# MailOfMine – Analyzing Mail Messages for Mining Artful Collaborative Processes

Claudio Di Ciccio[1], Massimo Mecella[1],
Monica Scannapieco[2], Diego Zardetto[2], and Tiziana Catarci[1]

[1] Sapienza – Università di Roma, Dipartimento di Informatica e Sistemistica
Via Ariosto 25, Roma, Italy
{catarci|cdc|mecella}@dis.uniroma1.it
[2] Istituto Nazionale di Statistica
Via Balbo 16, Roma, Italy
{scannapi|zardetto}@istat.it

**Abstract.** Artful processes are informal processes typically carried out by those people whose work is mental rather than physical (managers, professors, researchers, engineers, etc.), the so called "knowledge workers". In this paper we propose the MailOfMine approach to automatically build, on top of a collection of *e*-mail messages, a set of workflow models that represent the artful processes which lay behind the knowledge workers activities.

**Keywords:** object matching, process mining, text mining, *e*-mail analysis, knowledge workers, artful process, declarative workflows

## 1 Introduction

For a long time, formal business processes have been the main subject of workflow related research. Only few papers addressed the study of informal processes, often under names such as "artful processes" [**?**]: informal processes are typically carried out by those people whose work is mental rather than physical (managers, professors, researchers, engineers, etc.), the so called "knowledge workers" [**?**]. With their skills, experience and knowledge, they are used to perform difficult tasks, which require complex, rapid decisions among multiple possible strategies, in order to fulfill specific goals. In contrast to business processes that are formal and standardized, often informal processes are not even written down, let alone defined formally, and can vary from person to person even when those involved are pursuing the same objective. Knowledge workers create informal processes "on the fly" to cope with many of the situations that arise in their daily work. Though informal processes are frequently repeated, they are not exactly reproducible even by their originators – since they are not written down – nor can they be easily shared. Their outcome releases and their information exchanges are very often done by means of *e*-mail conversations, which are a fast, reliable, permanent way of keeping track of the activities that they fulfill.

Understanding artful processes involving knowledge workers is becoming crucial in many scenarios, here we mention some of them:

– *personal information management (PIM)*, i.e., how to organize one's own activities, contacts, etc. through the use of software on laptops and smart

devices (iPhone/iPad, smartphones, tablets). Here, inferring artful processes in which a person is involved allows the system to be proactive and to drive the user through its own tasks (on the basis of past performances) [**?**,**?**];
- *information warfare*, in particular in supporting anti-crime intelligence agencies: let us suppose that a governative bureau is able to access the *e*-mail account of a suspected person. People planning a crime or an act out of law are used to speak a language of their own to express duties and next moves, where meanings do not match with the common sense. Though, a system should build the processes that lay behind their communications anyway, exposing the activities and the role of the actors. At that point, translating the sense of misused words becomes an easier task for investigators, and allows inferring the criminal activities of the suspected person(s);
- *enterprise engineering*: in design and engineering, it is important to preserve more than just the actual documents making up the product data. For knowledge-heavy industries it is of critical importance to also preserve the "soft knowledge" of the overall process, the so-called product lifecycle. The idea here is not only to send the designs into the future, but also the knowledge about processes, decision making, and people [**?**,**?**,**?**].

The objective of the approach proposed in this paper is to automatically build, on top of a collection of *e*-mail messages, a set of workflow models that represent the artful processes which lay behind the knowledge workers activities. There are many advantages out of this work. First of all, the unspecified agile processes that are autonomously used become formalized: since such models are not defined *a priori* by experts but rather inferred from real-life scenarios that actually took place, they are guaranteed to respect the true executions and not reflect the expected ones (often Business Process Management tools are used to show the discrepancy between the theoretical workflows and the concrete ones). Moreover, such models can be shared, compared, preserved, so that the best practices might be put in evidence from the community of knowledge workers, to the whole business benefit. Finally, an analysis over such processes can be done, so that bottlenecks and delays in actual executions can be found out.

The approach we would like to pursue, in order to retrieve a collection of process models out of an initial set of *e*-mail messages, involves many research fields at a time, each concerning a phase of the overall processing. We make use of *object matching* algorithms to obtain clusters of related *e*-mail conversations, from the previous extracted subset. Every cluster is subsequently treated by *text mining information extraction* procedures, in order to find out which tasks *e*-mail messages are about. *process mining* is used to abstract process models representing the workflows, which the sets of subsumed tasks were considered traces of. Our approach, named MailOfMine, is being validated on a corpus of about 10 Gigabyte of *e*-mail messages, derived from the activity of one of the authors in about 10 years of works in research projects, in order to infer common processes that partners adopted during software/deliverables' production. A more extensive validation is planned to be performed on enterprise engineering data.

