1 Social Network Mining from Event Logs

Van der Aalst, Reijers, and Song [8] present a foundational approach for discovering social networks from event logs recorded by enterprise information systems. While traditional process mining focuses on control-flow and performance, this work extracts organizational relationships by analyzing performer information in event logs.

The authors define four categories of metrics for constructing sociograms:

1. **Metrics based on causality**: These track how work flows between performers. The *handover of work* metric measures how often one performer's activity is directly followed by another's for the same case. The *subcontracting* metric identifies when work is delegated and returned.

2. **Metrics based on joint cases**: These measure how frequently two individuals perform activities for the same process instance, indicating collaborative relationships.

3. **Metrics based on joint activities**: These compare performer "profiles" based on which activities they execute, using distance measures like

Hamming distance and Pearson's correlation coefficient to identify similar roles.

4. **Metrics based on special event types**: These analyze events like reassignment or delegation to reveal hierarchical relationships.

The paper introduces MiSoN (Mining Social Networks), a tool that extracts event logs from systems like Staffware, converts them to a standardized XML format, and exports sociograms compatible with SNA tools such as AGNA, UCINET, and NetMiner.

A case study at a Dutch public works department demonstrates the approach using approximately 5,000 invoice handling cases with 33,000 events involving 43 employees. The analysis revealed unexpected findings including duplicate user accounts, informal performer specialization, and inefficient routing patterns causing over 17% of invoices to experience erroneous back-and-forth transfers.

The work highlights important ethical considerations regarding privacy and employee monitoring, noting that while event logs can measure individual performance, organizations must comply with data protection legislation and obtain employee consent.

2 Event Detection from Time Series Data

Guralnik and Srivastava [7] address a fundamental preprocessing challenge in temporal data mining: automatically detecting events from raw sensor time

series data. While most temporal pattern mining assumes events are already identified, this work tackles the problem of determining *when* significant behavioral changes occur—the change-point detection problem.

The authors generalize traditional approaches in two key ways:

1. **Unknown number of change-points**: Rather than requiring the number of change-points to be specified a priori, the algorithm iteratively discovers them using a likelihood-based stopping criterion.

2. **Flexible model selection**: Instead of assuming a fixed functional form between change-points, the approach uses universal approximation theory with basis functions (e.g., polynomials) and selects the best model via cross-validation.

The core algorithm uses maximum likelihood estimation (MLE) to evaluate candidate change-points. For a time series $y(t)$, the method fits piecewise models $f_i(t, \mathbf{w}_i)$ to segments and minimizes the likelihood criterion $\mathcal{L} = \sum_{i=1}^{k} m_i \log \sigma_i^2$ for heteroscedastic errors. A hierarchical splitting procedure examines each segment to determine if partitioning improves the overall likelihood significantly.

Two algorithm variants are presented:

- **Batch algorithm**: Processes complete datasets, achieving global optimization. Uses a stability threshold s to stop when likelihood improvements become negligible.

- **Incremental algorithm**: Detects change-points in real-time as data arrives, useful for online monitoring applications like highway traffic control. Uses a likelihood increase threshold δ to confirm detected changes.

Experimental evaluation on synthetic saw-tooth functions demonstrates robustness to noise when the signal-to-noise ratio is adequate ($h \geq 8$). Real-world validation uses highway loop detector data sampled at 5-minute intervals over 24-hour periods. Comparison with human visual inspection shows the algorithm produces statistically superior segmentations by the likelihood measure, avoiding the human tendency to over-segment smooth curves into piecewise linear approximations.

This work provides essential methodology for converting continuous sensor streams into discrete event sequences suitable for subsequent episode mining and pattern discovery.

3 Fault Diagnosis Using Digraph-Based Fault Trees

James, Gandhi, and Deshmukh [4] address the challenge of systematic fault diagnosis in automobile systems, extending the digraph-based fault tree methodology originally developed by Lapp and Powers for chemical process industries. The approach provides a structured method for developing fault trees that explicitly incorporates system topology.

