The Discovery of the Implemented Software Engineering Process Using Process Mining Techniques

Zayed, Mostafa Adel
Information Systems Department,
Faculty of Computers and Information,
Helwan University, Egypt

Ahmed Bahaa Farid
Information Systems Department,
Faculty of Computers and Information,
Helwan University, Egypt

***Abstract*—*Process* model guidance is an important feature by which the software process is orchestrated. Without complying with this guidance, the production lifecycle deviates from producing a reliable software with high-quality standards. Usually, teams break the process deliberately or impulsively. Application Lifecycle Management (ALM) tools log what teams do even if they break the process. The log file could be a key to discover the behavior of the undertaken process against the targeted process model. Since the date of its introduction, Process Mining techniques have been used in business process domains with no focus on the software engineering processes. This research brings the Process Mining techniques to the software engineering domain. The research shows a conclusive effort that used a Scrum adapted process model as an example of Agile adoption. This research has applied Process Mining discovery techniques to capture the actually implemented process by the Scrum team. This application clarifies the gap between the standard process guidance and the actually implemented one. The research's results showed that Process Mining techniques have the ability to discover and verify the deviation on both levels; the process itself as well as the work items state-machine workflows.**

Keywords*—*Process Mining; Process Models Discovery; Software Engineering; Agile; and Scrum

I. Introduction

Software engineering process has become an integral part of any software production lifecycle definition. Without a model that governs the state-machine of the process phases, software engineering process will not be clearly defined. Software factories choose one of the available process models according to a set of aspects e.g. product team preparation, customer type, project development period, etc. Correspondingly, contemporary ALM tools have been developed to provide sort of automation, collaboration, and process model guidance.

In late 1970s, computer scientists focused on efficient data retrieval and storing. Nevertheless, this was inadequate, because any business is not only composed of data but also a process. A good start was in the nineties, where processes' representation, monitoring, and visualization drew computer scientists' attention as an essential block for extending the capabilities of information systems to enforce business processes. This type of information systems is denoted by Business Process Management (BPM) systems [1]. Business process management systems examples are Staffware, MQSeries, Microsoft Work Flow, COSA, and FLOWer [2, 3, 4, 5]. Despite BPM systems do straighten many deviations, deviations do occur. Therefore, a methodological solution is needed to identify these deviations after they happened and logged, then analyze these deviations. Since information system records all that happened, an event log can be prepared –if it was not originally prepared, and use it to identify the actual process that the event log depicts [6]. This identification phase is denoted by Discovery. In addition, discovery can lead to process analysis. To check what rules are applied and what is neglected or overwritten that is denoted by Conformance Checking [7]. In order to check process conformance and analyze it, a discovery phase has to be applied to the recorded event log. Using both discovery and conformance checking the current process can be enhanced, considering the deviations are exceptions or unknown cases that are not handled by the process model. The result of enhancement is repairing or extending the current applied process model [8]. Discovery, Conformance Checking, and Enhancement are the Process Mining types. Fig. 1 depicts the need and positions the Process Mining from organizational process models' interaction with current systems, and rectification of the current process models.

II. Literature on Process Mining

A. Review of Process Mining algorithms and techniques

In [9, 10] α-algorithm and α^{+}-algorithm discuss the most common discovery algorithm with test datasets for testing the capabilities of the algorithm at the edge. Then, in [11] a HeuristicsMiner algorithm is introduced to discover the process and identify the noise in the process. The used dataset was hospital information system of patients. Then, in [12, 13] a Fuzzy Mining algorithm was built upon the HeuristicsMiner algorithm introduced in [11]. This algorithm is built to discover the unstructured logged processes. The datasets used were machinery test and usage logs, development process logs, hospital patient treatment logs, and logs of case handling systems and web servers. Then, a more mature framework was introduced in [14], which introduced a sound and complete solution for discovering structured processes that has compliance with the four quality dimensions introduced in [15].

