Towards Traditional Simulation Models of Context Using Process Mining

Paolo Pileggi, Alejandro Rivero-Rodriquez and Ossi Nykänen
Department of Mathematics
Tampere University of Technology
Tampere, Finland
e-mail: {paolo.pileggi, alejandro.rivero, ossi.nykanen}@tut.fi

***Abstract*—Context (sensor) systems are hard to model: they require constant updating and insightful approaches, especially considering the increasing data volume, variety, and generation rate of contemporary networking paradigms, like the *Internet of Things*. In this paper, we argue that intelligent process models can be mined to look at the actual system activity from alternative context perspectives, i.e., perspectives observable from the sensor attributes themselves. We explain how the close relationship between the models derived using Process Mining, and Event-Driven Simulation can be exploited to help not only better understand what is happening in such systems but also provide alternative models for the intelligent solutions they support, such as context inference. We demonstrate this using a real-world example and discuss the feasibility of extending these alternative process models to be viewed as simulation. We envision automated steps that would result in traditional simulation models of context using Process Mining.**

Keywords-Performance modelling; Process Mining; Context-aware computing.

I. INTRODUCTION

Modelling context (sensor) systems is hard. A key challenge is to efficiently represent huge networks of sensors that produce a large amount of a wide variety of data at an extremely high rate, and with high real-time relevance [1]. An example network of a contemporary networking paradigm is the *Internet of Things* [2], where *machine type communication* (or *Machine-to-Machine*) [3] data must seamlessly coexist with that of high social activity.

Ideally, models of such systems should enable designers to effectively plan and appropriately respond to system activity by first being able to experiment with them. Moreover, from a computational intelligence perspective, model-driven solutions of such complex and complicated systems stand to benefit greatly from models that are more easily understood by the modellers and developers of intelligence algorithms, e.g., context inference [4, see '*prediction*'].

The success of a centralised context software service, such as the *Context Engine* previously argued by the authors [5], depends on whether the models it makes available to application developers, take into account that some applications deal with systems that are significantly more complex and complicated that others. The application developer need not only be able to interpret the models they use, in order to enhance context-awareness in their solution, but the solution must also be efficient enough to not consume too much of the resources (like device power, memory, etc.). It should be timely in its response as well.

In this paper, we explain how we use Process Mining principles to obtain models that represent alternative perspectives of the desired activity. We give the necessary background information, in Sect. II, and present the artefacts to obtain the desired models.

In Sect. III, we discuss in more detail what we mean by observing the system from the perspective of a context (sensor) attribute.

Our main contribution is then presented in Sect. IV, where we show, by way of a real-world example, an alternative model to the actual process model of the system activity. The actual model is mined directly from real recorded event log data. The advantage is that the event log reports the real activity of the system. With additional information, the event log or resulting models may be modified accordingly.

Finally, in Section V, we conclude with our discussion about the feasibility of using our models in the same vein as Event-Driven Simulation [1], which is a well-understood flexible and accessible modelling paradigm. The discussion is supported by virtue of the relationship between the Petri net formalism and the *event* (or *simulation*) graph model, discussed in the sequel. We give a description of future work that will support our vision of automated steps that would result in traditional simulation models of context using Process Mining.

II. BACKGROUND

In the mid to late 1970s, *block diagrams* [6] and *process networks* [7] encouraged the rise in popularity of process-oriented modelling. Nowadays, we refer to the method, by which to derive a structured process description from a set of actual executions, as *Process Mining* (PM).

Extracting acceptable models from a minimal set of information is one of the key challenges of PM, as is for workflow modelling [8]. Consistency between the mined process model and the actual information collected, i.e., the *event log* or *trace file*, is naturally an important criterion to measure the quality of the model.

