Paper draft - please export an up-to-date reference from
http://www.iet.unipi.it/m.cimino/pub
XIII Convegno dell'Associazione Italiana di Tecnologia Meccanica

AITEM 2017

Pisa, 11-13 Settembre 2017

A new perspective on Process-oriented Software Engineering based on BPMN Process Mining

Gionata Carmignani[a], Mario G.C.A. Cimino[b], Franco Failli[c,*], Antonella Santone[d], Gigliola Vaglini[b]

[a]*University of Pisa, Dept. Energy, System, Territory and Construction, Largo L. Lazzarino 1, Pisa 56122, Italy*
[b]*University of Pisa, Dept. Information Engineering, Largo L. Lazzarino 1, Pisa 56122, Italy*
[c]*University of Pisa, Dept. Civil and Industrial Engineering, Largo L. Lazzarino 1, Pisa 56122, Italy*
[d]*University of Sannio, Dept. Engineering, Via Traiano 1, Benevento 82100, Italy*

Abstract

Nowadays, the quality of an IT product/service is increasingly based on the development process, not only on development costs and on time. This trend fosters the software process quality certification, focusing on the entire life cycle, and the adoption of agile software development processes, which rely on a self-organizing culture for creating tailored products. In parallel, the business of IT companies is evolving into a "software factory" model, to increase productivity and reduce development costs. Accordingly, the development of a software project is distributed among different factories, depending on their specialties and workload. This highly industrialized development process requires appropriate methods for management control and accountability. Different factories, and different project managers in the same factory, may adopt different software development processes, and a number of factors may disrupt the process execution. In the software production domain, there is a large availability of systems recording audit trails about process execution. A fundamental challenge is to assess from audit trails if an IT solution is developed through the appropriate procedures, as well as to discover the procedures actually adopted without imposing any management practice. In this study, we adopt Process Mining techniques based on the Business Process Model and Notation (BPMN) for the automatic discovery of the software development processes. Our approach can be used to measure the alignment of people and processes, to investigate causes of disruption, as well as to generate a process model from audit trails. The paper illustrates the proposed techniques and discusses their application through a pilot real-world case study.

Keywords: Business Process Engineering, Process-oriented Software Engineering, BPMN, Process Mining, Conformance Analysis.

1. Introduction and background

A large number of methodologies for managing IT projects have evolved over the years, ranging from prescriptive steps used in day-to-day work to frameworks generating flexible procedures for many projects or groups[1,2]. A system to assess if an IT solution is developed through the appropriate procedures, or to discover the

* Corresponding author. Tel.: +39-050-221-8133; fax: +39-050-221-8140.
E-mail address: franco.failli@unipi.it

procedures actually adopted, is then fundamental for software process engineering[3,4]. The software engineering process usually comprises a number of phases and activities, which often are specific to the process model and to the subject area. Implementing an IT project requires careful planning, methodical and deliberated approach, as well as a necessary level of commitment, time and resources, for a number of reasons[5]. First, usually, not all key stakeholders have the appropriate levels of education and awareness. Second, many organizations partially manage the inter-related processes of an IT project. Third, often project managers are not well established in terms of formal commitment or recruitment. Fourth, many organizations perform only some degree of reactive management (preventing past incidents from re-occurring), and very few organizations perform proactive management (preventing incidents from occurring). Fifth, conventional tools and technology do not support integrated processes. Sixth, there is a poor link between the error details and change management processes.

To improve the operating support to software engineering methodologies, they can be represented declaratively in knowledge-based systems, and employed by generic inference engines providing answers for a number of cases[6]. Declarative languages economize knowledge development and maintenance, and make it easily accessible. However, human can better exploit knowledge when explicitly decomposed into an operational description, as a workflow[7], especially if such knowledge is distributed among different people. For this reason, workflow management technology is increasingly adopted to support decentralized decision-making processes, since it facilitates communication, coordination, and collaboration between participants[8]. The reasons for making workflow (or process) models are manifold: they are used for communications, ISO 9001 certification, system configuration, analysis, simulation, etc. A process model may be descriptive, i.e., it tries to capture existing processes without being normative.

In the literature of the broader context of Business (Process) Intelligence (BPI) and Business Activity Monitoring (BAM), there are many process discovery and conformance analysis techniques. However, the focus of such techniques is on clustering and performance analysis rather than in causal relations. Indeed, the techniques adopted in our approach employ a mining paradigm that operate at the workflow (net) level rather than at sequential or lower representations (e.g., Markov chains, finite state machines, regular expressions, etc.).