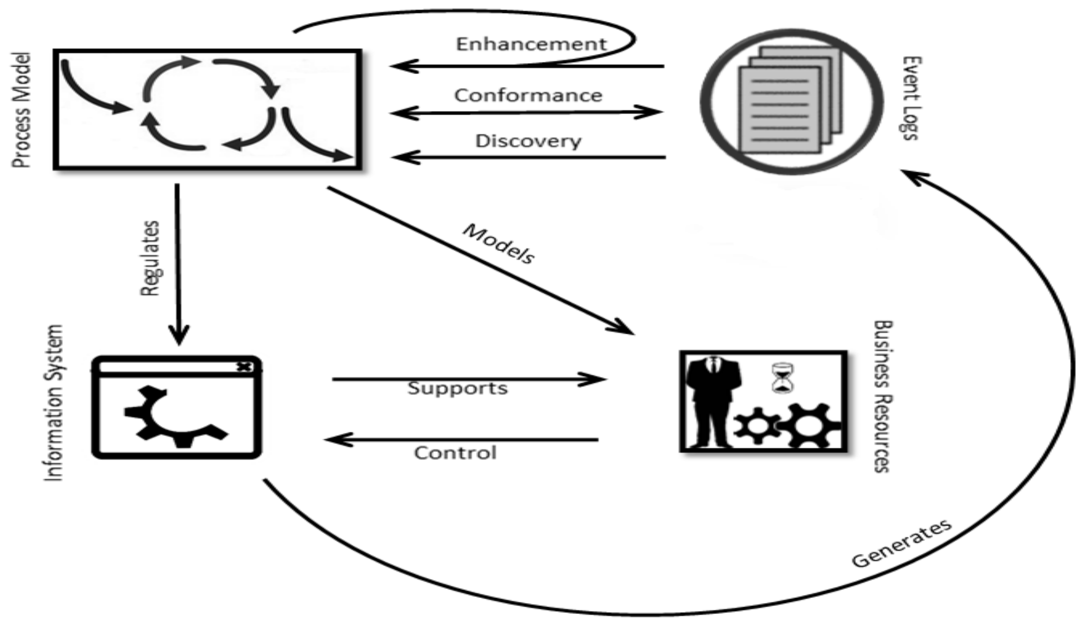

Fig. 1. Process Mining occurrence in organizational process models

In [14] the framework was applied on CoSeLoG project that used municipalities' data in Netherlands. These quality dimensions were studied to show how crucial are they and how important to implement them in [16].The evaluation criteria depends on the four quality dimensions. These quality dimensions are 1: Fitness, 2: Precision, 3: Generalization, and 4: Simplicity. Fitness is represented by this question, is the observed behavior captured by the model? I.e. How many cases can be replayed from the event log over the extracted model? Precision is represented by this question: How precise the process model describes the observed behavior? The less behavior the process model allows that is not observed in the event log, the more precise the process model describes the behavior. Generalization is represented by this question: Does the model allow for more behavior than encountered in reality? Simplicity is represented by this question: How simple, or human-readable, is a process model? [17] A comparison shows the differences between discovery algorithms is shown in TABLE 1 [14]. The tilde (~) sign in the table represents algorithm finish with remaining work. A sound (correct) process models can be produced with ETM*d* algorithm as shown in the last row in TABLE I. . ETM*d* algorithm is a part of ETM (Evolutionary Tree Miner) framework, which is built using an evolutionary algorithm, genetic programming specifically. The best part of evolutionary algorithms is that it finds a solution as long as you have the time to wait for its solution, unless you defined an early exit criterion. In general, Process Mining has a set of characteristics, guiding principles and challenges [18]. ETM framework has covered seven challenges out of eleven challenges mentioned in the manifesto. A full list of challenges that ETM framework has overcome can be read from here [14]. An important challenge is “Improving the Representational Bias Used for Process Discovery”, as mentioned by the process mining manifesto, or “Separation of Visualization and Representational Bias”, as mentioned by [14]. ETM framework uses Process Trees to overcome this challenge. In order to have a deeper sense of this challenge, it should be considered that process models have standard notations like Petri nets, business process management notation (BPMN) [19], and event-driven process chain (EPC) [20]. The discovery algorithms do not mostly use these standards –as mostly each uses its own notation, which makes it hard to business users to read and use such algorithms, which make their own notation. Examples of such algorithms are fuzzy models [13, 12], casual nets [21], and heuristics nets [11, 22]. The crucial point of ETM framework considering the representation challenge is the usage of process tree as it is mentioned before. Process trees notation by itself is not considered a common notation to business people, but process tree notation can be easily converted to a plethora of other common processes modeling notations like Petri nets, EPC, YAWL, casual net, heuristic net, fuzzy model and process algebra. This paper is going to use ETMd as the research's discovery algorithm; in addition, this paper will show the discovered process model using the α-algorithm. This research is going to use software engineering data, based on Scrum process specifically. In fact, up to the moment, software engineering domain is considered a virgin domain for applying Process Mining techniques. The research's results will be visualized using three different forms: process tress,

Petri nets, and BPMN as the OMG standard notation.

The rest of this section discusses the research motivation in subsection C. The process tree notation is showed in subsection D. In addition, the dataset collection and preparation that the research was based on is discussed in section II. Moreover, in section II, this literature has a subsection discussing the Scrum process definition template. Discovery of the both process levels is discussed in section III. The results are compared to the process definition template in section IV. Finally yet importantly, a quick summary is showed in section V.