Even though we do not set out to determine the way in which to select the PM algorithm best-suited to a particular application domain or enterprise, it remains a very relevant and important task. Wang *et al* emphasise that we do not

[1] Henceforth *Event-Driven Simulation* is referred to as *simulation*.

have a widely accepted benchmark for algorithm evaluation and comparison [9]. They emphasise that benchmarks are proposed but that such evaluation requires significant computation, is time-consuming and particularly laborious, especially for large-scale enterprises. They propose a scalable solution to evaluate, compare and rank algorithms, and give a framework for the efficient selection given a specific set of models.

Popular contemporary formalisms used to represent systems of processes are the Petri net and the Business Process Modelling Language (BPML) notation, where the latter is more focused on modelling activity within businesses [10].

Our formalism of choice is the Petri net, a well-established mathematical formalism used to model concurrency and synchronisation in systems. For a more detailed description of Petri nets, the unfamiliar reader might consider texts such as that by Bause and Kritzinger [11]. However, it suffices to understand that a Petri net is a tuple construct made up of *places*, *transitions*, *arcs* that connect these, and *tokens* that can be consumed and generated by transitions, across the arcs, from and to connected places.

We use the basic α-algorithm, originally proposed by Van der Aalst *et al* [1,9], to mine our process models. It is capable of the rediscovery of a large class of process models but it has its shortcomings:

- The actual model needs to be absent of short loops for it to be effective.
- Some identical relations may be mined for models that clearly exhibit different behaviour.
- Unique labels are assumed per Petri net definition, which may not be the case. Labelled Petri nets [12] can be used to solve this problem but it does complicate the matter tremendously.

We used *ProM* [13], a software tool that supports event log analysis and PM, offering a variety of algorithms, including the α-algorithm. To draw the process models shown in the figures we present, we used *PIPE2* [14], a Petri net analysis and visualisation tool.

There are a variety of other methods to mine processes for the sake of process discovery and conformance checking. For example, Cook and Wolf developed a Markov method for process discovery through the analysis of software processes [15]. Another example is the genetic algorithm approach by Van Eck *et al* that allows guided discovery that depends on user preference parameters, like replay fitness, precision, generalisation and simplicity [16].

A. Context models and reasoning

When we speak of context modelling, we refer to the representation (or formulation) of a system, which allows us to interpret and understand various system nuances depending on the technique and tools used. Many modelling techniques and tools already exist, and many deal specifically with contexts of systems with adaptive characteristics [17]. Perera *et al* [18] give a classification of modelling categories particularly when dealing with the *Internet of Things* paradigm, clarifying strengths and weaknesses, and mentioning suitable application domains.

An example of a modelling approach is the ASSL formal tool, an autonomic system specification language to specify management policy events originating in hardware [19]. Another is a model-based approach that adapts and isolates regular behaviours in order to introduce state transfers and global invariants [20]. In the second example, the system variables are verified at run time using model-checking.

On the other hand, context reasoning is the way to deduce (or reason about) the knowledge gained by modelling the system context. However, a modelling approach does not always allow for reasoning about the system, such as in our ASSL example.

Decision models are not new and many textbooks refer to them, especially in the field of artificial intelligence [4]. Types of models useful for reasoning include hidden Markov models, decision trees, artificial neural networks, heuristics, fuzzy reasoning and ontology, to mention only a few.

Primarily due to simplicity and relatively low resource requirements, Lim and Dey found that, in a study of 114 reviewed context-aware applications, more than half of the participants (54%) implemented rule-based reasoning [21]. The least popular of the approaches were those that applied k-nearest neighbour techniques: these are more complex and require more resources to reason effectively.

Rather broad categories of context reasoning were also singled out and compared by Perera *et al* [18]. They give taxonomy of commonly supported functionality in existing systems, highlighting the following four aspects:

- *Acquisition*, e.g., discovery, labelling, aggregation;
- *Modelling*, e.g., techniques, data storage;
- *Reasoning*, e.g., techniques, context quality;
- *Distribution*, e.g., techniques, formats, access.

However, this paper is not about these specific context modelling paradigms and reasoning techniques *per se*. It suffices to know that we present an approach to complement existing methods, ideally supporting our vision of the realisation of simulation of context using PM.