## 2 The Process Model

In this Section, we present the process model adopted for describing the mined artful processes. We will clarify it through an example and we further discuss on technical and theoretical implications of our assumptions.

**Definition 1 (Actor).** *An* actor *is the subject directly or indirectly taking part in the progress of a work. She is called* contributor *when her collaboration is either proven by the participation to the collaborative communications or by the production of outcomes. Otherwise, she is named* spectator.

The spectator is the person who receives messages but does not reply, i.e., is among the recipients but is never a sender during the discussion. Nonetheless, her presence is perhaps fundamental, since she is a stakeholder, or a manager controlling the thread though having no reason to intervene ("no news, good news").

**Definition 2 (Task).** *A* task *is an elementary unit of work. Each Task is connected to* (i) *its expected* duration, (ii) *zero or more* outcome*s, and* (iii) *one or more* actor*s. A task is called* productive *when linked to one or more outcomes. Otherwise, it is named* clarifying.

Clarifying tasks are not less important than productive ones. Often they are crucial to define, e.g., aim and methodology of the further steps. The outcome is whatever adds a bit of information to the exchanged knowledge among actors. In our context we simplify the definition by considering *document oriented* outcomes, i.e., we consider as new products of the tasks only those documents which are attached to the *e*-mail body.

**Definition 3 (Activity).** *An* activity *is a collection of tasks or (recursively) other activities.*

**Definition 4 (Key Part).** *A* key part *is each* unique *piece of text belonging to the* e-*mail messages exchanged in a communication trace. Thus, given a collection of duplicated pieces of text (coming from the same or different* e-*mail messages in the trace), just a single representative is selected as key part.*

For instance, HTML signatures used by the sender of the *e*-mail, quotations of previous *e*-mail messages used in replies, etc., are all examples of redundant information that may appear in a communication trace, but will be filtered out by means of the *key part* concept. On the other hand, any piece of text not appearing in any other *e*-mail message in the discussion thread will be interpreted as *key part.*

**Definition 5 (Indicium).** *An* Indicium *is any communication trace, or part of it, attesting the execution of a task, an activity, or a process instance.*

Indicia are key parts sets (or sets of key parts sets) containing any evidence that a task, or an activity, or a process instance, has been performed or is being performed.

**Definition 6 (Process Scheme (Process) and Process Describing Grammar ($\mathcal{PDG}$)).** *A* process scheme *(or* process *for short) is a semi-structured set of activities, where the semi-structuring connective tissue is represented by the set of constraints stating the interleaving rules among activities or tasks. Such a set of constraints, formulated on top of activities and tasks, is named* Process Describing Grammar *($\mathcal{PDG}$).*

Constraints do not force the tasks to follow a tight sequence, but rather leave them the flexibility to follow different paths, to terminate the enactment, though respecting a set of rules that avoid illegal or non-consistent states in the execution.