TABLE I. PROCESS DISCOVERY ALGORITHMS COMPARISON FROM [14]

Algorithm	**Error-Free?**	**Replay Fitness**	**Precision**	**Generalization**	**Simplicity**
α-algorithm	Yes	~	Yes	No	No
Genetic miner	~	~	~	No	No
Heuristics miner	~	No	Yes	No	No
ILP miner	~	Yes	No	Yes	No
Inductive miner	Yes	Yes	No	Yes	Yes
Language-based region theory	Yes	No	No	No	No
Multi-phase miner	~	Yes	No	No	No
State-based region theory	Yes	Yes	No	No	No
ETM*d*	Yes	Yes	Yes	Yes	Yes

B. Process Tree Notation

Process trees have six different operators: sequence (→), the reversed sequence (←), exclusive-choice (×), parallelism (∧), non-exclusive choice (∨) and the loop (⥁) [11]. Sequence operator (→) forces the mentioned sequence of child nodes from left to right and vice versa considering the reversed sequence operator (←). Exclusive-choice operator (×) of child nodes a, b, and c results in one of the following combinational sequence <a>, <b>, or <c>.

Parallelism operator (∧) of child nodes a, b, and c results in the set of {a, b, c} in all possible orders, i.e. the result can be one of the following combinational sequences <a, b, c>, <a, c, b>, <b, a, c>, <b, c, a>, <c, a, b>, or <c, b, a>. Nonexclusive choice operator (∨) of child nodes a, and b results in one of the following combinational sequences <a>, <b>, <a, b>, or <b, a>. On one hand, any of the previous operators could have any number of nodes starting from two nodes – of course; one child node has no meaning since it should be cloned to its parent. On the other hand the loop operator (⥁) has, at least, three child nodes. Loop operator (⥁) of –specifically ordered, child nodes a, b, and c results in one the following combinational sequences <a, c>, <a, b, c>, <a, b, a, c>, <a, b, b, b..., c>.

III. RESEARCH MOTIVATION

No previous research effort tried to use the aforementioned Process Mining techniques and algorithms in order to be used in the software engineering domain. According to Forrester's State of Agile 2015 report, 40% of the developed software is doing wrong things due to deviations from applying the lifecycle process [23]. In 59% of cases, the major impediment that prevents a correct Agile process adoption is the lack of skilled people [23]. If there were an intelligent way to discover these deviations in the applied lifecycle process, this would give the software factories the opportunity to bring their development back to the track. Thus, decreasing the possibility of producing the incorrect thing. This research utilizes Process Mining techniques to discover the deviation of the actually implemented software engineering process. Since, 86% of the Agile, teams are using Scrum [23]; this research was applied on a project log file of a team that is supposed to be working using the Microsoft Scrum process template definition.

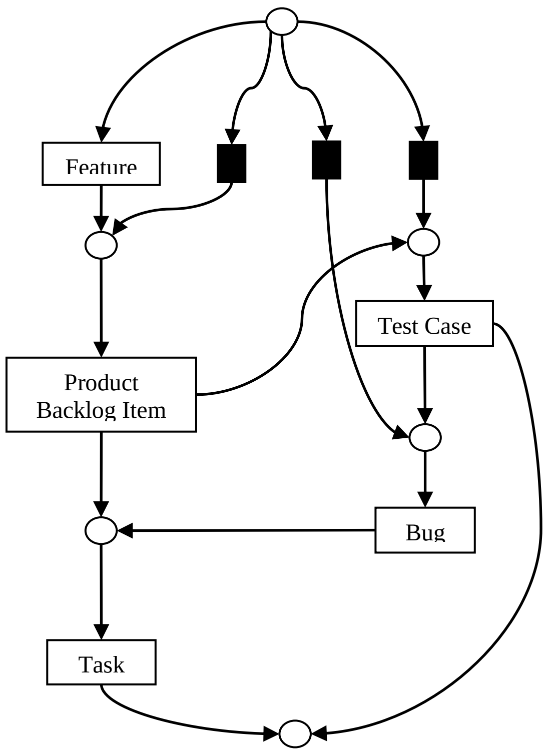

Fig. 2. Scrum Process Work Items Workflow in Petri Net

IV. RESEARCHED DATASET ACQUISITION

Scrum process categorizes all work in the project life cycle into six categories applied by the Microsoft Visual Studio Scrum 3.0. These six categories are feature, product backlog item (PBI), task, test case, bug, and impediment. Each one of them is called work item (WI) and has its own workflow states' model. Altogether, they formulates the software engineering Scrum process. According to the held data, this study is going to concentrate on subset of these WIs, which are feature, product backlog item, bug, and task. The Scrum process as defined by Microsoft is shown in Figure 2. The workflow model of each one of them is depicted in Fig. 3 [24]. The CASE tool used to capture the team's data and apply the desired process is Microsoft Team Foundation Server (TFS) 2013.

TFS data resides in Microsoft SQL Server. In order to discover the generated the event log, this study used ProM tool for generating the event logs, and discovering process off the data [25]. This research used SQL queries to extract the required process attributes and data, and then it used XESame to generate event logs in order to able to discover the process using discovery algorithms implemented in ProM. XESame is an integrated software in ProM responsible for extracting event it uses JDBC. XESame generates the event logs in defined formats with extensions (.XES), and the event log should be composed of events, cases, traces, and attributes.