In last years, a number of specifications have been proposed for describing processes or workflows. Business Process Model and Notation (BPMN)[9] is the most widely accepted one, due to the following reasons. First, it provides intuitive and representative semantics. Second, it includes a number of constructs and design patterns to model decentralized business-collaborations[10]. Third, the service-oriented computing, which is at the core of the BPMN 2 conception, provides flexible, dynamic, component-oriented interoperability, for the dynamic composition of business processes[11]. Fourth, it offers different types of activity and event for the specification of business processes with well-defined semantics, readable by software agents[11]. Fifth, it can be extended to explicitly incorporate problem solving knowledge in process modeling. Therefore, we rely on BPMN as a reference language for workflow modeling.

A BPMN process model represents the routing of work in the software process management. Information about the software process is gathered as it takes place. Such model can be generated either by traditional process modeling, or by process discovery of pilot software processes. We assume here that it is possible to collect workflow events such as tasks, cases, and timestamps, in audit trails. Indeed, the most transactional information system offers nowadays this information in some form. Such information is then used for conformance analysis, which aims to detect inconsistencies between the process model and its corresponding execution log. The analysis is not limited to post-runtime inspections. Instead, it can be used for many purposes. First, for operational support by detecting traces being executed that do not follow the intended process. Second, for provisioning of recommendations to the user when selecting the next activities in the process. Third, for deriving information for the design of software processes before they are implemented.

In this paper, we introduce the formal foundations and define the proposed approach, by applying it to a real-world case study in the areas of software engineering in order to draw some conclusions and future work. Finally, we show some potential application to other processes, such as those in the manufacturing industry.

2. Problem statement and pilot case study

The SENIOR (Software ENgineering for Input/Output pRoblems) case study is an excerpt of a real-world dataset, truncated to serve as a pilot case for illustrating the approach and the techniques adopted. It contains 16 cases, for 241 total events. Each case contains the audit trails of a different software developer, working on the Java Standard Edition programming language version 8, and the NetBeans integrated development environment version 8. More specifically, the application developed are functionally different, but structurally made by five input/output (I/O) software modules: file based I/O, network based I/O, eXtensible Markup Language (XML) based I/O, GUI based I/O and Database Management System (DBMS) based I/O. The same normative development software process has been assigned to all cases, to be carried out individually by each worker.

Fig. 1 shows a BPMN diagram of such normative process. To clearly understand its logic, the basic elements of the language are first summarized. The interested reader may refer to[9] for a detailed study of the language. In BPMN, *events* (represented as circles) model something that can happen during the process. A workflow is activated by a *start event* (the circle with a single thin border, on the top-left) and terminated by an *end event* (the circle with a single thick border, on the top-right), while *intermediate events* (circles with double thin border, not used in the figure) can occur anywhere within the flow. *Sequence flows*, represented by a solid arrow, model the order of execution of activities in the workflow. *Tasks* (the rounded-corner rectangles) are atomic activities of the workflow, whereas *gateways* (the diamonds) are decision points to control the flow of work. In particular, the *exclusive gateway* (the diamond carrying the 'x' marker) routes the incoming flow to one of the mutually exclusive outgoing flows, based on a logic condition. Differently, the *parallel gateway* (diamond carrying the '+' marker) waits for all incoming flows before triggering the flow through all outgoing flows.

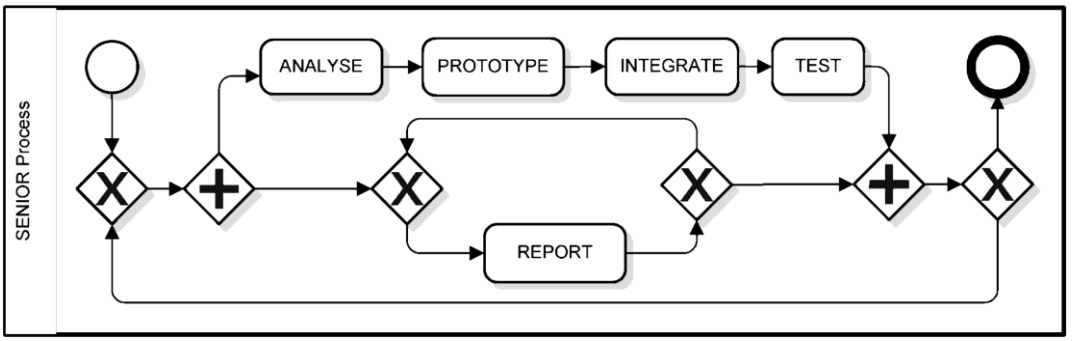

Fig. 1. Pilot case study: a BPMN diagram of the SENIOR normative process.

