Enhancing business process event logs with software failure data

Gruszczyński K.

Department of Information Systems, Poznań University of Economics and Business,
al. Niepodległości 10, PL 61 875 Poznań, Poland
e-mail: k.gruszczynski@live.com

Received March 01.2019: accepted March 22.2019

Abstract. Process mining techniques allow for the analysis of the real process flow. This flow might be disturbed for many reasons, including software failures. It is also possible for failure occurrence to be the consequence of the faulty process execution. A method for measuring the harmfulness of the software failure regarding business processes executed by the user would be a valuable asset for quality and reliability improvement. In this paper, we take the first step towards developing this method by providing a tool for enhancing XES event logs with failure data. We begin with an introduction to this topic and background analysis in the field of failure classification and process mining techniques supporting failure analysis. Then we present our method for merging operational and failure data. By carrying out a case study based on real data, we evaluate our tool and present the aim of our future work.

Keywords: business process modeling, exceptional process paths, software reliability, failure classification, process mining

INTRODUCTION

The aim of this paper is to provide a method for matching software failures with activities of a business process that could not be properly executed due to an unexpected software behavior. Using process mining techniques, it is possible to identify outliers in process instances that could be affected by a software defect. However, current process mining solutions do not support more advanced failure data analysis. With a tool for preparing event logs enriched with failure properties, like severity or occurrence date, we create possible paths for further research in this area.

Our article consists of four parts. In this section, we present the background for our research. It is the literature study in the field of software failure classification and failure data analysis using process mining. In the second section, we describe our method for enhancing event logs with failure data, which resulted in developing a dedicated tool for supporting this operation. Then we present results of our case study that we carried out with data from KRUK Group, one of the leading enterprises on the European debt collection market. The last section presents not only the ideas for future research but also difficulties in using our method.

Software failure classification. Collecting defects data is the key process for further analysis – preparing software reliability growth models, predicting error occurrences and estimating the cost of failures. Error information is gathered twofold: by application users and testers, who send requests with a description of encountered faults or by bug tracking mechanisms integrated with the developed software. In all cases, a well-designed scheme with detailed attribute values is a highly desired property for engineers and analysts. In literature, there are many classification schemes prepared for the fault data (characterizing defects encountered during different stages of the software implementation, such as requirements analysis, coding or testing) and failures (data connected with a particular event that raised the problem) [12]. A very important step is the data preparation – in a simplified form, it includes consistency checking and preparation of the representative sample [17].

Discovering relations between failures and faults is an essential operation – it allows to analyze the consequences and causes of errors. Thus a good classification model for complete error data analysis ought to be a combination of both attribute schemes that are common for those two data sets. The attributes presented in [17] for the defect modification requests created during various stages of an optical transmission network element are phase detected, defect type, real defect location, defect trigger, barrier analysis info. The classification presented in [12] consists of elements that are the fault, failure, and environment specific. Attributes of fault are origin, activity responsible for detection, identification stage, cause, isolation, corrective action, root cause analysis (error introduction stage). Failure attributes are mode, date and time, operation in progress, trigger, duration, extent, criticality, affected component, a detection mechanism, recovery mechanism, restoration mechanism, prevention and circumventions recommendations. An additional classification for the environment provides extra information to compare data across different sites like server and client data and application-specific attributes – for example, the number of lines of code and testing methods. Freimut *et al.* in [7] presents categorization models and processes for industrial embedded systems and describes the HP Scheme classification that consists of three main elements: origin (activity that led to detection), type, mode (reason for being a defect). A two-step model for creating the failure

categorization is presented in [16]. During the first stage of operation, attributes allowing to define symptoms of the failure are extracted. During the second step, the focus is on low-level attributes. A combination of the failure analysis and root cause analysis for high-return process improvement decisions is described in [9]. A common set of attributes is found there – origin (stage of software implementation), type, mode. When it comes to analyzing data from web applications, an example categorization is shown in [11] with logic and compatibility fault attributes. IEEE [15] developed the standard classification for software anomalies that covers attributes on the high level (i.e. type, mode, severity, description) and low level (i.e. date closed, date observed) of abstraction. It includes a relation between the fault and failure that uncovered defect.

Between statistical data models and root cause analysis, there is a wide gap that orthogonal defect classification (ODC) aims to fill [6]. It provides better analysis by developing a measurement system based on semantics. ODC defines two main attributes for classification: defect type and trigger [18]. Moreover, it introduces goals for well-defined categorization: orthogonality, consistency across phases and uniformity across products. Although there are many successful implementations of this method in many different business environments, ODC is hardly adopted between different companies – each case requires its own development [7].