For example, if you have two product backlog items, so, each one of them is called a case. Each case is composed of a set of chronologically ordered events using timestamp, called a trace. Each PBI has a team member changes its state. Both timestamp and the team member are called attributes. From TABLE II. and TABLE III. you can see how the event logs are composed. For the extraction phase, in order to discover the workflow of each work item this research defined the work item type and work item ID as the case, and this research defined work item state as the event transition life cycle. Data is filtered for the WIs workflow that is a single process too, which composes of 1482 cases, 12 event types, and event's frequency of 8609. In addition, to discover the life cycle of the scrum process, iteration ID is selected as the case, and work item types is selected as the event transition life cycle. Data is filtered for the lifecycle of the scrum process that is a single process, which composes of 500 cases, 5 event types, and event's frequency of 2496. After this step, it is capable of extracting the two XES files to discover each of their process models.

A. Scrum Process WI's Petri Net Discussion

In Figure 2 depicts the work item's (WI) process. The black transitions called siltent transitions. This literature used this transitions to be able to depict PBI WI, Bug WI, and Test Case WI can come as the first transition. In addition, PBI WI can come after Feature WI. Test Case WI can be the child of the PBI WI as well as the Task WI. Task WI's parent can be also a Bug WI. Bug WI's parent can be Test Case WI. Any case happens other than the mentioned cases is a clear devision from the definition template. An example of the deviation is a Task WI is a child of Feature WI.

V. Discovery of the Actually Applied Process Model

The most common and well-known discovery algorithm is α-algorithm [9]. Using 'Mine for a Petri Net using Alpha-algorithm' plug-in from ProM, which implements the α-algorithm and applying it over the extracted XES file that contains the data of different work items workflows, and visualize the discovered in petri net. Fig. 4 depicts the discovered workflow of different work items. Fig. 4 consolidates all work items workflows in single petri net.

Another way to discover this event log is using 'Mine a Pareto Front with ETMd' plug-in from ProM. Pareto Front is multi-objective optimization technique that offers a set of the best-optimized solutions. The trade-off between objectives in this study is the four quality dimensions. The plug-in offers a set of process models of Pareto fitness equals to 0.991326, which means that most of the offered solutions are feasible. Pareto fitness is the average fitness of the whole Pareto Front.

For the research's experiment, it has only two models of fitness equals to one. In this paper, only one model is presented –out of 193 models, of replay-fitness equals to one. This plug-in has the power of visualizing its discovered process model as BPMN as default as depicted in Fig. 5. In addition, the plug-in provides the process tree string, and the string is depicted in Fig. 6.

TABLE II. Detailed Event Log

WI Type	State	Timestamp	Changed By
PBI (1)	New (N)	Day (1)	Team Member A
PBI (1)	Approved (A)	Day (2)	Team Member A
PBI (2)	New (N)	Day (2)	Team Member A
PBI (2)	Approved (A)	Day (3)	Team Member B
PBI (1)	Committed (C)	Day (3)	Team Member B
PBI (2)	New (C)	Day (4)	Team Member A