Given the above elements, the SENIOR process can be specified in natural language as follows: *a given problem should be first analysed; a solution must be then prototyped and subsequently integrated into the system; finally, the solution must be tested. In parallel, the activities must be reported, possibly in multiple steps. After testing and reporting, the flow can restart from analysis in order to improve the solution iteratively.*

Let us denote, for short, by *a*, *p*, *i*, *t*, and *r*, the activities *analyse*, *prototype*, *integrate*, *test*, and *report*, respectively. It is apparent that the process model of Fig. 1 allows some execution sequences of the activities such as $\langle a, p, i, t, r\rangle$, and $\langle a, p, r, i, t, r, a, p, i, t, r\rangle$, but not $\langle a, p, r, a\rangle$ or $\langle a, p, i, p, i\rangle$. Such examples raise the interesting question "do the model and the execution log conform to each other?". When process model and process execution are fully conform, it is also interesting to analyse how frequent certain parts in the model are actually used, to potentially remove obsolete parts (model maintenance). Assuming that the cases in log are representative and a sufficient subset of possible instance of processes has been observed, we can even extract from the log the entire process model actually performed, without using any other a-priori information.

The starting point for process mining techniques is then an audit trail (or event log). Here, we assume that it is possible to sequentially record events referring to the execution of activities. More precisely, each event log refers to an *activity* (i.e., a well-defined step in the process), and a *case* (i.e., a process instance). Additional information may

be stored, such as the *performer* (i.e., the person executing or initiating the activity), the *timestamp*, the activity *duration*, the *resources* (e.g., a device), and so on. Table 1 shows an excerpt of the event log of the pilot case study. Here, Case 477447 generated the sequence ⟨*a*, *p*, *i*, *t*, *r*⟩, which fully fits the normative process. In contrast, Case 479570 generated the sequence ⟨*a*, *p*, *t*, *p*, *t*, *i*, ...⟩, which does not fit. Indeed, to carry out *test* before *integrate* is forbidden by the normative model, since it implies only a unit test and not a system test. Furthermore, to carry out *prototype* after *test* without *analyse* implies that there is an implementation update without a corresponding design update: it is well known that this is an error-prone practice, leading to inefficient management of the product lifecycle.

Table 1. Pilot case study: an excerpt of the event log.

Case	Timestamp	Activity
477447	16/12/2014 16.00	start
477447	16/12/2014 16.01	analyse
477447	27/12/2014 14.41	prototype
477447	24/01/2015 11.23	integrate
477447	26/01/2015 09.31	test
477447	26/01/2015 10.31	report
477447	26/01/2015 16.01	end
479570	12/12/2014 16.55	start
479570	12/12/2014 16.56	analyse
479570	27/01/2015 10.18	prototype
479570	31/01/2015 17.59	test
479570	02/02/2015 11.00	prototype
479570	03/02/2015 16.54	test
479570	04/02/2015 11.02	integrate
…	…	…

Table 2 shows a collapsed view of the audit trails. Here, each violation consisting in a forbidden sequence of activities has been represented in boldface style. Other violations, such as missing activities, have been also computed. The number of violations per case is shown as a separate column. It is apparent that only three cases fit the normative model. Thus, the normative process might not describe accurately what actually happens.

Table 2. Pilot case study: a summary of the overall log.