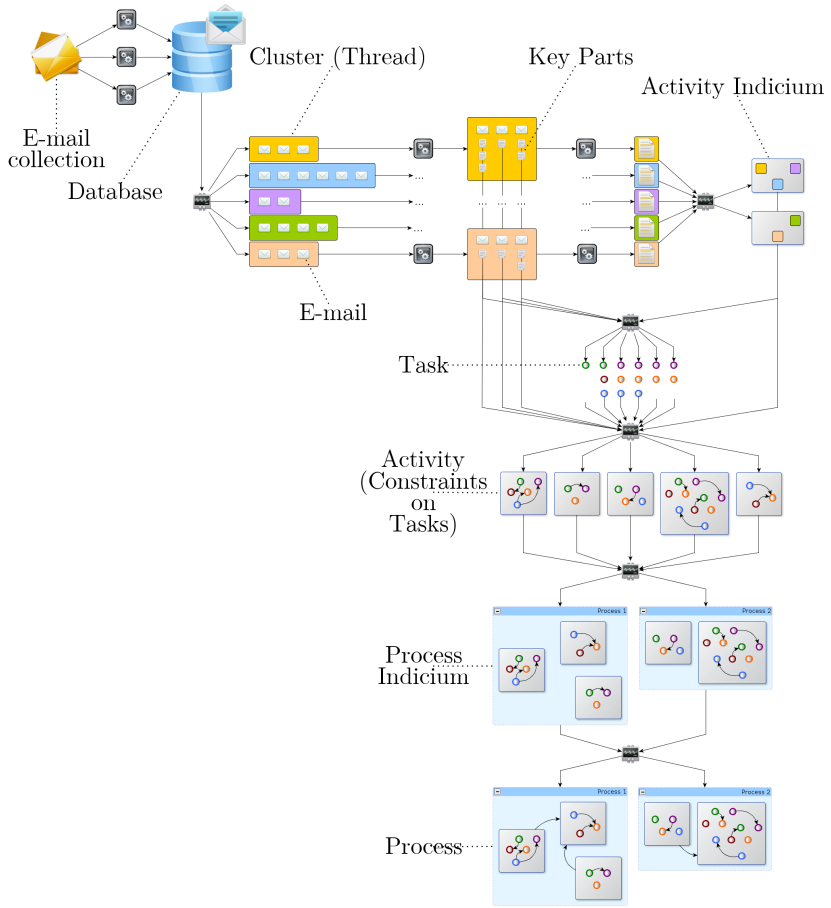


Fig. 1: The MailOfMine approach

### 2.1 On the Process Describing Grammar ($\mathcal{PDG}$)

Let us consider every possible task [3] in a process as a character of an alphabet. Activities are thus structured actions which are either allowed or not to be executed in a process, as characters may or may not appear inside a string. The execution of some activities can imply others to be implied, at any point in time, as well as production rules enable the modification of the string expression in case a given sequence of characters is reached.

In this paper, we argue that each constraint in the set which can be used to define an artful mined process is expressible through regular grammars. Regular grammars are recognizable through Finite State Automata (FSA) [**?**] (either deterministic or non-deterministic [**?**]). Constraints on the process always hold and must be respected altogether at each point in time. This means that the intersection of all the constraints must always hold. In other words, the FSA recognizing the correct traces for processes (i.e., accepting the valid strings) is the intersection of all the FSAs composing set of constraints (it is known that such grammars are closed to the operation of intersection ([**?**]). Thus, we consider each task as a terminal character in the $\mathcal{PDG}$. Each constraint on tasks is a $\mathcal{PDG}$ itself. An activity is thus a $\mathcal{PDG}$ built as the intersection of all of the constraints' $\mathcal{PDG}$s.

In order to define the process scheme, we shift to regular expressions, which are an equivalent way to describe regular grammars (and accepting FSAs as well). Regular expressions are closed to their operators, thus we are able to recursively define activities as the intersection of constraints on activities. The process scheme, in turn, is the intersection of constraints on activities.

We finally summarize the concepts introduced so far, from a hierarchical recursive point of view. Tasks are terminal characters, building blocks of constraints on tasks. Constraints are regular expressions, equivalent to regular grammars. Activities are regular expressions which are equivalent to the intersection of the constraints' regular grammars. Constraints can be formulated on top of activities, being regular expressions themselves. The process scheme is the intersection of constraints defined on top of activities.

### 2.2 An Example

Let us suppose to have an *e*-mail archive, containing various process instances indicia, and to focus specifically on the planning of a new meeting for a research project. We suppose to execute the overall technique explained in Section **??**, and we report the possible result, starting from the list of tasks in activities (Process Description **??**).

The "bookFlight" task is supposed to be retrieved by confirmation *e*-mail messages received by the user when the booking is completed. The $\langle Flight \rangle$ activity is composed by this task only. $\langle Agenda \rangle$ is supposed to be slightly more complex instead. We suppose that a final agenda will be committed ("confirmAgenda") after that requests for a new proposal ("requestAgenda"), proposals

[3] In the following, we focus on task identifiers, omitting, for sake of readability, details about duration, actors and outcomes. Hence, we identify tasks by their names only.

**Process Description 1** Activities and tasks list

Activity: $\langle Flight \rangle$
  Task: $boFl$ ("bookFlight"): Productive
    Actors: {$You$: Contributor, $FlightCompany$: Contributor}
    Duration: 2 hrs.
Activity: $\langle Agenda \rangle$
  Task: $prAg$ ("proposeAgenda"): Productive
    Actors: {$You$: Contributor, $Community$: Spectator}
    Duration: 4 dd.
  Task: $rqAg$ ("requestAgenda"): Clarifying
    Actors: {$Participant$: Contributor, $Community$: Spectator}
    Duration: $\perp$
  Task: $coAg$ ("commentAgenda"): Clarifying
    Actors: {$Participant$: Contributor, $Community$: Spectator}
    Duration: $\perp$
  Task: $cnAg$ ("confirmAgenda"): Productive
    Actors: {$You$: Contributor, $Community$: Spectator}
    Duration: 2 dd.
Activity: $\langle Accommodation \rangle$
  Task: $esHo$ ("establishHotel"): Productive
    Actors: {$Organizer$: Contributor, $Community$: Spectator}
    Duration: $\perp$
  Task: $boHo$ ("bookHotel"): Productive
    Actors: {$You$: Contributor, $Hotel$: Contributor}
    Duration: 2 dd.