To create a good classification scheme, the involvement of both statistics and domain expert is needed. Both sides are the main source of information as they bring knowledge about the well-prepared categorization model and steps that were followed to fix the actual defect. Their cooperation is the core condition to benefit from the root cause analysis: lesser defects, fault detection in the early stage of the product lifecycle, lower fixing effort and increased effectiveness [17]. A successful fault and failure classification model provides many advantages for the whole process of software development: feedback for developing software design standards, guidance for software testers, evaluation of verification and validation tools, implementation of reliability models [3]. Still, some guidelines should be followed to achieve those goals, such as those defined in [12]: to minimize the number of categories, to create open classification framework, to create mutually exclusive categories within each field, separate fault, and failure data.

Detecting failures with process mining. Software failure occurrence might cause an impact on the business process that is being executed using faulty application. Thus, in our approach, we find it highly valuable as for being able to discover automatically the consequence of the failure. With plenty of data being stored in information systems, process mining is the technique that turns this data into valuable insights [20] and allows to visualize how the business process was executed after the failure occurrence.

Process mining techniques are based on event logs – data describing how business processes are being executed. It allows to discover, monitor and improve real processes with extracting knowledge from the event log [20]. Each event log consists of traces – sets of activities that were performed during business process execution. When the software failure occurs, the effect might be observed in the trace.

The usage of process mining techniques in solving outlier detection problem was covered in [8]. Authors proposed an algorithm based on clustering and took account of concurrency constructs. It requires firstly the modeling standard paths first so anomalies could be identified in accordance with those "normal" paths.

The problem of discovering anomalies in event logs was also addressed in [2]. In this paper, the authors presented an approach based on existing tools available for process mining framework ProM. The proposed method classifies traces as anomalous or normal with respect to the so-called "appropriate model". Similarly to [1, 8], this approach discovers anomalies based on the sequence of activities in a trace.

Calderón-Ruiz and Sepúlveda in [4] took into account different types of potential failures: missing tasks, unnecessary tasks, different behavior (sequence of activities) and different timing (anomalous durations of activities). They used Performance Sequence Diagram Analysis [13] as it considers control flow and time aspects. As a result, they developed a plug-in to ProM that is able to identify potential causes of failures.

Rogge-Solti and Kasneci in [14] focused their work on temporal anomalies in a group of activities. They implemented an anomaly detection plug-in to ProM. The approach was evaluated using data from a Dutch hospital. The goal of their research was to detect anomalies in traces and to investigate temporal anomalies in the entire case.

In [5] process mining techniques were used to discover sources of failures in business processes, thus enhancing the root-cause analysis. Authors implemented a plug-in to ProM for filtering event logs. They classified sources of failures into three groups: missing tasks, unnecessary tasks, and different behavior. Their work presented a different approach to failure analysis using event log data, treating anomalies in traces not only because of failures but also as the possible source for anomalous behavior.

Methods using process mining techniques in detecting outliers might be helpful in investigating failure data as the behavior and features of a trace in the event log might be affected by a failure. On the other side, every outlier in a business process execution might be a trigger for a defect occurrence. Still, there is a lack of research taking into account typical features of a software failure log consisting of data describing system state at the runtime. This data is considered valuable in our research and we present a method to work with it.

MATERIALS AND METHODS

In this section, we present our approach to event logs enhancement with data from failure tracking systems. We developed a tool Failure Analyst (FA) in the .NET Framework to improve the process of modifying event logs. FA's main purpose to extend XES event logs (which is the correct file format to work with ProM Framework) with failure-related data. The user is able to connect proper failure types with process steps that are affected by them.

Providing this information, we make the first step towards more complex analysis: identifying the most error-prone business process activities, the most harmful failures, and the root cause analysis support.

Two files are expected to be provided to FA to work on failure analysis – XES event log and failure log – currently supported format is XLS. Fig. 1 presents a view of the developed tool. FA extracts information from those files and does the following:

1. Creates the list of available activities in the event log.
2. Allows the user to map the failure log data to properties of the implemented failure classification scheme.
3. Presents data from the failure log in a data grid with a calculated number of occurred failures for a given type, date of the first and last failure occurrence.