Case	Audit trail	No. of violations
477447	a, p, i, t, r	0
491381	a, p, i, t, r	0
477089	a, p, i, t	1
485552	a, p, r, **p**, r, **p**, i, t	2
490875	a, p, i, t, r, **i,** r, **t**	2
456762	a, p, i**, p**, **i**, **p**, **i**, t, r	4
493724	a, r, p, **a, i, p,** a, r, **i**, r, p, i, t, **i**, **t**	5
490259	a, p, i, **p**, **i**, t, p, i, t, p, **t**, i, t, **i**, **t**, p, i, t, r	5
479570	a, p, **t, p, t,** i, t, r**, i**, r, **t, a**	6
475754	a, p**, a**, **p,** r**, p**, i**, p**, **i, p**, **i**, t	7
492746	a, p, i, **p**, **i**, t, **p**, i, t, **i**, r, i, **p**, i, r, t, **i**, **t**	7
476904	a, p, r, **p**, i, t, **i, t, i, t, i, t**, r, **t**	8
490756	a, p, i, **p**, **i**, **p**, **i**, **p**, **i**, t, r	8
493534	a, p, i, a, p, **a**, **p**, i, a, p, **a**, **p**, i, t, r, **t**, a, p, i, t, a, p, i, t, r, **t**, r, **i**, r, **t**	8
492998	a, p, **a, p, a, p**, i, **p, i, p, i, p, i**, t, r, **t**	11
483505	a, p, **a**, **p, a, p**, i, **p, i, p, i, p, i, p, i, p, i, p, i**, r	16

3. BPMN process discovery and conformance analysis

3.1. BPMN process discovery

Discovery techniques can be used to mine a process model on the basis of the behavior observed in the event data. For a given audit trail several models can be derived. The difference between models can be assessed in terms of our essential quality criteria:

(i) fitness is the degree of log behavior that a process model is able to replay. For example, the fraction of event patterns represented by the model, the fraction of cases that can be replayed in the model;

(ii) simplicity means that the resulting process model should be readily understandable. For example, a complexity metrics for process models such as model size or degree of structuredness;

(iii) precision refers to the degree of behavior that is allowed by the model, but not observed in the logs. In general, it is easy to create a process model allowing the execution of all tasks in any arbitrary order, but we can hardly learn any specifics from such a model;

(iv) generalization refers to the ability of a process model to abstract from the behavior documented in the logs. A discovery technique able to generalize helps to work with incomplete behavior.

In this study, we show the application of a recent technique: the BPMN Miner algorithm[12], which is based on the Inductive Miner algorithm. In essence, it aims to discover block-structured process models fitting the behavior represented in event log. Inductive Miner partitions the activities, selects the most important process constructs, splits the log and recurses until a base case is encountered. It is based on a process tree, a hierarchical representation of a block-structured workflow net. In a process tree, the leaves of the tree are activities, representing transitions. The nodes of the tree, operators, describe how their children are combined: exclusive choice, sequential composition, parallel composition, and loop. As a basic example of application of the BPMN Miner, Table 3 shows a synthetic case study. Here, the base pattern <*a, p, i, t*> is iterated one and many times, placing <*r*> on different positions.

As a result, Fig. 2 shows the model generated by the BPMN Miner: since there is no violation in the audit trail, the generated model is very similar to the normative process. In contrast, Fig. 3 shows the "spaghetti" model generated via the pilot case study. It is very different from the normative process, due to the high number of violations.

Table 3. Synthetic case study: a summary of the overall log.

Case	Process log
0	r, a, p, i, t
1	a, r, p, i, t
2	a, p, r, i, t
3	a, p, i, r, t
4	a, p, i, t, r
5	a, p, i, t, r, a, r, p, i, t
6	a, r, p, i, t, a, p, r, i, t

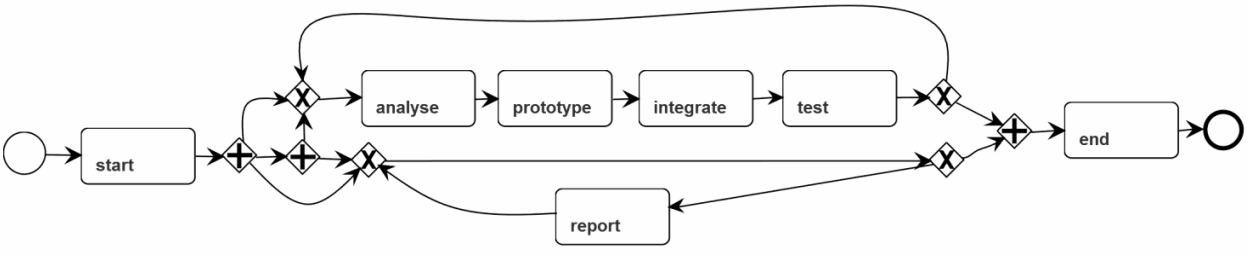

Fig. 2. Synthetic case study: the generated BPMN diagram.