TABLE III. Two-Ways Summarized Event Log

Case	Trace of Events
PBI (1)	N, A, C
PBI (2)	N, A, N

Trace	Occurrence Count
N, A, C	1
N, A, N	1

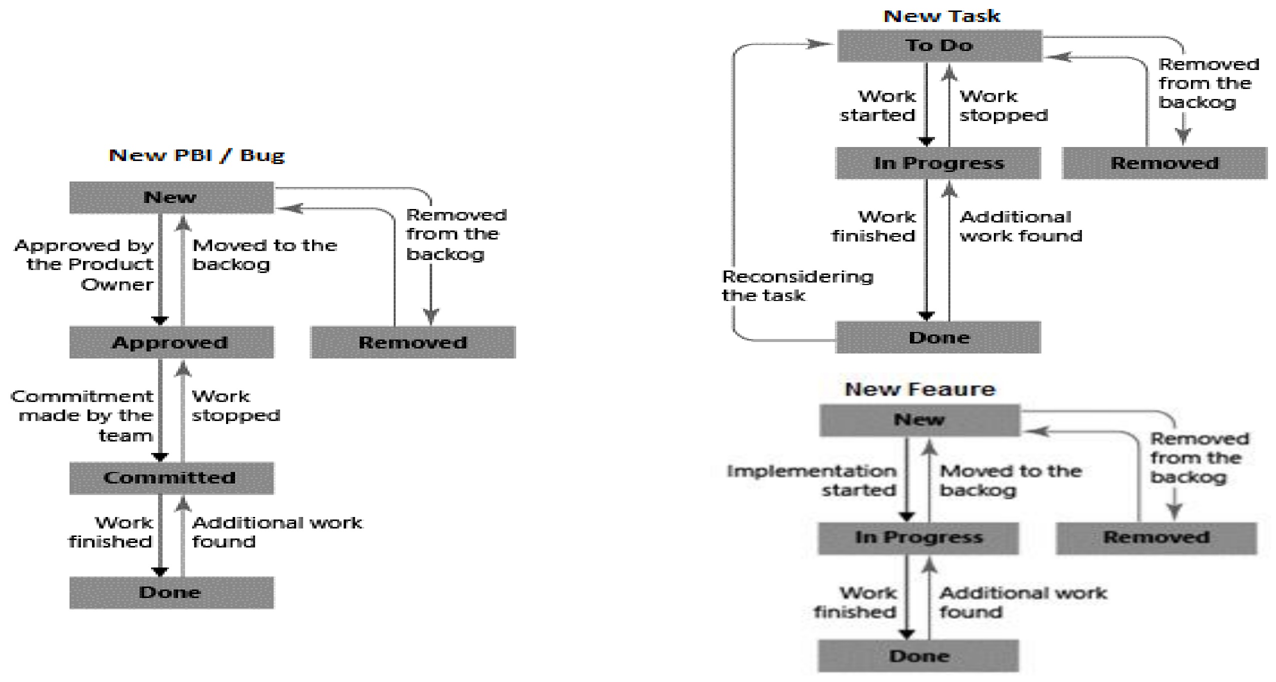

Fig. 3. Scrum Work Items Workflow [24]