B. Context overlays using principles of Process Mining

A context can be identified by some observable value or a combination thereof. For example, using a thermometer to measure the temperature in a room, measuring the sharpness with which a person brakes when stopping at a red traffic light, or a headcount of the total number of people in a room, are all examples of what we mean by the *observable* context. Hence, we call the recording agent, the context (sensor) attribute.

A relationship exists between the attribute and the actual activity being recorded. This relationship is an equivalence class on the activity by the attribute. A simple example is where we want to know if the lights are switched on in a

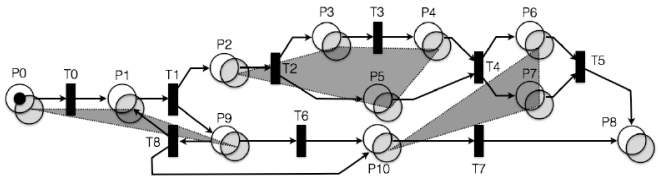

Figure 1. Reproduction of the laboratory case to illustrate the COVER (attribute-shading) over an actual process model [22].

meeting room at night. In the typical case, we would expect that when there are any people in the room, the light will be on, otherwise not. There are then two categories for the attribute value, say headcount: *no people*, and *one or more people*. Moreover, the activity process is only defined by two tasks or events, namely, *lights on* and *lights off*.

In previous work, we have called and represented this equivalence class relation as the *Context OVERlay* (COVER) [22]. Reproducing our laboratory case in Fig. 1, it suffices to understand that the COVER can be visually indicated by the shaded areas of some predefined process model representing the actual activity we are interested in. Particularly, in the figure, the COVER has four categories, where no process in the actual model is associated with more than one attribute category value but all the actual processes are assigned a value.

Then there are two dimensions to classify COVERs:

- Completeness: either *complete* or *incomplete*.
- Precision: either *precise* or *vague*.

In Fig. 1, we have an example of a complete but vague COVER because all the places in the process model have an attribute category value associated with it but some of those places share the same category value.

Ideally, for predictive ability, we would like to have a complete and precise COVER but this is unrealistic, especially given the noise and error-prone nature of the real-world. Moreover, if the system is very complicated and complex, being complete and precise does not do anything for scalability. In fact, the same model structure would apply, only additional associations must be coded. Therefore, one would expect to encounter more COVERs that are either incomplete and precise, or complete and vague. Incomplete and vague COVERs are also expected but it becomes particularly harder to deal with.

C. Simulation graph as a process model

Petri nets are very diffuse in system modelling, perhaps due to its simplicity and hence, extensibility. The formalism is usually extended through adding metadata, such as with Coloured Petri nets [23].

New modelling techniques are also developed: the *event* (or *simulation*) graph was developed to help with the efficient design of (discrete-event) simulation models [24]. Initially, Törn proposed simulation graphs as a tool for modelling simulation design [25] but shortly after, it became the first of much dedication to the topic under the guise of event graphs by Schruben and his collaborators [24, 26, 27].

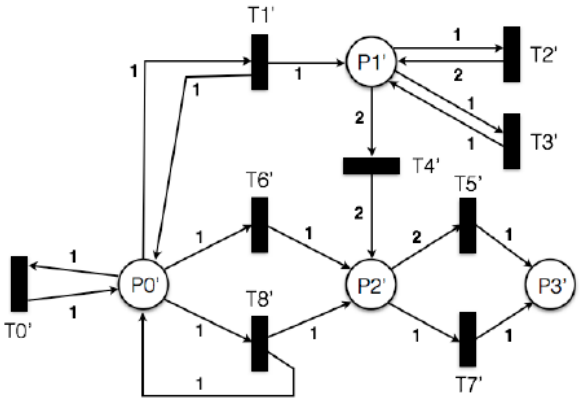

Figure 2. Reproduction of the laboratory Cover net to show the complete and vague COVER's perspective of the actual model [22].