Working with FA requires mapping each failure type to the proper process activity and selecting the severity of each failure. As a result, in the outcome file, FA adds to each XML node information regarding occurring failures. This file is a standard event log with additional information that is added similarly to other extensions available for XES (http://www.xes-standard.org/xesstandardextensions). It is ready to be loaded to ProM, yet plugins to work with extended failure information are planned to be developed.

We implemented the simplified failure classification scheme from [15] to our tool. We selected 5 failure properties which, in our opinion, are the minimum set to describe every failure record:

1. Failure ID – the unique identifier of failure occurrence.
2. Title – makes the data readable for the user.
3. Description – helps to identify the failure origin, i.e. stack trace.
4. Observed date – the date of failure occurrence.
5. Reference – identifier for the failures of the same type, grouping information.

A case study presenting the usage of FA is described in the following section. In the article's summary, we present our ideas for the further development of FA and the core problems we have discovered during the first use in a real-world scenario.

RESULTS AND DISCUSSION

We carried out a case study using operational and failure data from KRUK Group – a debt collecting company with subsidiaries in 7 European countries. The failure data was collected from the bug tracking system (BugTracker) database, where every exception that occurs during software execution is stored. The operational data that is related to the business processes were collected from the database of the core software being used in KRUK Group – Delfin.

For our case study, we selected a real business process that we presented in our previous work [19] – verification of the client's email address data. In this paper, we made the next step with enabling process mining techniques on the data related to this process. Our first goal was to verify if the process model could be generated from the created event log (Fig. 2) using process mining. The second goal was to verify if this event log could be enhanced with software failure data for the purpose of future analysis. We used the ProM Framework and Failure Analyst software as the main tools for this case study. We selected the Alpha Algorithm [21] as the tool for creating the business process model.

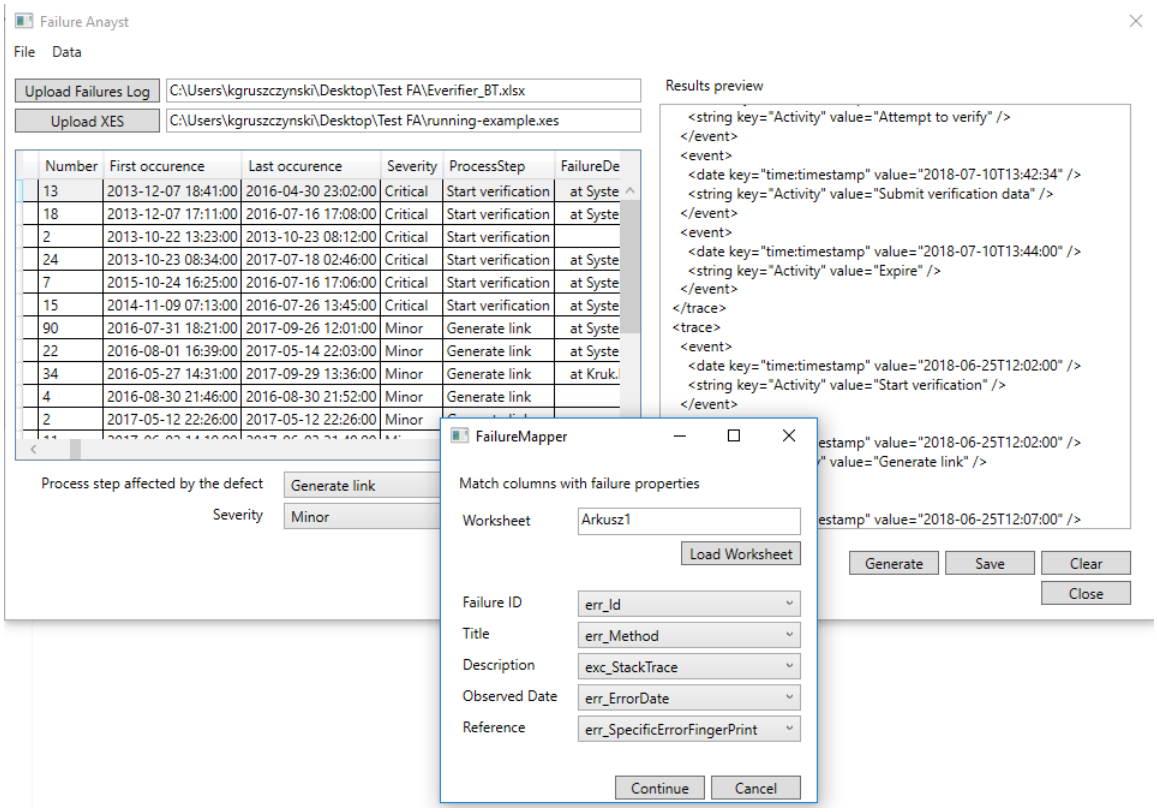

Fig. 1. The view of the developed tool – Failure Analyst.