themselves ("proposeAgenda") and comments ("commentAgenda") have been circulated in the group ($Community$, as identified in the Actors list). Finally, $\langle Accommodation \rangle$ is the activity related to the reservation of rooms, in a hotel located where the meeting will take place (or nearby). Beyond the confirmation $e$-mail messages received by the user when the booking is completed (indicia for the "bookHotel" task), the "establishHotel" stems from the $e$-mail messages informing about the hotel arranged by the meeting organizers.

The reader may notice that sometimes $You$ appears among the Actors. We suppose it to be the special identifier adopted whenever the owner of the $e$-mail archive is a Contributor. In case she is only a Spectator (see, e.g., "requestAgenda"), such an information is not reported, since it is redundant (if the owner were not among the recipients, there should have been no way to access $e$-mail messages related to that task). The usage of $\perp$ as Duration stands for an unknown value: whenever there is only one indicium related to a task (i.e., only one $e$-mail message proving its execution, for each activity indicium), no inference about Duration can be performed.

The aforementioned tasks and activities are bound to the following constraints. In order to express them, we use the base set introduced in [**?**], starting with the *existence* constraints of Process Description **??**, which specify how many times (from a minimum to a maximum) a task can be performed when executing an activity.

The same constraint can be formulated on activities. In this case, if the constrained activity is part of a more comprehensive activity (considering a containment hierarchy), it specifies how many times (from a minimum to a maximum[4]) an activity can be executed in the context of the containing activity execution.

[4] Where $*$ stands for an unbounded, though finite, value.

Otherwise, it specifies how many times (from a minimum to a maximum) an activity can be executed in the overall process execution.

In this example, we have no containment hierarchy, thus Constraints on activities refer to the execution context of the process.

**Process Description 2** Existence constraints on the example tasks and activities

Activity: $\langle Flight \rangle$ : [0, $*$]
  Task: $boFl$ : [1, 2]
Activity: $\langle Agenda \rangle$ : [1, 1]
  Task: $prAg$ : [0, $*$]
  Task: $rqAg$ : [0, $*$]
  Task: $coAg$ : [0, $*$]
  Task: $cnAg$ : [1, 1]
Activity: $\langle Accommodation \rangle$ : [0, 1]
  Task: $esHo$ : [0, 1]
  Task: $boHo$ : [1, 1]

In Process Description **??** we report the relation constraints holding in this example process. We recall here [**?**]:

$initial(a)$ implies $a$ to be the first task (activity) for the containing activity (or process);
$final(a)$ implies $a$ to be the last task (activity) for the containing activity (or process);
$response(a, b)$ implies $b$ to appear in the activity (or process) execution after $a$ is performed, if $a$ is performed;
$respondedExistence(a, b)$ implies $b$ to appear in the activity (or process) execution if $a$ is performed, no matter when;
$succession(a, b)$ implies $b$ to appear in the activity (or process) after $a$ is performed, and no $b$ can be executed before $a$ is done;
$coExistence(a, b)$ implies $b$ to appear in the activity (or process) if $a$ is performed, no matter when, and viceversa ($a$ must be performed if $b$ is done).

**Process Description 3** Relation constraints on the example tasks and activities

Activity: $\langle Flight \rangle$
  $initial(boFl)$
  $final(boFl)$
Activity: $\langle Agenda \rangle$
  $final(confirmAgenda)$
  $response(rqAg, prAg)$
  $respondedExistence(coAg, prAg)$
  $succession(prAg, cnAg)$
$coExistence(\langle Flight \rangle, \langle Accommodation \rangle)$
$respondedExistence(\langle Agenda \rangle, \langle Accommodation \rangle)$

The reader should note that in the above process description, for sake of conceiveness, we did not report either the $\mathcal{PDG}$, nor the FSAs. However, here

we make the mapping explicitly by showing the translations of such constraints into regular expressions (corresponding to $\mathcal{PDG}$s):

| | |
|---|---|
| $initial(a)$ | `^a.*$` |
| $final(a)$ | `^.*b$` |
| $response(a, b)$ | `^[^a]*(a[^b]*b)*[^a]*$` |
| $respondedExistence(a, b)$ | `^[^a]*((a[^b]*b)\|(b[^a]*a))*[^a]*$` |
| $succession(a, b)$ | `^[^a^b]*(a[^b]*b)*[^a]*$`; |
| $coExistence(a, b)$ | `^[^a^b]*(a[^b]*b)*[^a^b]*$` |