In a short presentation, Schruben and Yücesan show how to transform a certain type of stochastic Petri net into an event graph model [26]. The latter artefact, of course, can be optimised and used to generate the simulator itself.

Not surprisingly, some authors (although only a few) have attempted to use the event graph to model their system. Liu *et al* derived event graphs from comprehensive and integrated workflow logs containing control flow information, data and resource information of a credit card application (business process) [28]. They managed to generate evaluation detail in terms of precision, generalisation and robustness. More importantly, they could simulate the process under different scenarios and analyse simulation logs for different generic problems in the specific case study, all based on the event graph model constructed.

The event graph clearly gives an augmented view, which is very useful when it comes to modelling context in the way we discuss next.

Conveniently and to the great advantage of the event graph, Yücesan and Schruben have shown how to use the event graph to perform structural equivalence tests [27], a key task associated with PM.

III. Alternative Perspective Models

A. Context sensors as observable attributes

As we just mentioned, the context (sensor) attribute is an observable agent, whose data is available in the event log that indicates the equivalence class relationship with the actual activity since the sensing happens when the activity itself is recorded in the log. However, we need to discretise the attribute value into categories, in case it is not a discrete variable already. We show examples of both discrete and continuous variables in our real-world example in Sect. IV.

In Fig. 1, if we assume that the attribute is say, location, those four categories could have been determined by the sensor agent simply indicating location labels, such as "At home", "At work", etc. Alternatively, an agent that reports

the latitude-longitude pair would be reporting a continuous variable and hence, the modeller would have to discretise it.

Discretisation can take place either through a logical selection process, where one could simply assume that, continuing our example, a certain bounded physical area would indicate a specific place, such as "At home". In the case of more complicated or advanced attributes, research on algorithms for data clustering are a classical problem and is still ongoing [29,30].

Next, we illustrate and explain an alternative model perspective using the laboratory case reproduced in Fig. 1.

B. Cover nets

By considering each of the attribute category values, we can use the actual process model covered by the COVER, as shown by the shaded areas in Fig. 1. We call the resulting process model the Cover net, since it is the Petri net model of the system activity, where each context (sensor) value is now a point of observation (represented by places in the net) for all the process types (represented by the transitions).

Specifically for the Fig. 1 example, we reproduce Fig. 2, which shows the corresponding Cover net. The algorithm is not explicitly given here but what ultimately happens is that all the system activity is preserved and, depending on the type of COVER, in this case it being complete and vague, the Cover net introduces a certain amount of deterioration in accuracy.

In this specific case, the first feature to note is that the number of places in the Cover net are bounded, i.e., depending on the number of attribute categories, the upper limit of places in the net are known. It is however possible that some attribute values are never reported in the event log, in which case, there will be no places in the net to indicate them.

Secondly, the number of transitions in the Cover net are exactly the same as those in the actual model, for this example. Actually, there could be fewer transitions (i.e., processes) in the net than in the actual model but the Cover net also captures concurrency (i.e., split and join operations in the actual model) using individual transitions. In our laboratory case, we included various types of concurrency structures on purpose to illustrate this fact.

It is clear from our example that human-understandable labels are not necessary for this modelling method to work but it should also be clear that any knowledge of the activities related to any of the transitions can be transferred between these models.

For the reader still in doubt, we refer to our previous publication explaining the approach in more detail [22].

IV. Real-world Example

Our real-world example uses Smartphone social interaction data collected by a software suite, called *nodobo*, developed by researchers at the University of Strathclyde, Scotland. They collected both device usage patterns and social interactions from *Google Nexus One* mobile phones.

Bell *et al* provide more detail about the architecture and trial study, presenting initial social network analyses [31]. The dataset (for anonymous user data) is available for free on the *nodobo* website[2].