In the next step, we created a Petri Net of the email verification business process using ProM. There were 8 unique process steps discovered by Alpha Algorithm (Fig. 3). Most activities are executed sequentially. The process ends with a gateway determining the result of the process instance. There are 3 possible process results – the user could pass the verification positively, submit incorrect data or the process instance could expire.

During the next phase, we used the Failure Analyst tool to enhance the event log with data from BugTracker system. We extracted failure data to a spreadsheet where every failure occurrence was represented by a different row. There were 38 properties describing every failure, each of them was inserted into a separate column in the spreadsheet. Our former work presents a more comprehensive description of the data extraction activity for the case study.

Finally, we loaded the failure and event log data to the Failure Analyst. The most time-consuming activity was matching each failure type with the proper process step. It required expert's knowledge and a thorough failure description analysis – based on the failure trace, where the name of application method raising the failure is written, it was possible to deduce the appropriate process activity. Also, we indicated the severity of every failure type. As a result, we received the event log data enhanced with additional information regarding failures that were affecting process activities. A sample process activity from the event log that was the most error-prone is presented in Fig. 4.

Every process activity that was matched with a failure type, in the resulting event log received new nodes describing failure types. This event log is an initial artifact for future work on data analysis using developed tools and process mining techniques.

CONCLUSIONS

Software failures might notably affect the organization's business processes [10]. In this paper, we aimed at creating a method for enhancing event logs with software failure data, thus allowing for the more advanced analysis in the field of measuring the failure impact. As a result, we developed a tool for merging failure and operational data that generates an enhanced event log. This event log is prepared to work with ProM Framework or any other tool supporting XES format. In our case study, we tested our tool with the real data from KRUK Group.

We encountered several problems while using Failure Analyst and we treat them as a trigger for the further development of our tool. Firstly, the failure schema for XES should be enabled for broader usage. By publishing it as a standard for other XES users, other tools could be prepared to work with this solution. Secondly, it isn't possible to match many process activities to one failure occurrence for now. During our case study, we encountered situations when one failure type affected more than one process step. Currently, it is also impossible to match a concrete process instance with a concrete failure. Our solution works only on failure classes and business process models. Working on concrete instances would allow for a more comprehensive root cause analysis with the possibility to observe the real process flow after the failure occurrence.

```
<trace>
   <event>
      <date key="time:timestamp" value="2018-06-25T13:05:00"/>
      <string key="Activity" value="Start verification"/>
   </event>
   <event>
      <date key="time:timestamp" value="2018-06-25T13:05:00"/>
      <string key="Activity" value="Generate link"/>
   </event>
   <event>
      <date key="time:timestamp" value="2018-06-25T13:10:00"/>
      <string key="Activity" value="Send email"/>
   </event>
   <event>
      <date key="time:timestamp" value="2018-07-09T13:02:07"/>
      <string key="Activity" value="Attempt to verify"/>
   </event>
   <event>
      <date key="time:timestamp" value="2018-07-09T13:03:34"/>
      <string key="Activity" value="Submit verification data"/>
   </event>
   <event>
      <date key="time:timestamp" value="2018-07-09T13:05:00"/>
      <string key="Activity" value="Expire"/>
   </event>
</trace>
```

Fig. 2. A sample trace from the generated event log.