3.2. BPMN conformance analysis

Conformance analysis techniques aim to detect and quantify inconsistencies between the process model and its corresponding execution log, by means of algorithms and metrics[13,14]. The process owner can use conformance analysis for process both controlling and monitoring. Process controlling deals with the analysis of historic process execution, e.g., a quarter or a fully year. It is an offline activity which provides insights into whether the general objectives of a process have been met and whether the Key Performance Indicators (KPIs) are in line. Process monitoring measures the quality of currently running process instances. It is a continuous online activity working with objectives and rules formulated for individual cases, and triggering counteractions when these rules are violated.

A basic idea of conformance checking is to replay each trace of the log recording at each step whether an activity is allowed to be executed according to the model. More specifically, at each step, the number of tokens that are required for replaying an activity is compared with the actually available tokens, counting relevant facts such as the tokens correctly produced, correctly consumed, missing and remaining. A fitness measure of the case is then calculated by using the fractions missing-to-consumed and remaining-to-produced. An important fitness measure is the *trace-fitness*, representing how well the event log can be replayed in the Petri Net generated by the normative BPMN model. A fitness value of 1 means that the log can be successfully replayed, whereas a value of 0 means that this is completely not the case. As an example, Fig. 4 shows the trace fitness measures for two cases, i.e. 491381 and 477089. Here, the alignment of the events carried out by each case is shown. In general, a number of violations can be detected, such as synchronous move, unobservable move, skipped event, inserted event, replaced violation, and swapped violation.

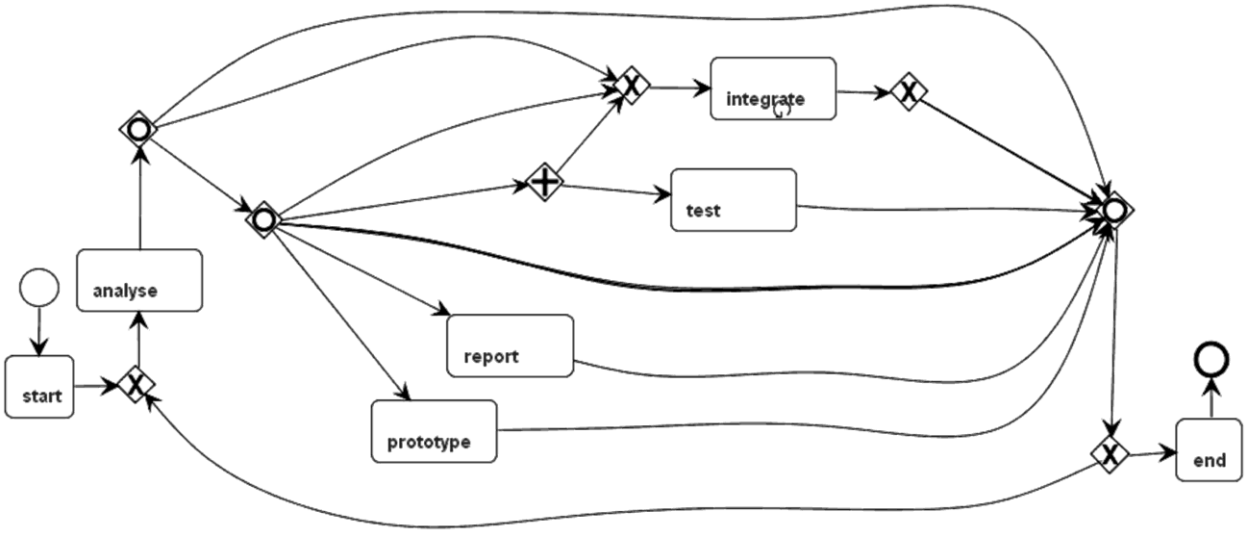

Fig. 3. Pilot case study: the generated BPMN diagram.