They collected data over roughly five months, from September 2010 into February of the following year. 27 senior high school students participated. Examples of the type of data available are

- *anonymous* caller and callee **identities**,
- **call type** (*outgoing*, *incoming* and *missed)*,
- **call duration** (missed call duration is 0 minutes),
- **call time** of the activity.

Anomalies and outlier data were not removed: for example, a few students left school at the end of 2010 but continued to participate in the study, and of the mobile phones malfunctioned but remained in the study. We feel that this gives our dataset a degree of authenticity, i.e., this is the kind of data one encounters in real-world situations and it requires additional analyses to pre-process data, where possible.

A. Mined activity model

We used the α-algorithm to mine for the actual process activity. We could choose which activity we wanted to model from the types of data available in the dataset.

The categories for call type are clearly fixed as *outgoing*, *incoming*, and *missed*. For the duration and call time of day, we decided categories at our discretion, as shown in Table I. Determining the ideal clustering of data is beyond the scope of this paper.

TABLE II. Event Log data type value range per category

Category Value	*Duration (d, minutes)*	*Time of Day (hour)*
0	0	Early morning (5 – 9)
1	$0 < d < 1$	Morning (9 – 12)
2	$1 < d < 5$	Afternoon (12 – 15)
3	$5 < d < 10$	Late afternoon (15 – 18)
4	$d > 10$	Evening (18 – 21)
5	*not applicable*	Night (21 – 5)

What we noticed by selecting each data type, in turn, is that each of these resulted in fully-connected Petri nets, i.e., there is a transition connecting each place to every other, including the place itself. This is due to the nature of the system, where calls follow rules that are not very strict and easy to violate. For example, there is no rule that prevents a student from performing any type of call activity at any time of the day.

The Petri net process model for the *call duration,* as categorised in Table I, is shown in Fig. 3. There are five places P0 to P4, each corresponding to the category value indicated. For the purpose of this demonstration, we assume the *duration*-activity as that which we want to model.

Clearly, even for a small number of categories, the system can quickly become complicated to represent and understand. Process models are typically even more complicated, referred to as *Spaghetti* processes by Van der Aalst [32], amongst others.

[2] **nodobo**, Web: http://nodobo.com/, Accessed 30/4/2015.

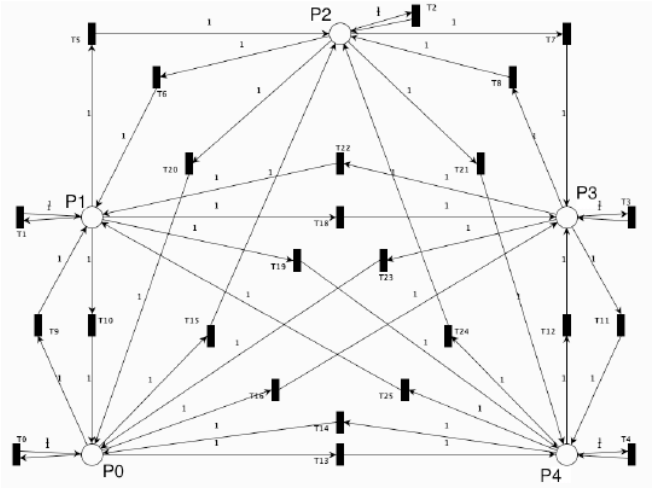

Figure 3. Call type Cover net derived from the COVER obtained from the event log, with respect to the duration categories identified

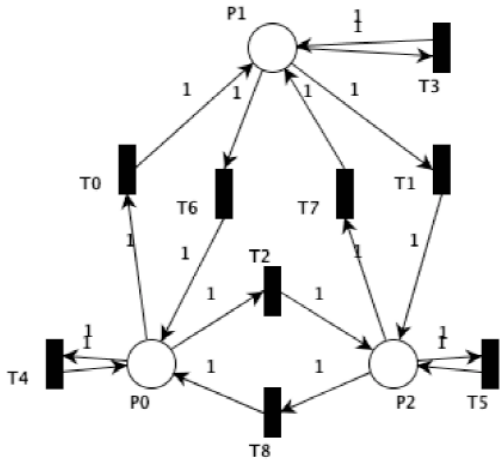

Figure 4. Petri net process model for the *call duration*, i.e., the actual process we choose to model.