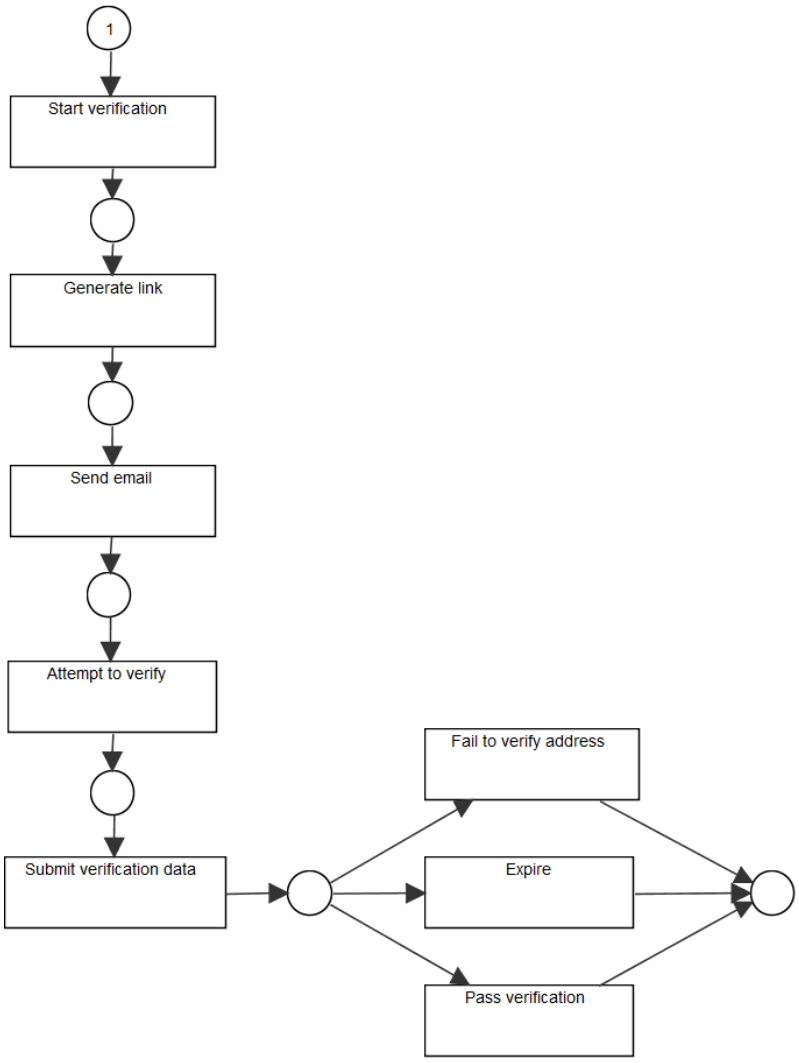

Fig. 3. The generated Petri Net of the email verification process.

```
<event>
  <date key="time:timestamp" value="2018-07-10T13:41:07" />
  <string key="Activity" value="Attempt to verify" />
  <failure>
    <string key="failure:reference" value="213415788d0c09877ec476fa33577a4d" />
    <string key="failure:severity" value="Minor" />
    <string key="failure:count" value="9" />
    <string key="failure:firstoccurence" value="2015-03-03 22:07:00" />
    <string key="failure:lastoccurence" value="2016-04-30 23:02:00" />
  </failure>
  <failure>
    <string key="failure:reference" value="575515037b6f79383bf714cf1b5d6fa6" />
    <string key="failure:severity" value="Minor" />
    <string key="failure:count" value="7" />
    <string key="failure:firstoccurence" value="2015-10-24 16:25:00" />
    <string key="failure:lastoccurence" value="2016-07-16 17:06:00" />
  </failure>
  <failure>
    <string key="failure:reference" value="0f7d9657a7b8129e0410f12efd116a09" />
    <string key="failure:severity" value="Minor" />
    <string key="failure:count" value="54" />
    <string key="failure:firstoccurence" value="2016-08-20 07:27:00" />
    <string key="failure:lastoccurence" value="2017-07-18 00:41:00" />
  </failure>
  <failure>
    <string key="failure:reference" value="d4f305e1fd11b27f1f496358b57494bb" />
    <string key="failure:severity" value="Minor" />
    <string key="failure:count" value="41" />
    <string key="failure:firstoccurence" value="2017-07-22 12:22:00" />
    <string key="failure:lastoccurence" value="2017-07-26 09:48:00" />
  </failure>
</event>
```

Fig. 4. A sample activity in the XES event log enhanced with failure data.

Our future research will focus on developing methods to work with the results of this study. We find the process of failure data analysis one of the key indicators of quality assurance. Software failures raise the cost for the organizations and users working with the unreliable software. Our main objective in all our work is to measure this cost based on the business process distortion caused by failures. This research is an important step in achieving our goal by presenting a method for preparing the proper data for process analysis.