## 3 The MailOfMine Approach

The MailOfMine approach (and the tool we are currently developing) adopts a modular architecture, the components of which allow to incrementally refine the mining process. We describe it into three parts: *(i)* Section **??** shows the preliminary steps, from the retrieval of *e*-mail messages to the reconstruction of communication threads; *(ii)* Section **??** draws the extraction of key parts, the activities and tasks indicia detection, and the tasks definition; *(iii)* Section **??** outlines the final steps, from the activities definition to the final mined process extraction.

### 3.1 Preliminary Steps and *E*-mail Clustering

Initially we need to extract *e*-mail messages from the given archive(s). Archives are compliant to different standards, according to the *e*-mail client in use, hence the component for reading the messages is intended to be plug-in based: each plug-in is a component able to access different archive formats (e.g., `.pst`, Mozilla Thurderbird files, etc.) and/or online providers (e.g., Gmail accounts, etc.). The outcome is the population of a database, on the basis of which all the subsequent steps are carried out. The first of them is the clustering of retrieved messages into extended communication threads, i.e., flows of messages which are related to each other. The considered technique, detailed in Section **??**, which is used to guess such a connection, is based not only on the Subject field (e.g., looking at "Fwd:" or "Re:" prefixes) or SMTP Header fields (e.g., reading the "In-Reply-To" field), but on the application of a more complex object matching decision method. Such indicators, indeed, though likely trustworthy, might be misleading: *(i)* in the everyday life, it is a common attitude to reply to old messages for talking about new topics, which may have nothing to do with the previous one; *(ii)* conversely, it may happen that a new message is related to a previous, though written and sent as if it were a new one.

Once the communication threads are recognized, we can assume them all as activity indicia candidates.

After the clustering into threads, messages are analyzed in order to identify key parts. Key parts are identified by adding to the [**?**] technique for the removal of quoted material, an iterative approach over the *e*-mail messages in the thread. Such an approach is based on the Unix `diff` command, performed between a given *e*-mail and the key parts identified so far.

### 3.2 Identifying Activities

Once key parts and messages in threads are gathered, MailOfMine can build activity indicia as the concatenation of all the key parts. Then, the clustering algorithm is used again, this time to identify the matches between activity indicia. E.g., let us suppose to have *(i)* a thread $T_{del41}$ related to the writing of Deliverable 4.1 of a research project, *(ii)* another thread $T_{del52}$ related to the writing of Deliverable 5.2 of the same research project, and, finally, *(iii)* a $T_{air}$ dealing with an airplane flight booking, for the next review meeting. Thus, the algorithm is expected to cluster as follows: $\{T_{del41}, T_{del52}\}, \{T_{air}\}$. Its output represent the activities set.

By taking into account the activities set and the key parts (task indicia candidates), the clustering algorithm checks for matching key parts, identifying them as tasks. E.g., let us suppose to have *(i)* a key part $k_{airData}$ specifying the booked airplane data, *(ii)* another key part $k_{airBook}$ containing the confirmation of the booking, and, finally, *(iii)* a $k_{del41}$: "Please find attached Deliverable 4.1". Thus, the algorithm is expected to cluster as follows: $\{k_{airData}, k_{airBook}\}, \{k_{del41}\}$.

Hence, each set is a task indicium for one task. The analysis, though, does not terminate here. In order to understand whether the task under investigation is clarifying or productive, MailOfMine checks *(i)* if any document oriented outcome in the original *e*-mail messages was attached; *(ii)* if key parts contain Speech Acts ([**?**]), according to the technique proposed by [**?**]. If at least one of the listed conditions holds, it is productive. Otherwise, it is considered as clarifying. The detection of Speech Acts is supposed to be assisted by the experts, who are required to provide a dictionary of keywords of their domain field, as in [**?**]. For further information on the relation between Speech Acts and workflows, see [**?**].

Taking as input all of the preceding outcomes, MailOfMine starts searching for execution constraints between tasks within the activities they belong to. For this step, we exploit and extend the SPIRIT techniques for regular patterns mining ([**?**]). Specifically, on the basis of all the possible constraints, a selection of those that are valid over most of the activity indicia is made. The selected constraints for each activity are its $\mathcal{PDG}$.