B. Attribute model offering alternative perspective

First, we select the context (sensor) attribute: we decided to use the call type as the attribute since it will demonstrate the reduction of places, i.e., introducing a higher level of generality, also suggesting hierarchical attribute relationship.

Next, we obtain the cover: we processed the event log to determine the call type category values assigned to each of the places in Fig. 3.

The resulting COVER, shown in Table II (as output by processing the event log in order) is then used to determine the actual Cover net, i.e., the process model that offers the call type sensor's perspective of the duration-activity.

The call type Cover net is shown in Fig. 4. By our classification, the COVER is complete and particularly vague.

The complexity is reduced in that there are fewer places in the call type Cover net (P0=outgoing, P1=incoming, P2=missed) shown in Fig 4, than in the duration-activity process model show in Fig. 3. However, still the Cover net is a fully-connected net.

The net is fully-connected due to the nature of the system itself. Indeed, we can expect an even greater reduction in the number of transition, should the relationship between the context attributes allow this.

In brief, selection of the Cover net parameters allows reducing the amount of information in the Petri net model. This approach can be iteratively used, e.g., to gain insight to the system by a human designer, starting from a very strict selection(s) of the attributes, resulting in a simple model, and relaxing or modifying the parameter, and thus adding model complexity. Other straightforward use-cases include using Cover nets to explicitly take the ambiguity of traces into account in Petri net models, and using Cover nets to analyse the information gain of attribute selection(s), for models with *optimal* complexity.

There is also a need to introduce data statistics into the Petri net formalism used. The call type values are reported in a specific order, as show in Table II, however, PM does not distinguish between the number of times a specific process activity (or a specific sequence thereof) occurs in the trace. If it occurs once, it is included in the model. This is a strong motivation why using event graphs can add significant value, above and beyond the benefits discussed in our conclusion.

TABLE III. CALL TYPE COVER AS PRODUCED BY IN-ORDER ANALYSIS OF THE EVENT LOG

Duration category	***Call type values reported***
0	*{outgoing, incoming, missed}*
1	*{outgoing, incoming}*
2	*{incoming, outgoing}*
3	*{outgoing, incoming}*
4	*{outgoing, incoming}*

V. CONCLUSION AND FUTURE WORK

We have described a way to obtain alternative models of a system of processes using the context (sensor) attributes as indicators of actual system activity. Using principles from Process Mining, together with an understanding of how context sensors can relate to the actual system activity, we have shown that we can generate useful models that may offer lower complexity at a cost of degraded accuracy, should they be used in some model-driven solution, such as context inference.

The example using real-world data shows that modellers and solution developers can benefit significantly. The model becomes more tractable in that it is more visible. Some effort in semantic labelling, better design decisions can be made.

From our discussion about simulation graphs, there is a clear parallel between a process model mined or constructed for a context attribute, and the simulation model. Simulation is known for being easier to comprehend than most other modelling paradigms and there is a vast amount of research to support simulation optimisation, especially using event graphs. Therefore, we envision an automated approach that takes an event log and realises the simulation model for the appropriate algorithms and choice of attributes.

This approach requires work in terms of the following:

Our approach does not require the semantic process labels to be known. However, improving this would provide

clear event labels in the simulator to accommodate a better understanding for the modeller, thus complementing computational intelligence techniques, like context inference.

Determining how to evaluate the attributes will help with attribute selection, providing the ideal perspective.

Clustering research will complement understanding the better attributes to select and how they are to be categorised.

Finally, each of the variety of PM algorithms are more suitable to model different kind of system. Knowing which is better suited to mine the system's processes will complement our work by leading to a better simulator.

ACKNOWLEDGMENT

This work was financed by the EU FP7 Marie Curie Initial Training Network, MULTI-POS (Multi-technology Positioning Professionals), under grant no. 31652.