The methodology centers on *directed graphs* (digraphs) that model rela-

tionships between system parameters. Each node represents a measurable parameter (pressure, flow rate, temperature), and directed edges capture how deviations in one parameter cause deviations in others. The authors define two types of parameter relationships:

- **Normal gain**: A positive deviation in the input causes a positive deviation in the output (and vice versa for negative deviations).
- **Inverse gain**: A positive deviation in the input causes a negative deviation in the output (and vice versa).

The fault tree development process follows two key steps:

1. **Digraph construction**: Model the system by identifying parameters and their causal relationships based on physical principles and system structure.
2. **Fault tree synthesis**: Starting from a top-level deviation (the observed fault symptom), trace backward through the digraph to identify all possible root causes, generating the fault tree structure automatically.

A significant contribution is handling *feedback loops*, which are common in control systems. The authors demonstrate how cyclic paths in the digraph can be properly resolved in the fault tree by carefully tracking the propagation of deviations through the loop.

The case study applies this methodology to a hydraulic power steering system. The digraph captures relationships among parameters including pump

pressure, flow rates, steering effort, and valve positions. For the top event "high steering effort," the resulting fault tree identifies root causes such as pump failure, low fluid level, belt slippage, and valve malfunctions. The hierarchical structure enables efficient diagnosis by guiding technicians through a logical sequence of checks.

This work demonstrates how graph-based system modeling can bridge the gap between system design knowledge and diagnostic procedures, providing a systematic alternative to experience-based troubleshooting guides.

4 ISO 26262 Safety Cases for Automotive Systems

Palin, Ward, Habli, and Rivett [6] address the emerging requirement for explicit safety cases in automotive functional safety under ISO 26262. While other safety-critical industries (nuclear, aerospace) have long required formal safety cases, the automotive domain has historically relied on implicit safety through compliance with regulations such as UN-ECE and FMVSS. The introduction of ISO 26262 changes this by requiring a functional safety case for electrical/electronic (E/E) systems.

The authors identify a critical tension in ISO 26262's safety case requirement (clause 6.4.6.2): the standard could be interpreted as merely requiring a collection of work products, encouraging "box ticking" compliance rather than genuine safety assurance. They argue that a *product safety argument*—not

just process compliance—should form the core of any safety case. The four interdependent elements of a safety case are:

- **Requirements**: Safety objectives that must be addressed, including predefined regulatory requirements and developed requirements from hazard analysis.
- **Evidence**: Information from testing, analysis, and simulation that demonstrates compliance.
- **Argument**: The logical structure showing how evidence satisfies requirements.
- **Context**: The operational, environmental, and structural conditions under which the argument is valid.

The paper proposes using the *Goal Structuring Notation* (GSN) to represent safety arguments graphically. GSN links goals (requirements), solutions (evidence), and strategies (argument) through "supported by" and "in context of" relationships. The modular extension of GSN allows safety cases to be decomposed into reusable modules that can independently stand up to scrutiny while capturing dependencies.

The proposed safety case architecture consists of hierarchical modules:

1. **Corporate Safety Policy**: Macro-level claims about a company's safety culture and management.

2. **Product Line Safety**: Arguments applicable to similar product variants.

3. **Product Safety**: Arguments specific to a particular vehicle or system, supported by Crash Protection, Particular Risks, and E/E Systems Safety sub-arguments.

For E/E systems specifically, the authors distinguish between *product arguments* (addressing safe behaviors through testing, analysis, and in-service history) and *process arguments* (providing confidence in the trustworthiness of evidence). With over 100 work products required by ISO 26262, this structured approach helps organize material into manageable modules while making explicit why the evidence supports claims of acceptable residual risk.

5 PM4Py: A Process Mining Library for Python

Berti, van Zelst, and van der Aalst [1] introduced PM4Py, an open-source Python library designed to bridge the gap between process mining and data science. While existing tools such as ProM and Apromore provide graphical interfaces suitable for non-expert users, they hamper large-scale scientific experimentation and algorithmic customization. Commercial tools like Celonis and Disco offer even less flexibility for custom algorithm development. PM4Py addresses these limitations by integrating with the Python data science ecosystem (pandas, numpy, scipy, scikit-learn), enabling seamless combination of process mining with machine learning and statistical analysis.

As detailed in subsequent work [2], PM4Py has evolved substantially, with over one million downloads. The library addresses three central disciplines:

1. **Process discovery**: Automatically constructing process models from event logs using algorithms such as the alpha miner, inductive mining, heuristics miner, ILP miner, and correlation miner.