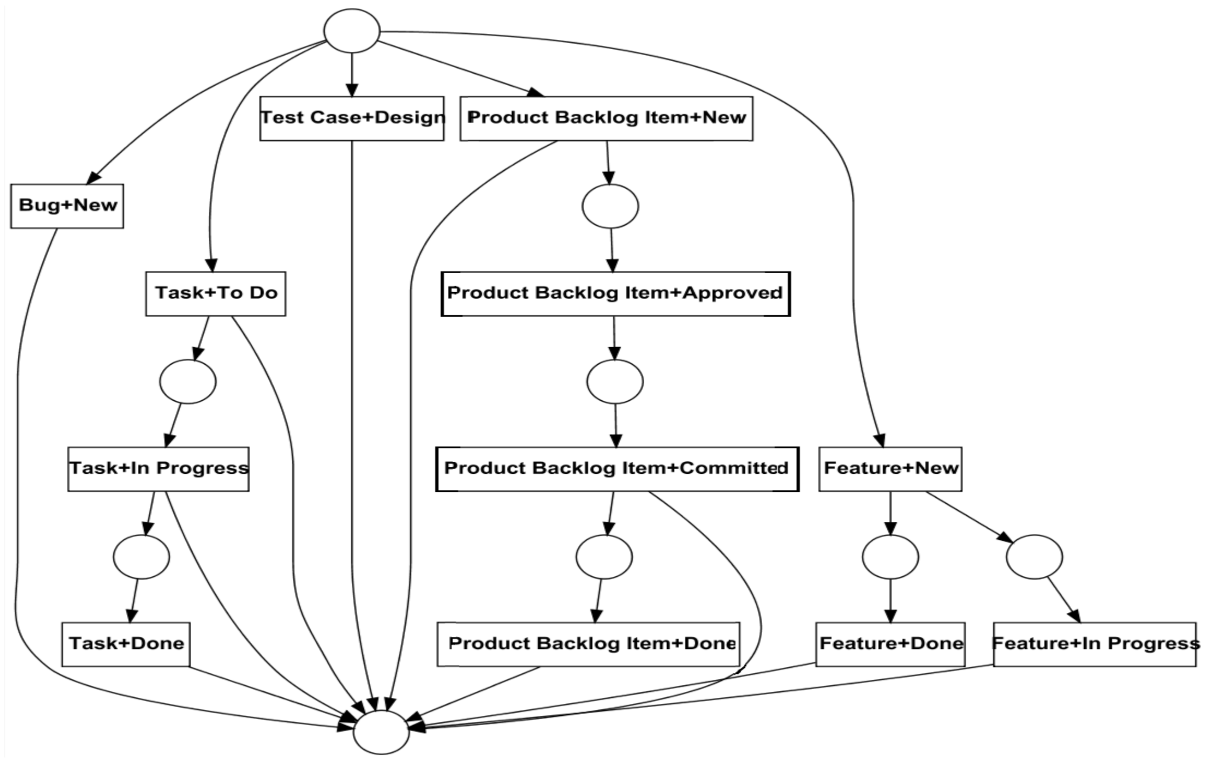

Fig. 4. Discovered Petri Net for Scrum WIs workflows

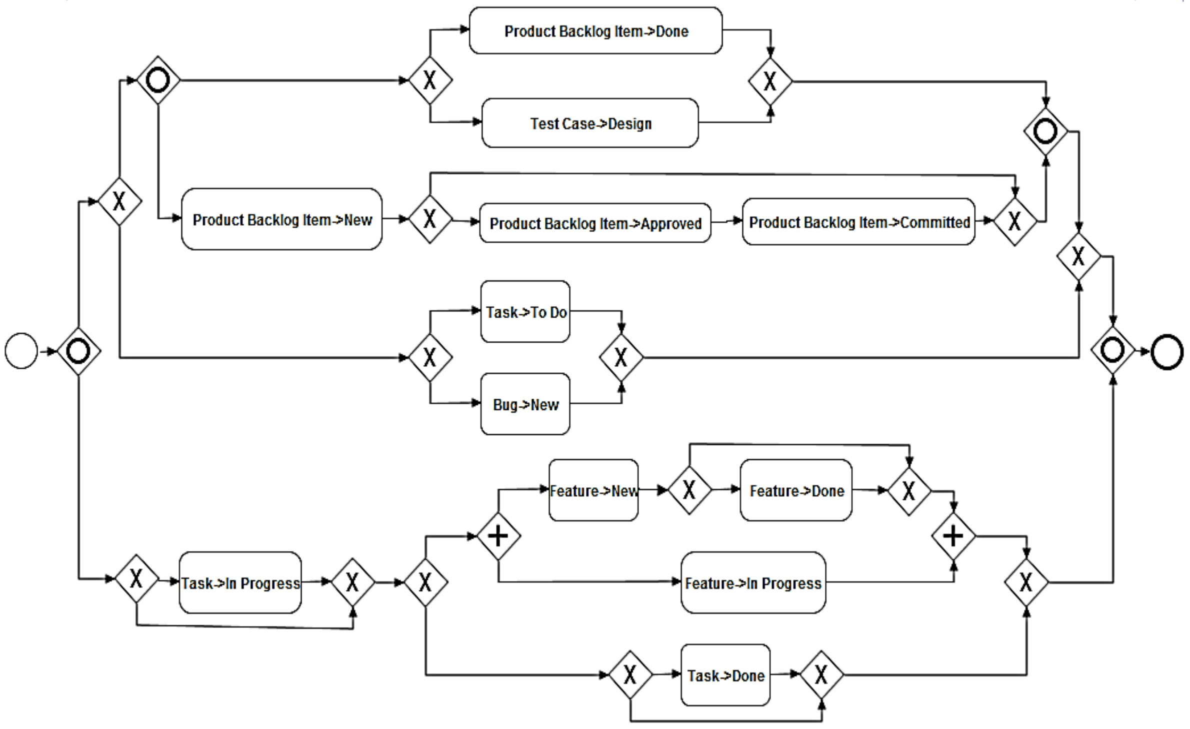
Product Backlog Item->Done
Test Case->Design
Product Backlog Item->New
Product Backlog Item->Approved
Product Backlog Item->Committed
Task->To Do
Bug->New
Feature->New
Feature->Done
Feature->In Progress
Task->In Progress
Task->Done

Fig. 5. Discovered Pareto Front as BPMN for Scrum WIs workflows

VI. Results Discussion

The discovered Pareto Front model's quality dimensions values are 1: Fitness = 1.0, 2: Precision = 0.771, 3: Generalization = 0.579, and 4: Simplicity = 0.906. These quality dimensions' values reflect the logged events are all applicable by the discovered model that is a positive aspect, the discovered process model describes the log well but it can allow much more behavior than recorded that is a down side, and the discovered process model is readable. Comparing the discovered model with Fig. 3 a huge gap away of the standard process model definition is still noticed. This can be deduced, because the scrum team is not adhering to the process definition as it is obviously discovered from the two algorithms. On one hand, the discovered PBI WI, Task WI, and Feature WI states are highly correlate to the process definition in Figure 3. On the other hand, it is found that the discovered Case WI and Bug WI states are not correlating with Figure 3 at all.

After discovering a detailed view of the WIs workflows, it is needed to know if conformance to the Scrum process as whole occurs. This study used α-algorithm [9] by applying 'Mine for a Petri Net using Alpha-algorithm' plug-in from ProM. The output result is visualized in the petri net of Fig. 7. Obviously, the discovered model is away from conformance to the Scrum process model depicted in Fig. 2. This nonconformance may be mainly due to the same reason mentioned for the WIs workflow that is the team overlooks the defined process is still applicable for noncomplying the Scrum process. The overlook here can reflect the team's lack of knowledge by the purpose of work item types and the workflow that organizes them. Deviations can be categorized into two categories. The first is fatal deviation, the second one is some events may still in progress so you can see it in the discovered model as deviated, this can be called slight deviations. Considering the fatal deviations, results showed in the discovered model that the PBI WI's parent is Bug WI as depicted in Figure 6 while in the template definition in Figure 2 PBI's parent is the Feature WI. In addition, Bug WI's child is Task WI according to the template definition not PBI as discovered. Considering the slight deviations, Feature WI has no child PBI WI. Bug WI has no child Task WI. Test Case WI has no PBI WI parent. There is no Bug WI generated from a Test Case WI, this point could not be considered a deviation as bugs could not be generated from the defined test cases but from just exploratory testing [26].