REFERENCES

[1] J-P. Dijcks, "Oracle: Big data for the enterprise," Oracle White Paper, Oracle, 2013, pp. 1-14, oracle.com.

[2] C. Poulin, *et al*, "IBM-XForce Threat Intelligence Quarterly 4Q 2014," WGL03062-USEN-00, IBM, 2014, pp. 1-16, ibm.com.

[3] T. Taleb and A. Kunz, "Machine Type Communication in 3GPP Networks: Potential, Challenges, and Solutions," Comm. Mag., IEEE, vol. 50(3), pp. 178-184, 2012, doi:10/1109/MCOM.2012.6163599.

[4] S. J. Russell and P. Norvig, "Artificial Intelligence: A Modern Approach," 3rd *ed*., Pearson, 2010, ISBN 978-0-136-04259-4.

[5] A. O. Nykänen and A. Rivero-Rodriguez, "Problems in Context-Aware Semantic Computing," Internat. J. of Interactive Mobile Technologies, vol. 8(3), 2014, pp. 32-39, doi:10.3991/ijim.v8i3.3870.

[6] T. J. Schriber, "Simulation Using GPSS," John Wiley & Sons, New York, 1974, ISBN 978-0-471-76310-9.

[7] A. A. B. Pritsker, "Modeling and Analysis Using Q-GERT Networks," John Wiley & Sons, New York, 1977, ISBN 978-0-470-99231-9.

[8] W. M. P. Van der Aalst, A. J. M. M. Weijters and L. Maruster, "Workflow mining: discovering process models from event logs," Trans. on Knowledge and Data Engineering, vol. 16(9), IEEE, 2004, pp. 1128-1142, doi:10.1109/TKDE.2004.47.

[9] J. Wang, R. Wong, J. Ding, O. Guo and L. Wen, "Efficient Selection of Process Mining Algorithms," Trans. on Services Computing, vol. 6(4), IEEE, 2013, pp. 484-496, doi:10.1109/TSC.2012.20.

[10] W. M. P. Van der Aalst, "Process Mining: Discovery, Conformance and Enhancement of Business Processes," Springer-Verlag, 2011, ISBN 978-3-642-19345-3.

[11] F. Bause and P. S. Kritzinger, "Stochastic Petri Nets - An Introduction to the Theory," 2nd *ed*., Vieweg Verlag, 2002, ISBN 978-3-528-15535-3.

[12] W. Reisig and G. Rozenberg, *eds*. "Lectures on Petri Nets I: Basic Models: Advances in Petri Nets," Lecture Notes in Computer Science, vol. 1491, Springer-Verlag, 1998, ISBN 978-3-540-65306-6.

[13] B. F. Van Dongen, A. K. A. De Medeiros, H. M. W. Verbeek, A. J. M. M. Weijters and W. M. P. Van der Aalst, "The ProM Framework: A New Era in Process Mining Tool Support," Proc. Internat. Conf. on Applications and Theory of Petri Nets, Springer-Verlag, 2005, pp. 444-454, doi:10.1007/11494744_25.

[14] N. J. Dingle, W. J. Knottenbelt and T. Suto, "PIPE2: A Tool for the Performance Evaluation of Generalised Stochastic Petri Nets," SIGMETRICS Perf. Eval. Rev., vol. 36(4), ACM, 2009, pp. 34-39, doi:10.1145/1530873.1530881.

[15] J. E. Cook and A. L. Wolf, "Discovering Models of Software Processes from Event-Based Data," Trans. on Software Engineering and Methodology, vol. 7(3), ACM, 1998, pp. 215-249, doi:10.1145/287000.287001.

[16] M. L. Van Eck, J. C. A. M. Buijs and B. F. Van Dongen, "Genetic Process Mining: Alignment-based Process Model Mutation," Lecture Notes in Business Information Processing, vol. 202, Springer International, 2015, pp. 291-303, doi:10.1007/978-3-319-15895-2_25.