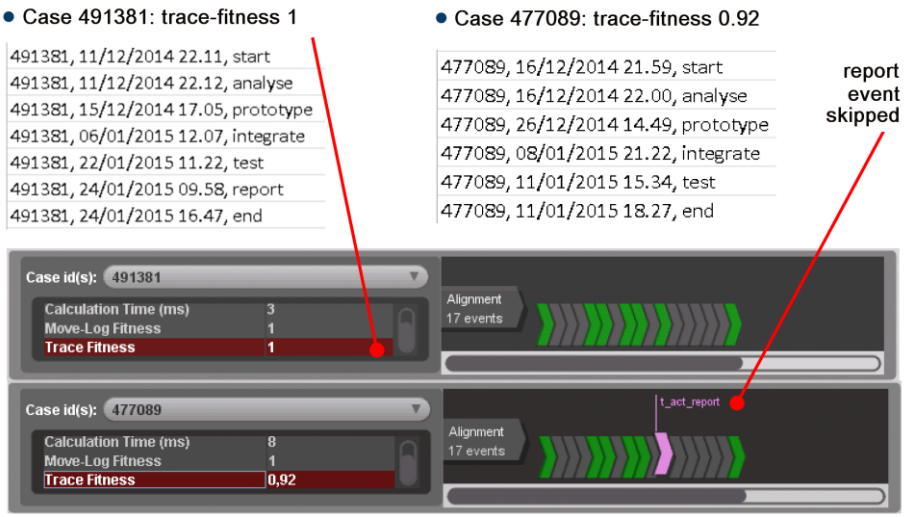

Fig. 4. Pilot case study: trace fitness of two cases.

The overall trace fitness of all cases is shown in Fig. 5. A comparative analysis with Table 2 reveals that the cases ordering based on number of violations and the ordering based on trace fitness are very similar. The differences can be ascribed to the higher types of violations considered by the trace fitness.

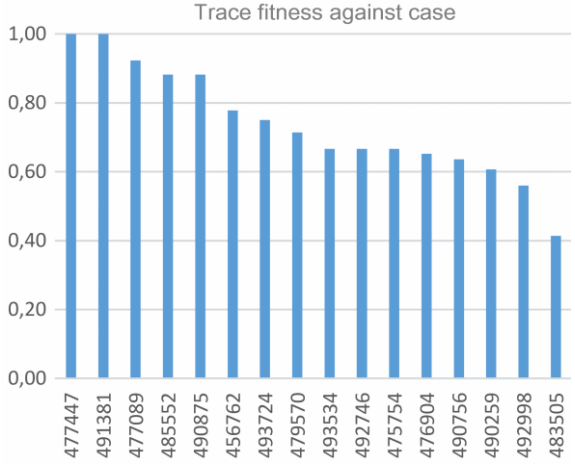

Fig. 5. Pilot case study: trace fitness of all cases.

4. BPMN modeling and example in other industrial sector

The methodology presented in this paper is based on the availability of process models expressed in the BPMN standard and of corresponding audit trails. As a such, it can be extended to other industrial sectors. As an example, Fig. 6 shows the BPMN model of a manufacturing company.

