190, 1988.

[111] R. Yager. Applications and extensions of OWA aggregations. *International Journal Man-Machine Studies*, 37:103–122, 1992.

[112] Zhiqiang Yan, Remco M. Dijkman, and Paul W. P. J. Grefen. Fast business process similarity search. *Distributed and Parallel Databases*, 30(2):105–144, 2012.

[113] O.F. Zaidan and C. Callison-Burch. Crowdsourcing Translation: Professional Quality from Non-Professionals. In *Proceedings of the 49th*

Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pages 1220–1229, Portland, Oregon, USA, June 2011.