### 3.3 Mining the Process

Once activities and tasks are recognized, a supervised learning process takes place, in order to cluster activities into processes. This step cannot be fully performed by the system, since no linguistic connection could exist related activities. E.g., the activity of drawing slides and the reservation of the airplane could sound completely apart, though those slides could be the material to present at a review meeting, hold in the city which the airplane was booked to reach.

The activity grammar should not be exposed to the user building the test and verification set, of course. Instead, the threads are shown as witnesses for the activity; hence, the user associates part of them to different processes, and the learner associates all the related activities to processes.

Once processes are identified, MailOfMine performs the second step for the construction of the $\mathcal{PDG}$, i.e., the mining of production rules among activities

inside the same process, by using the SPIRIT techniques for regular pattern mining again.

## 4 E-mail Clustering

In this section we describe how we tackled the problem of performing a Similarity Clustering (SC) of *e*-mail messages. The purpose of such a clustering step is to identify groups of related *e*-mail messages to be used for the subsequent steps aimed at identifying tasks composing process activities. To face the SC problem, we exploited a previously proposed Object Matching (OM) algorithm [**?**], providing ad-hoc extensions for the task at hand. Here we briefly illustrate the rationale for adopting an OM algorithm to cluster *e*-mail messages. From an abstract point of view, OM and SC share the common objective of classifying data-object pairs according to an hidden (i.e. not self-evident) property; for OM, the relevant hidden property to be discovered is if two objects *"do represent – or not – the same real-world entity"*, whereas, for SC, one needs to asses if two objects *"do belong – or not – to the same group"*. Moreover, in order to *infer* the class the pairs belong to, both OM and SC rely on pairwise distance (or, equivalently, similarity) measures. With respect to the choice of the specific OM algorithm [**?**], we point out that it relies on its fully automated nature and on its independence on the specific data object representation. Lastly, we remark that the statistical properties at the basis of the chosen algorithm are perfectly valid for the SC problem.

We implemented our Similarity Clustering as a four steps procedure:

1. In the first step, we chose a representation format for *e*-mail messages, in order to translate them into processable data objects.
2. The second step consisted of the definition of a specific distance metric to compare *e*-mail objects.
3. As a third step, we ran a decision algorithm whose output was a set of *e*-mail pairs declared as "matches", i.e. – in the present context – belonging to the same cluster.
4. The fourth and last step implied the application of a function performing a transitive closure among "match" pairs in order to build *e*-mail clusters.

This procedure is detailed in the following together with some preliminary test results.

### 4.1 E-mail Object Representation and Distance Metric Definition

In order to compare *e*-mail objects we first defined a representation format for each *e*-mail message. Specifically, we considered each *e*-mail as a record consisting of: *(i)* some header fields, *(ii)* the body and *(iii)* the names of attached files (if present). According to RFC 5322 and RFC 3864 (`http://www.ietf.org/rfc.html`), each message has exactly one header, which is structured into fields. Some of these fields are mandatory (such as the *message_id*), others are instead optional but indicated as common. Among such common header fields, we took

into account the ones listed in the following; for each of them we also specify the chosen representation format:

- $emailIdentifier$: this is a string associated to each $e$-mail that permits its unique identification.
- $Sender$: $e$-mail address of the sender represented as a string.
- $Receivers$: $e$-mail addresses represented as strings and concatenated into a unique string.
- $Subject$: represented as one single string.

As far as the body, we choose to consider it as a piece of free text and we represented it as one single string. Finally, $e$-mail attachments were taken into account by considering their names. More specifically, we represented attachments by a unique string resulting from the concatenation of the names of all the present attachment files. Having as objective to build clusters consisting of $e$-mail exchanged to accomplish a specific task, we found reasonable to merge $Sender$ and $Receivers$ fields into one single field. This is because the clusters we want to determine are, in a sense, *invariant* with respect to the $e$-mail exchange direction, as they are instead focused on the actions performed by means of such $e$-mail exchanges.

On the basis of the representation choices above described, all the $e$-mail fields but the body could be compared by means of traditional string metric distances (see [**?**] for a survey). In our experiments, we used the Levenshtein distance for these fields.

The body is an exception as it is a piece of free text rather than a (more or less) *structured* text field like the others[5]. Bodies were considered as a corpus of documents. As a first step, each body was processed and transformed into a set of 3-grams. Treating such 3-grams as "terms", we built a Term-Document matrix. To each 3-gram of a body a weight was assigned, based on the traditional TF-IDF metric. Each body was in this way represented by a vector of a vector space and hence easily comparable to the other bodies. To such a scope, we used the cosine distance function. 3-grams were chosen in alternative to words, because identification of words by means of specific separators is problematic for $e$-mail messages, as they are typically written without a deep consideration of syntactic rules. On the other hand, 3-grams do not imply any information loss as compared to words, and are processable for not so long texts like typical $e$-mail messages. TF-IDF was considered appropriate because of its ability to increase the importance of a term proportionally to the term frequency inside the document, while penalizing terms that are found to be very common in the corpus.