2. **Conformance checking**: Comparing observed behavior against modeled behavior using techniques including footprints, token-based replay, alignments, and log skeleton methods.

3. **Process enhancement**: Improving existing models based on observed data, including performance analysis and decision mining.

PM4Py supports multiple standard data formats for interoperability: the XES format for traditional event logs, OCEL for object-centric event logs, PNML for Petri nets, PTML for process trees, and BPMN 2.0 for business process models. The library provides diverse visualizations including directly-follows graphs (DFGs), process trees, accepting Petri nets, and BPMN diagrams, as well as analytical views such as dotted charts and performance spectra for temporal pattern detection.

A notable feature is support for *object-centric process mining* (OCPM), which extends traditional case-centric analysis to capture complex inter-object relationships. This addresses real-world scenarios where processes involve multiple interacting entities rather than isolated cases.

The library evaluates log-model quality through multiple metrics:

- **Fitness**: How well the model can reproduce observed behavior
- **Precision**: Whether the model allows behavior not seen in the log
- **Generalization**: Whether the model captures the underlying process rather than just the observed traces
- **Simplicity**: The complexity of the resulting model

The library follows key architectural principles that maximize code reuse and experimentation:

- **Strict separation**: Objects (event logs, Petri nets, DFGs), algorithms (miners, conformance checkers), and visualizations are organized in distinct packages.
- **Factory methods**: A standardized interface provides single access points for algorithms, accepting event data and parameters while allowing variant selection (e.g., Alpha vs. Alpha+ miner).
- **Multiple data structures**: Support for event logs (lists of traces), event streams (unorganized events), and pandas DataFrames for efficient handling of large datasets.

Beyond process-focused analysis, PM4Py supports organizational mining through social network analysis (handover of work, working together, subcontracting metrics), role discovery, and resource profiling—connecting directly to the sociogram extraction work of van der Aalst et al. [8]. The library has

achieved XES certification with maximum score and has become a foundational tool for both academic research and industry applications in healthcare, finance, and IT security.

6 In-Vehicle Telematics for Driving Behavior Monitoring

Boylan, Meyer, and Chen [3] present a systematic review of 45 studies examining how in-vehicle telematics data has been used to monitor and model driving behaviors. With road traffic deaths reaching 1.35 million globally in 2016 and preventable behaviors (speeding, impaired driving, fatigue) being leading causes, telematics technology offers potential for both behavior improvement and risk assessment.

The review identifies two primary application domains:

1. **Insurance applications** (28 studies): Focused on improving claim frequency prediction models and classifying drivers as "safe" or "risky" for premium adjustment. Insurance companies use pay-as-you-drive (PAYD) and pay-how-you-drive (PHYD) schemes based on these classifications.

2. **Behavioral applications** (17 studies): Investigating how telematics feedback influences driving behavior and measuring the effectiveness of interventions.

The most commonly analyzed telematics variables were:

- **Speed** (32 studies): Measured as average velocity, time above speed limit, or speeding event counts
- **Acceleration** (32 studies): Often using thresholds (e.g., 0.3–0.7 g's) to identify "harsh" events
- **Braking** (24 studies): Similar threshold-based detection of harsh braking
- **Distance/driving time** (15 studies): Trip distance, duration, and contextual factors (urban/rural, day/night)
- **Turning/cornering** (14 studies): Lateral acceleration and turn frequency

Machine learning dominated the analysis approaches (22 studies), particularly for insurance applications. Neural networks, random forests, support vector machines, and XGBoost were used to create predictive models for crash risk and driver classification. For claim frequency prediction, Poisson regression models enhanced with neural network-derived risk scores significantly outperformed classical insurance models. Logistic regression, while slightly less accurate than complex machine learning models, was recommended for crash risk prediction due to its interpretability.

Studies on behavioral interventions found that feedback combined with financial incentives produced improvements in driving behavior, with reductions in harsh braking, harsh acceleration, and speeding. However, the review notes methodological limitations: most studies failed to account for the *hierarchical*

structure of telematics data (trips nested within drivers) and rarely controlled for demographic factors or vehicle differences.