[17] K. B. Balavalad, S. S. Manvi and A. V. Sutagundar, "Context Aware Computing in Wireless Sensor Networks," Proc. Internat. Conf. on Advances in Recent Technologies in Communication and Computing, IEEE, 2009, pp. 514-516, doi:10.1109/ARTCom.2009.85.

[18] C. Perera, A. Zaslavsky, P. Cristen, and D. Georgakopoulos, "Context Aware Computing for The Internet of Things: A Survey," Comm. Surveys & Tutorials, vol. 16(1), IEEE, 2014, pp. 414-454, doi:10.1109/SURV.2013.042313.00197.

[19] E. Vassev and M. Hinchey, "The ASSL Approach to Specifying Self-managing Embedded Systems," Concurrent Computing: Practice and Experience, vol. 24(16), pp. 1860-1878, John Wiley & Sons, 2012, doi:10.1002/cpe.1758.

[20] J. Zhang and B. H. C. Cheng, "Model-based Development of Dynamically Adaptive Software," Proc. Internat. Conf. on Software Engineering, ACM, 2006, pp. 371-380, doi:10.1145/1134285.1134337.

[21] B. Y. Lim and A. K. Dey, "Toolkit to Support Intelligibility in Contextaware Applications," Proc. Internat. Conf. on Ubiquitous Computing, ACM, 2010, pp. 13-22, doi:10.1145/1864349.1864353.

[22] P. Pileggi, A. Rivero-Rodriguez and O. Nykänen, "Using Context Overlays to Analyse the Role of a priori Information with Process Mining," Proc. Internat. Systems Conference, IEEE, 2015, pp. 639-644, ISBN 978-1-479-5927-3.

[23] K. Jensen and L. M. Kristensen, "Coloured Petri Nets: Modelling and Validation of Concurrent Systems," Springer, 2009, ISBN 978-3-642-00284-7.

[24] L. Schruben, "Simulation modeling with event graphs," Comm. of the ACM, vol. 26(11), 1983, pp. 957-963, doi:10.1145/182.358460.

[25] A. A. Törn, "Simulation graphs: A general tool for modeling simulation designs," Simulation, vol. 37(6), 1981, pp. 187-194, doi:10.1145/182.358460.

[26] L. Schruben and E. Yücesan, "Transforming Petri Nets into event graph models," Proc. Conf. on Winter Simulation, 1994, pp. 560-565, doi:10.1109/WSC.1994.717383.

[27] E. Yücesan and L. Schruben, "Structural and Behavioural Equivalence of Simulation Models," Trans. on Modeling and Computer Simulation, vol. 2(1), ACM, 1992, pp. 82-103, doi:10.1145/132277.132281.

[28] Y. Liu, H. Zhang, L. Chunping and J. J. Roger, "Workflow simulation for operational decision support using event graph through process mining," Journal of Decision Support Systems, vol. 52(3), Elsevier Science, 2012, pp. 685-697, doi:10.1016/j.dss.2011.11.003.

[29] A. K. Jain and R. C. Dubes, "Algorithms for Clustering Data," Prentice Hall, 1988, ISBN 0-13-022278-X.

[30] C. C. Aggarwal and C. K. Reddy, "Data Clustering: Algorithms and Applications," Chapman and Hall, 2013, ISBN 978-1-466-55821-2.

[31] S. Bell, A. McDiarmid and J. Irvine, "Nodobo: Mobile Phone as a Software Sensor for Social Network Research," Proc. Veh. Tech. Conf., IEEE, 2011, pp. 1-5, doi:10.1109/VETECS.2011.5956319.

[32] W. M. P. Van der Aalst, "Process Mining: Discovering and Improving Spaghetti and Lasagna Processes," Proc. Symp. On Computational Intelligence and Data Mining, IEEE, 2011, pp. 1-7, doi:10.1109/CIDM.2011.612946.