### 4.2 Preliminary Tests Results

As a first test-bed for our SC application, we processed 101 $e$-mail messages selected from a larger collection of about 10 GByte $e$-mail messages (this collection

[5] Though the subject can contain more than one words, we decided to consider it too as a structured field. Indeed, since typically subjects are not long unstructured texts, text mining techniques are unnecessary, or even not suitable, for their comparison.

has been gathered by one of the authors over 10 years of work on Italian and European research projects). Out of the 5050 pairs generated by the 101 input *e*-mail messages, our decision algorithm declared 98 *e*-mail-pairs to belong to the same group. The transitive closure performed on the set of such 98 "matching" *e*-mail-pairs determined 21 non overlapping and non trivial (that is containing at least 2 *e*-mail messages) clusters: these are the final output produced by our SC algorithm. The output clusters size ranges from 2 to 11 *e*-mail messages and the *e*-mail messages involved in the clusters are found to be 68 out of 101.

To assess the quality of the obtained result, we manually analyzed the input *e*-mail messages and the output clusters: we identified only 1 false positive cluster, whereas no false negatives were found. The false positive cluster involved two *e*-mail messages, which happened to have the same sender and receivers, and almost completely overlapping bodies. Specifically, the two bodies had very short texts (few words) but the same very long signature block of the sender. From these preliminary tests, we realized the need to take into account the "semantics" of the body. Our idea is to exploit a dictionary containing key terms relevant to the application domain of the *e*-mail messages to process. We could filter the Term-Document matrix on the basis of such a dictionary, and add to the distance also such a contribution. Notice that we do not intend to replace the body distance described above, but to *complement* it with a second component generated by a dictionary-aware run. The effect of this modification would be to decrease relatively (i.e. when compared to other pairs) the similarity between two bodies if their "semantics" are realized as unrelated with respect to the dictionary, as was the case for the described false positive example.

## 5 Related Work

*Text Mining*, or *Knowledge Discovery from Text (KDT)*, deals with the machine supported analysis of text: indeed, it refers generally to the process of extracting interesting information and knowledge from unstructured text. Text mining covers many topics that are out of the scope of this paper, see [**?**,**?**] for comprehensive surveys.

[**?**] proposes a method employing text mining techniques to analyze *e*-mail messages collected at a customer center. Based on [**?**], this work shows how to cope with the information extraction in the context of *e*-mail messages, for the construction of a key concept dictionary. Our aim is not restricted to the extraction of the key concept dictionary, but rather on the mining of activities performed on top of them; on the other side, in MailOfMine, we assume to rely on a user-provided dictionary of keywords, as described in Section **??**. [**?**] introduces the usage of an ontology of *e*-mail acts, i.e., Speech Acts [**?**], to classify messages.

*Process Mining*, a.k.a. *Workflow Mining* [**?**], is the set of techniques that allow the extraction of structured process descriptions, stemming from a set of recorded real executions. Such executions are intended to be stored in so called *event log*s, i.e., textual representations of a temporarily ordered linear sequence of tasks. There, each recorded *event* reports the execution of a *task* (i.e., a well-defined step in the workflow) in a *case* (i.e., a workflow instance). Events are

always recorded sequentially, even though tasks could be executed in parallel: it is up to the algorithm to infer the actual structure of the workflow that they are traces of, identifying the causal dependencies between tasks (*condition*s). ProM [**?**] is one of the most used plug-in based software environment for implementing workflow mining techniques.

Most of the mainstream process mining tools model processes as Workflow Nets (WFNs – see [**?**]), explicitly designed to represent the control-flow dimension of a workflow. From [**?**] onwards, many techniques have been proposed, in order to address specific issues: pure algorithmic (e.g., $\alpha$ algorithm, drawn in [**?**] and its evolution [**?**], $\alpha^{++}$), heuristic (e.g., [**?**]), genetic (e.g., [**?**]). Indeed, heuristic and genetic algorithms have been introduced to cope with noise, that the pure algorithmic techniques were not able to manage. A very smart extension to the previous research work has been recently achieved by the two-steps algorithm proposed in [**?**].