The authors recommend that future studies use multi-level models to properly account for trip-level and driver-level variables, provide detailed demographic information, and standardize definitions of trips and harsh events to enable cross-study comparisons.

7 Fault Tree Analysis for Automotive Reliability and Safety

Lambert [5] presents the application of fault tree analysis (FTA) to automotive systems, advocating for its adoption alongside the failure modes and effects analysis (FMEA) traditionally used in the automotive industry. While FMEA is an *inductive* technique asking "what if?" to determine the effects of component failures, FTA is a *deductive* technique asking "how can something occur?" by working backward from an undesired top event to identify all possible causes.

FTA originated in the aerospace industry in the 1960s for analyzing ballistic missile systems and has since become essential for probabilistic risk assessment in nuclear power and chemical processing. A fault tree is a graphical and logical representation of event combinations using standard AND and OR gates. Basic events at the bottom of the tree represent the limit of resolution: component failures, human errors, software failures, and environmental conditions.

The paper distinguishes two types of fault events:

- **Type I**: A system element fails to perform an intended function (e.g., "air bag fails to deploy when collision occurs")
- **Type II**: A system element performs an inadvertent function (e.g., "inadvertent deployment of air bag")

The FTA methodology follows a structured sequence:

1. Define the undesired event (top event)
2. Acquire system understanding through diagrams and flow sheets
3. Establish scope, bounds, and environmental conditions
4. List assumptions
5. Construct the fault tree using logic gates
6. Perform qualitative analysis: identify single-point failures, common-cause failures, and minimal cut sets
7. Perform quantitative analysis: calculate top event probability, importance measures, and uncertainty
8. Conduct tradeoff studies and document results

A case study of a car starting system demonstrates the approach. The top event "car does not start as intended" is decomposed using success criteria: a car successfully starts when the engine crankshaft rotates as intended AND adequate combustion occurs in all cylinders. Taking the Boolean complement

yields the failure logic. The resulting fault tree identified 35 minimal cut sets, including 19 single-point failures. Notably, the park neutral switch and oxygen sensor were identified as single-point failures—safety devices that paradoxically affect reliability.

Lambert emphasizes that FTA's value lies in showing logical progressions and system interactions not displayed in traditional FMEA, while FMEA can identify candidate undesired events for FTA. With increasing automotive electronics in engine control, cruise control, and braking systems, FTA is particularly well-suited to address complex failure mode interactions—complementing the digraph-based approach described by James et al. [4].

References

[1] A. Berti, S.J. van Zelst, and W.M.P. van der Aalst, "Process Mining for Python (PM4Py): Bridging the Gap Between Process- and Data Science," in *Proceedings of the 1st International Conference on Process Mining (ICPM 2019), Demo Track*, pp. 13–16, 2019.

[2] A. Berti, S. van Zelst, and D. Schuster, "PM4Py: A Process Mining Library for Python," *Software Impacts*, vol. 17, p. 100556, 2023.

[3] J. Boylan, D. Meyer, and W.S. Chen, "A Systematic Review of the Use of In-Vehicle Telematics in Monitoring Driving Behaviours," *Accident Analysis and Prevention*, vol. 199, p. 107519, 2024.

[4] A.T. James, O.P. Gandhi, and S.G. Deshmukh, "Fault Diagnosis of Automobile Systems Using Fault Tree Based on Digraph Modeling," *International Journal of System Assurance Engineering and Management*, vol. 8, no. 2, pp. 494–504, 2017.

[5] H.E. Lambert, "Use of Fault Tree Analysis for Automotive Reliability and Safety Analysis," in *Proceedings of the 2004 SAE World Congress*, Detroit, Michigan, 2004.

[6] R. Palin, D. Ward, I. Habli, and R. Rivett, "ISO 26262 Safety Cases: Compliance and Assurance," in *Proceedings of the 6th IET International Conference on System Safety*, 2011.

[7] V. Guralnik and J. Srivastava, "Event Detection from Time Series Data," in *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '99)*, pp. 33–42, 1999.

[8] W.M.P. van der Aalst, H.A. Reijers, and M. Song, "Discovering Social Networks from Event Logs," *Computer Supported Cooperative Work*, vol. 14, pp. 549–593, 2005.