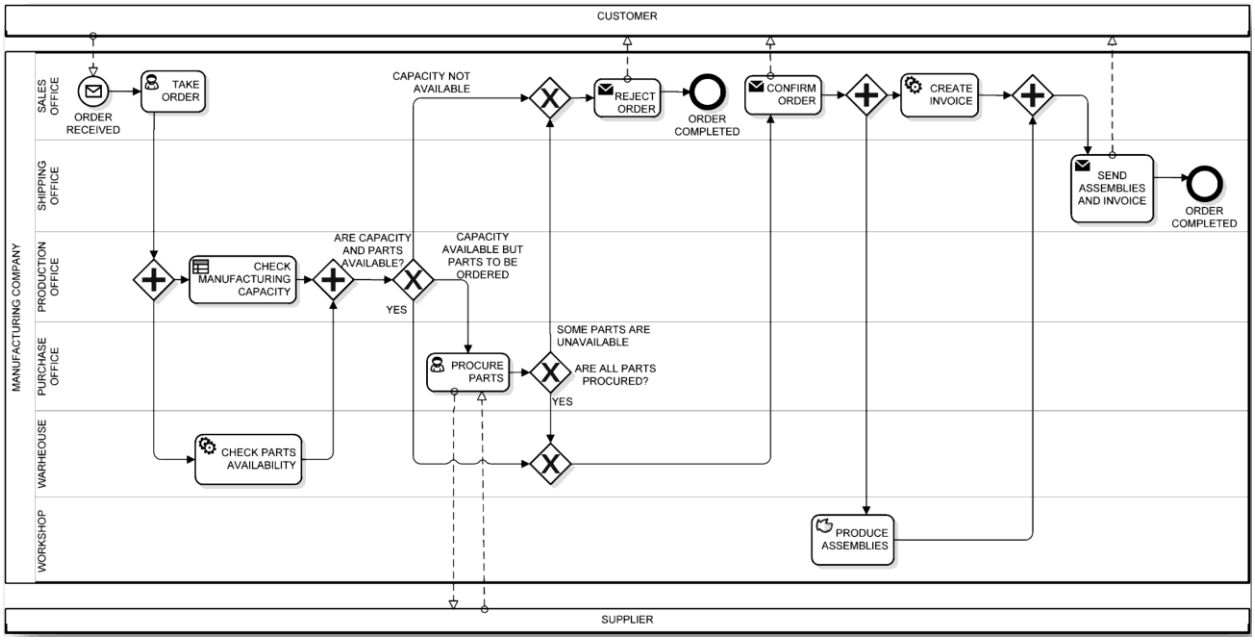

Fig. 6. The BPMN model of a typical process of a manufacturing company process.

Table 4. Acronyms of the sub-processes.

Acronym	Subprocess
TO	Take Order
CMC	Check Manufacturing Capacity
CPA	Check Parts Availability
PP	Procure Parts
RO	Reject Order
CO	Confirm Order
PA	Produce Assemblies
CI	Create Invoice
SAI	Send Assemblies and Invoice

Table 5. Examples of trails fitting the process model reported in Fig. 6.

Case	Trail
fit1	TO, CMC, CPA, PP, CO, PA, CI, SAI
fit2	TO, CPA,CMC, CO, CI, PA, SAI
unfit1	TO, CPA,CMC, CO, CI, PP, PA, SAI
unfit2	TO, CPA,CMC, CO, PP, PA, PP, PA, PP, PA, CI,SAI
unfit3	TO, CPA,CMC, CO, PP, PA, CI, PP, PA, CI, PP, PA, CI,SAI
unfit4	TO, CPA,CMC, CO, PP, PA, CI, SAI, PA, CI, SAI, PP, PA, CI, SAI

More specifically, the process model of Fig. 6 can be specified in natural language as follows: order processing in a manufacturing company takes place after an order has been received from a customer. The order is first taken by the sales office. Both manufacturing capacity and parts availability are then checked, via appropriated business rules and an automated information service, by the production office and the warehouse, respectively. Subsequently, an order rejection is sent to the customer by the sales office if capacity is not available. Otherwise, if parts are also available, an order confirmation is automatically sent to the customer by the sales office. Therefore, both assemblies and invoice are produced, by human operatives at the workshop and by the sales office via an automated information

service, respectively. Finally, assemblies and invoice are sent to the customer by the shipping office. In contrast, if there are parts to be ordered, the purchase office procures such parts, purchasing them from suppliers, so as to allow the aforementioned process to continue once such parts are available. However, if some parts are still unavailable (they cannot be procured), a rejection is then sent to the customer by the sales office.

The described process seems very simple and common, but the number of possible nonconformities presented by an actual process compared with the predefined model is virtually unlimited. Furthermore, such nonconformities can be very difficult to detect if only predefined KPIs are used to control the process.

Assuming the correspondence reported in Table 4, two examples of trails fitting the predefined process shown in Fig. 6 are reported in Table 5 (cases named fit1 and fit2). But, if the warehouse management is unreliable, the necessity of procurement could not be correctly detected. In such case the PP sub-process could occur immediately before the PA sub-process (case unfit1) due to the late detection of material lack. This situation, in the worst case, could occur more and more times (case unfit2, more dissimilar from the predefined model than case unfit1). In this last case, probably, also the invoice should be updated more and more times (case unfit3). The partial (and undetected) availability of parts, and/or the temporary (and unpredicted) unavailability of machines could also generate the splitting of the final part of the process in a series of chaotic repetitions of the last steps of the production process (case unfit4).

These are only few instances of nonconformities presented by a real process. Using only classical KPIs (e.g. lead time), due to the extreme diversity of the actual situations, neither the presence nor the ranking of the presented problems could be detected. In fact an oversight of the warehouse's personnel can generate a big delay, but it probably represents a "local" problem. A lot of violations of the predefined model diffused along the process can generate lesser delay, but they could be a signal of bigger problems in the process and concerning different parts of the organization. By using the BPMN process mining the different cases represented in Table 5 can be clearly distinguished.

BPMN process mining can be also very useful when the predefined process model is not more fitting the reality, and the personnel autonomously changes their behaviour to maintain the effectiveness and/or the efficiency of the process. In these cases the detected nonconformity can promptly stimulate the organization to update the model of the process, accepting or improving the modification made by the personnel. An example referred to the process shown in Fig. 6 could be the introduction of just in time method, that eliminates (or drastically reduces) the necessity of CPA sub-process. The vanishing of CPA sub-process from the trails could stimulate the organization to analyse the new situation, updating the model and thinking about its new configuration.