REFERENCES

1. **Bezerra F., Weiner J. 2013**. Algorithms for Anomaly Detection of Traces in Logs of Process Aware Information Systems. Journal Information Systems, 38: 33-44.
2. **Bezerra F., Weiner J., Van der Aalst W. M. P. 2009**. Anomaly Detection using Process Mining. Enterprise, Business-Process and Information Systems Modeling. Lecture Notes in Business Information Processing, 29: 149-161.
3. **Bowen J. B. 1980**. Standard error classification to support software reliability assessment. Proceeding AFIPS '80 Proceedings of the May 19-22: 697-705.
4. **Calderón-Ruiz G., Sepúlveda M. 2013**. Automatic discovery of failures in business processes using Process Mining techniques. Available online at: https://pdfs.semanticscholar.org/69de/0d6965c25bc5861c9ff2a87a1aff8279389b.pdf
5. **Calderón-Ruiz G., Sepúlveda M. 2014**, Improving Business Processes: Failure analysis with Process Mining. Available online at: https://www.researchgate.net/profile/Guillermo_Calderon-Ruiz/publication/283344715_Discovering_the_source_of_failures_Process_mining_can_identify_problems_while_saving_time_and_money/links/5776fe0e08aeb9427e279492/Discovering-the-source-of-failures-Process-mining-can-identify-problems-while-saving-time-and-money.pdf
6. **Chillarege R., Bhandari I. S., Chaar J. K., Halliday M. J., Moebus D. S., Ray B. K., Wong M.-Y. 1992.** Orthogonal defect classification - a concept for in-process measurements, Software Engineering, IEEE Transactions on Software Engineering, 18: 943-956.
7. **Freimut B., Denger F., Ketterer M.** 2005. An Industrial Case Study of Implementing and Validating Defect Classification for Process Improvement and Quality Management. 11th IEEE Internat. Software Metrics Sympos. (METRICS'05).
8. **Ghionna L., Greco G., Guzzo A., Pontieri L. 2008**. Outlier Detection Techniques for Process Mining Applications, Foundations of Intelligent Systems: 150-159.
9. **Grady R. B. 1996**. Software Failure Analysis for High-Return Process Improvement Decisions. Available online at: http://www.hpl.hp.com/hpjournal/96aug/aug96a2.pdf
10. **Gruszczyński K., Małyszko J.** 2017. Wpływ awarii systemów informatycznych na wydajność procesów biznesowych przedsiębiorstwa. Studia Oeconomica Posnaniensia, 5: 63-78.
11. **Guo Y., Sampath S. 2008**. Web application fault classification - an exploratory study. ESEM '08 Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement: 303-305.
12. **Hecht H., Wallace D. 1996**. Error Classification and Analysis for High Integrity Software. Available online at: http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=2D1B52134F5299B30C177781223C1FF7?doi=10.1.1.541.5697&rep=rep1&type=pdf
13. **Hornix P.T.G. 2007**. Performance Analysis of Business Processes through Process Mining. Available online at: http://www.processmining.org/_media/publications/hornix2007.pdf
14. **Rogge-Solti A., Kasneci G. 2014**. Temporal Anomaly Detection in Business Processes. Business Process Management. BPM 2014. Lecture Notes in Computer Science, 8659: 233-249.
15. **IEEE 2010,** Standard Classification for Software Anomalies. Available online at: http://www.ctestlabs.org/neoacm/1044_2009.pdf
16. **Jain S., Prinja R., Chandra A., Zhang Z.-L. 2008**. Failure Classification and Inference in Large-Scale Systems: A Systematic Study of Failures in PlanetLab. Available online at: https://www.dtc.umn.edu/publications/reports/2008_08.pdf
17. **Leszak M., Perry D. E., Stoll D. 2002**. Classification and evaluation of defects in a project retrospective. The Journal of Systems and Software, 61: 173–187.
18. **Lyu M. R. 1996**. Handbook of Software Reliability Engineering. Available online at: http://www.cse.cuhk.edu.hk/~lyu/book/reliability/
19. **Perkowski B., Gruszczyński K. 2018**. The Benefits of Modeling Software-Related Exceptional Paths of Business Processes. Business Information Systems Workshops. Lecture Notes in Business Information Processing, 339: 77-85.
20. **Van der Aalst W. M.** P. **2014**. Process Mining in the Large: A Tutorial. Business Intelligence. eBISS 2013. Lecture Notes in Business Information Processing, 172: 33-76.
21. **Van der Aalst W.M.P., Weijters A.J.M.M., Maruster L. 2003**. Workflow Mining: Discovering process models from event logs. IEEE Transactions on Knowledge and Data Engineering, 16: 1128-1142.