EMailAnalyzer [**?**] is an integrated ProM plug-in for mining processes from *e*-mail logs, that are XML files compatible with the ProM framework, built through the analysis of *(i)* senders and receivers, in order to guess the actors involved, and *(ii)* tags on *e*-mail messages and subjects, for extracting the task. *e*-mail messages are extracted from an Outlook archive. Our approach shares similar ideas for the disambiguation of actors and sociograms, and aims at extending it by *(i)* building a plug-in based platform capable to retrieve *e*-mail messages from multiple archives (not only Outlook), and *(ii)* extracting cases and relations among tasks from a more comprehensive analysis of *e*-mail fields (headers, threads, body, attachments, etc.).

The need for flexibility in the definition of processes leads to an alternative to the classical "imperative": the "declarative" approach. Rather than using a procedural language for expressing the allowed sequences of activities, it is based on the description of workflows through the usage of constraints: the idea is that every task can be performed, except what does not respect them. [**?**] shows how the declarative approach can help in obtaining a fair trade-off between flexibility in managing collaborative processes and support in controlling and assisting the enactment of workflows. Such constraints, in [**?**] (Chapter 6) are formulations of Linear Temporal Logic ([**?**] – Chapter 3). DecSerFlow [**?**] and ConDec [**?**] provide graphical representations for processes described through the declarative approach. Nonetheless, we believe that the declaration of collaborative workflows constraints can be expressed by means of regular expressions, rather than LTL formulae: regular expressions express finite languages (i.e., processes with finite traces, where the number of enacted tasks is limited). LTL formulae are thought to be used for verifying properties over semi-infinite runs instead. On the contrary, human processes have an end, other than a starting point. As a final consideration, we envision the $\mathcal{PDG}$ like a grammar describing the language spoken by collaborative organisms in terms of activities, thus being more related to formal languages rather than temporal logic.

The Declare framework [**?**] is a concrete implementation of a constraint-based process management system, supporting multiple declarative languages. Declare-miner is the plug-in for mining declarative processes from within the ProM framework. [**?**] described the usage of Inductive Logic Programming Tech-

niques to mine models expressed as a $\mathcal{S}$CIFF ([**?**]) theory, finally translated to the ConDec notation. Our approach, in the long term, aims at introducing a probabilistic component in the process models, as in [**?**].

## 6 Conclusions

In this paper we proposed the MailOfMine approach for mining artful processes from *e*-mail messages collections. We are currently in the process of realizing the various techniques into a working prototype and then validating it over a large *e*-mail messages collection.

Future work includes the extension of the approach to multiple users' *e*-mail messages collections, and a more extensive validation. Moreover, we plan to deal with privacy concerns that naturally arise when mining over personal *e*-mail messages. Indeed, it is an open issue how to perform private object matching, and the case of *e*-mail messages is particularly challenging due to the already limited amount of fields that can be used to perform the matching task.

Finally, this work aims at addressing the open research challenge of dealing with mixed logs, i.e., logs in which traces from multiple processes are present. Most of existing workflow mining approaches suppose to treat logs in which only traces of the same process are present; conversely, an *e*-mail collection can be abstracted as a log containing traces of different processes.

This document is a pre-print copy of the manuscript
(Di Ciccio et al. 2011)
published in the proceedings of SIMPDA.

## References

Di Ciccio, Claudio, Massimo Mecella, Monica Scannapieco, Diego Zardetto, and Tiziana Catarci (2011). "MailOfMine – Analyzing Mail Messages for Mining Artful Collaborative Processes". In: *SIMPDA*. Ed. by Karl Aberer, Ernesto Damiani, and Tharam Dillon, pp. 45–59. ISBN: 978-88-903120-2-1.

## BibTeX

```
@InProceedings{   DiCiccio.etal/SIMPDA2011:MailOfMineAnalyzingMail,
  author        = {Di Ciccio, Claudio and Mecella, Massimo and Scannapieco,
                   Monica and Zardetto, Diego and Catarci, Tiziana},
  title         = {{MailOfMine} -- Analyzing Mail Messages for Mining Artful
                   Collaborative Processes},
  booktitle     = {SIMPDA},
  year          = {2011},
  pages         = {45-59},
  crossref      = {SIMPDA2011},
  isbn          = {978-88-903120-2-1},
  keywords      = {object matching, process mining, text mining, email
                   analysis, knowledge workers, artful process, declarative
                   workflows}
}
@Proceedings{     SIMPDA2011,
  title         = {1st International Symposium on Data-Driven Process
                   Discovery and Analysis, SIMPDA 2011, Campione d'Italia,
                   Italy, June 29 - July 1st, 2011},
  year          = {2011},
  editor        = {Aberer, Karl and Damiani, Ernesto and Dillon, Tharam}
}
```