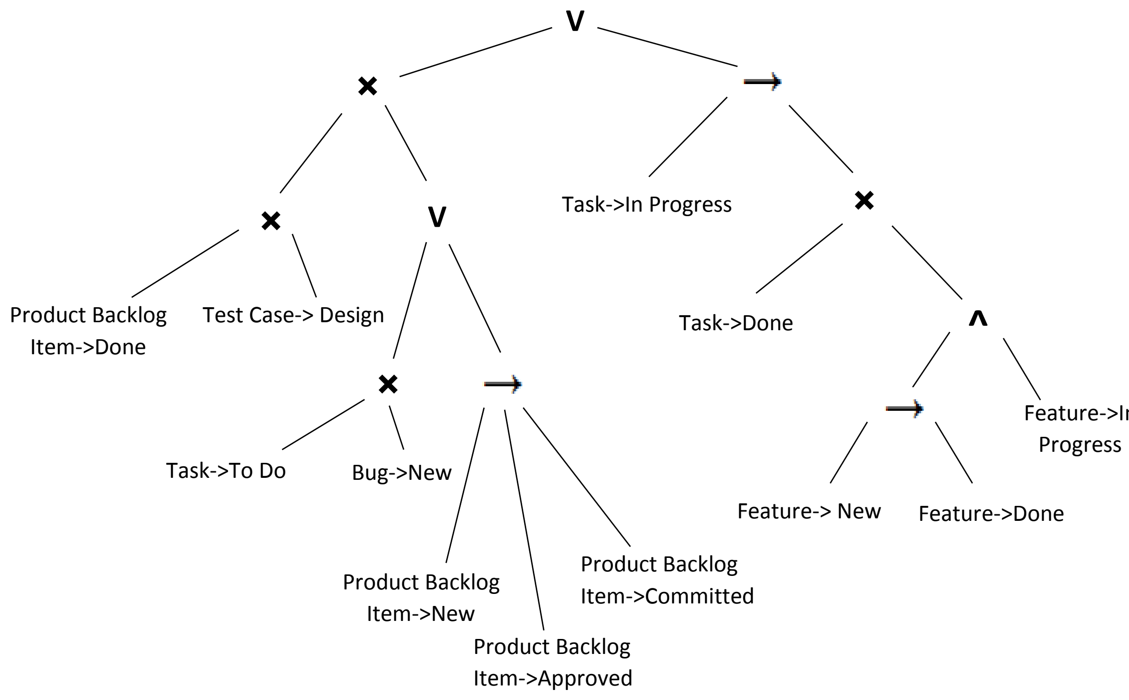

Fig. 6. Discovered Pareto Front as Process Tree for Scrum WIs workflows

VII. Future Work

We target applying conformance checking algorithms to acquire detailed knowledge about the deviations positions concerning software engineering domain. In addition, we are planning to apply enhancements algorithms to provide some recommendation in order to advise the team to conform to the process definition.

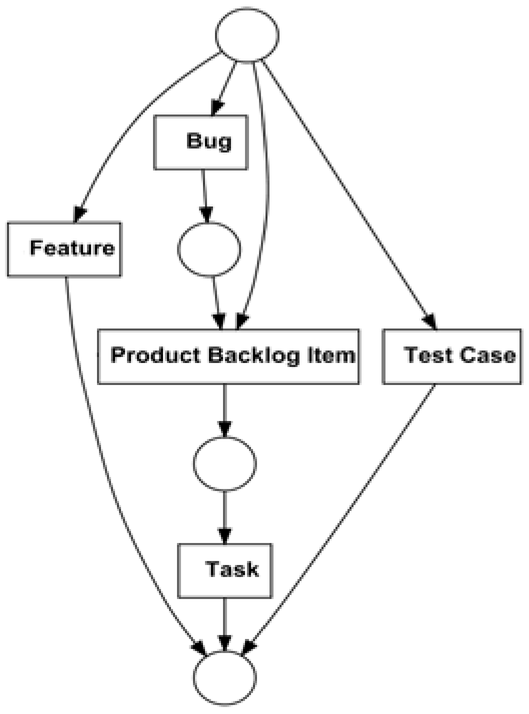

Fig. 7. Discovered Petri-Net for Scrum Process workflow

VIII. Conclusion

Research proved that Process Mining could unveil the deviation from the standard process and define the currently applied of the mostly applied Agile technique that is Scrum [23]. Software Engineering is considered an industry [27, 28, 29]. Investment in this industry is enormous and a process follow-up is needed in order to overcome any costly deviations. The study showed a significant gap between the actually applied process model and the standard process definition, which is obviously helpful. To discover the actual process model the research used ETM*d* Pareto Front algorithm [14] and α-algorithm [9]. In addition, this research used Petri nets, BPMN, and process trees to visualize the discovered models, which proves the overcome of the representational bias by separating representation and visualization.

Acknowledgment

Authors of this paper would like to thank the team members contributed ProM (www.ProcessMining.org) development, and (TU/e) for its support for such great tool. We also would like to thank the contributors who are implementing the extension plug-ins to broaden the utilization of Process Mining in their life.

This work has been done as part of the research and development efforts of CompuPharaohs software engineering team.