5. Conclusions

In this paper a novel approach of process-oriented software engineering is presented. The approach is based on the representation of business processes in the BPMN standard, and on the use of process discovery and conformance analysis techniques. The application on a real-world scenario has made possible the initial roll-out of the approach in real environments, showing the effectiveness of the method for both process monitoring and controlling. The power of the method is the possibility to overcome the standard way used in process monitoring practice, based on the collection of a series of values of predefined KPIs. Using the standard approach the control of the process can be performed merely by comparing the KPIs collected values with the threshold values concerning each single KPI. Conversely, by using a BPMN process mining approach a deeper process control could be performed, and also complex non-compliant behaviours could be highlighted. Also, process reengineering activity could be made easier, because it would be simple to understand in which way and why the personnel do not operate closely to the predetermined process model.

As an example, a possible application in the manufacturing domain is drawn.

Since the approach is based on process models, it can be extended to traditional industrial sectors, in particular to support processes such as supplying, warehousing, management of customer relations. The main difficulty to overcome is the generation of proper data flows, but the continuous development in sensing technology together with the evolution of computerized management systems allows to forecast a wider application of such method.

References

1. Rubin V, Günther CW, Van Der Aalst WM, Kindler E, Van Dongen BF, Schäfer W. Process mining framework for software processes. In: *International Conference on Software Process*. Springer Berlin Heidelberg; 2007. p. 169-181.
2. Lemos AM, Sabino CC, Lima RM, Oliveira CA. Using process mining in software development process management: A case study. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE. 2011. p.1181-1186.
3. Rubin V, Lomazova I, van der Aalst WM. Agile development with software process mining. In *Proceedings of the 2014 international conference on software and system* process. ACM. 2014. p. 70-74.
4. Saylam R, Sahingoz OK. A Process Mining Approach in Software Development and Testing Process: A Case Study. In *Proceedings of the World Congress on Engineering*. 2014;**1**.
5. Ahern T, Leavy B, Byrne PJ. Complex project management as complex problem solving: a distributed knowledge management perspective. *International Journal of Project Management*, 2014;**32**(8):1371-1381.
6. Fensel D, Motta E, van Harmelen F, Benjamins VR, Crubezy M, Decker S, Gaspari M, Groenboom R, Grosso W, Musen M, Plaza E, Schreiber G, Studer R, Wielinga B. The Unified Problem-solving Method Development Language UPML. *Knowledge and Information Systems* 2003;**5**(1):83-131.
7. Vidal JC, Lama, M, and Bugarín, A. A Framework for Unifying Problem-Solving Knowledge and Workflow Modeling. *Proceedings of the Association for the Advancement of Artificial Intelligence Spring Symposium: AI Meets Business Rules and Process Management*; 2005, Palo Alto, CA, USA, pp 105-110.
8. Kaster DS, Medeiros CB, Rocha HV. Supporting modeling and problem solving from precedent experiences: the role of workflows and case-based reasoning. *Environmental Modelling & Software*, 2005;**20** (6): 689-704.
9. OMG (Object Management Group). *Business Process Model and Notation (BPMN), Official specification*, 2011, Version 2.0. http://www.omg.org/spec/BPMN/2.0.
10. Bechini A, Cimino MGCA, Marcelloni F. and Tomasi, A. Patterns and technologies for enabling supply chain traceability through collaborative e-business. *Information and Software Technology*, 2008;**50**(4):342-359.
11. Cimino MGCA and Marcelloni F. Autonomic tracing of production processes with mobile and agent-based computing, *Information Sciences*, 2011;**181**(5):935-953.
12. Leemans SJ, Fahland D, van der Aalst, WM. Discovering block-structured process models from event logs-a constructive approach, In *Application and Theory of Petri Nets and Concurrency*, Springer, Berlin Heidelberg, 2013, pp 311-329
13. Rozinat A, van der Aalst WM. Conformance checking of processes based on monitoring real behaviour. *Information Systems* 2008; **33**(1) p. 64-95.
14. van der Aalst W, Adriansyah A, van Dongen B., Replaying history on process models for conformance checking and performance analysis. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2012;**2**(2):182-192.