References

[1] W. M. P. v. d. Aalst, A. H. M. t. Hofstede and M. Weske, "Business Process Management: A Survey," in International Conference on Business Process Management, Berlin, 2003.

[2] W. v. d. Aalst and K. v. Hee., Workflow Management: Models, Methods, and Systems, MIT press, Cambridge, MA, 2002.

[3] S. Jablonski and C. Bussler, Workflow Management: Modeling Concepts, Architecture, and Implementation, London, UK: International Thomson Computer Press, 1996.

[4] D. Marinescu, Internet-Based Workflow Management: Towards a Semantic, A. Y. Zomaya, Ed., New York: Wiley Interscience, 2002.

[5] W. v. d. Aalst and P. Berens, "Beyond workflow management: product-driven case handling," in International ACM SIGGROUP Conference on Supporting Group Work (GROUP 2001), New York, 2001.

[6] W. v. d. Aalst, Process Mining - Discovery, Conformance and Enhancement of Business Processes, Springer, 2011.

[7] A. Adriansyah, B. v. Dongen and W. v. d. Aalst, "Towards Robust Conformance," in BPM 2010 Workshops, Proceedings of the 6th Workshop on Business Process Intelligence, Berlin, 2011.

[8] W. v. d. Aalst, "Mediating between modeled and observed behavior: The quest for the "right" process: Keynote," in Research Challenges in Information Science (RCIS), Paris, 2013.

[9] W. v. d. Aalst, A. Weijters and L. Maruster, "Workflow Mining: Discovering Process Models from Event Logs.," IEEE Transactions on Knowledge and Data Engineering (TKDE), 2003.

[10] A. d. Medeiros, B. v. Dongen, W. v. d. Aalst and A. Weijters, "Process Mining: Extending the α-algorithm to Mine Short Loops," International Journal Of Engineering And Computer Science (IJESC), 2004.

[11] A. Weijters, W. v. d. Aalst and A. d. Medeiros, Process mining with the heuristics miner-algorithm, Technische Universiteit Eindhoven, 2006.

[12] C. Günther and W. v. d. Aalst, "Fuzzy mining - adaptive process simplification based on multi-perspective metrics," in Business Process Management (BPM), Lecture Notes in Computer Science, Brisbane, Australia, 2007.

[13] C. Günther, Ph.D. thesis: Process Mining in Flexible Environments., Eindhoven University of Technology, 2009.

[14] J. Buijs, Flexible Evolutionary Algorithms for Mining Structured Process Models, Eindhoven: Technische Universiteit Eindhoven, 2014.

[15] A. Rozinat, A. A. d. Medeiros, C. Gunther, A. Weijters and W. v. d. Aalst, "The Need for a Process Mining Evaluation Framework," in Business Process Management Workshops, Berlin, 2008.

[16] J. Buijs, B. v. Dongen and W. v. d. Aalst, "Quality dimensions in process discovery: The importance of fitness, precision, generalization and simplicity," 2014.

[17] W. M. v. d. Aalst, Process Mining: Discovery, Conformance and Enhancement of Business Processes, Springer, 2011.

[18] Technische Universiteit Eindhoven (TU/e), "Process Mining Manifesto," 2012. [Online]. Available: http://www.win.tue.nl/ieeetfpm/doku.php?id=shared%3Aprocess_mining_manifesto. [Accessed December 2015].

[19] OMG, "Business Process Model and Notation (bpmn) version 2.0," Object Management Group, 3 January 2011. [Online]. Available: http://www.omg.org/spec/BPMN/2.0/PDF. [Accessed December 2015].

[20] A. Scheer, Business Process Engineering, Reference Models for Industrial Enterprises, Berlin: Springer-Verlag Berlin Heidelberg, 1994.

[21] W. v. d. Aalst, A. Adriansyah and B. v. Dongen, "Causal nets: A modeling language tailored towards process discovery," in CONCUR, 2011.

[22] A. Weijters and J. Ribeiro, "Flexible heuristics miner (FHM)," in Computational Intelligence and Data Mining (CIDM), 2011.

[23] D. L. Giudice, "The 2015 State Of Agile Development," Forrester Research, 2015.

[24] Microsoft, "Scrum process work item types and workflow | VS 2013," 2013. [Online]. Available: https://msdn.microsoft.com/library/jj920147%28v=vs.120%29.aspx. [Accessed December 2015].

[25] TU/e, "ProM," Process Mining Group, Math&CS department, Eindhoven University of Technology., 2015. [Online]. Available: http://www.processmining.org/prom/start. [Accessed December 2015].

[26] C. Kaner, A Tutorial in Exploratory Testing, 2008.

[27] C. Baum, The system builders: The story of SDC, System Development Corp, 1981.

[28] M. Campbell-Kelly, From Airline Reservations to Sonic the Hedgehog: A History of the Software Industry, The MIT Press, 2003.

[29] M. A. Cusumano, Microsoft Secrets: How the World's Most Powerful Software Company Creates Technology, Shapes Markets and Manages People, Touchstone, 1998.