

# BayesInsights: Modelling Software Delivery and Developer Experience with Bayesian Networks at Bloomberg

Serkan Kirbas  
Bloomberg  
London, United Kingdom  
skirbas@bloomberg.net

Federica Sarro  
UCL, Bloomberg  
London, United Kingdom  
f.sarro@ucl.ac.uk

David Williams  
UCL, Bloomberg  
London, United Kingdom  
david.williams.22@ucl.ac.uk

## Abstract

As software in industry grows in size and complexity, so does the volume of engineering data that companies generate and use. Ideally, this data could be used for many purposes, including informing decisions on engineering priorities. However, without a structured representation of the links between different aspects of software development, companies can struggle to identify the root causes of deficiencies or anticipate the effects of changes.

In this paper, we report on our experience at Bloomberg in developing a novel tool, dubbed BayesInsights, which provides an interactive interface for visualising causal dependencies across various aspects of the software engineering (SE) process using Bayesian Networks (BNs). We describe our journey from defining network structures using a combination of established literature, expert insight, and structure learning algorithms, to integrating BayesInsights into existing data analytics solutions, and conclude with a mixed-methods evaluation of performance benchmarking and survey responses from 24 senior practitioners at Bloomberg.

Our results revealed 95.8% of participants found the tool useful for identifying software delivery challenges at the team and organisational levels, cementing its value as a proof of concept for modelling software delivery and developer experience. BayesInsights is currently in preview, with access granted to seven engineering teams and a wider deployment roadmap in place for the future.

## CCS Concepts

• **Computing methodologies** → **Bayesian network models**; • **Social and professional topics** → *Software management*; • **Software and its engineering** → *Software development process management*.

## Keywords

Developer experience, Bayesian networks, software delivery

## 1 Introduction

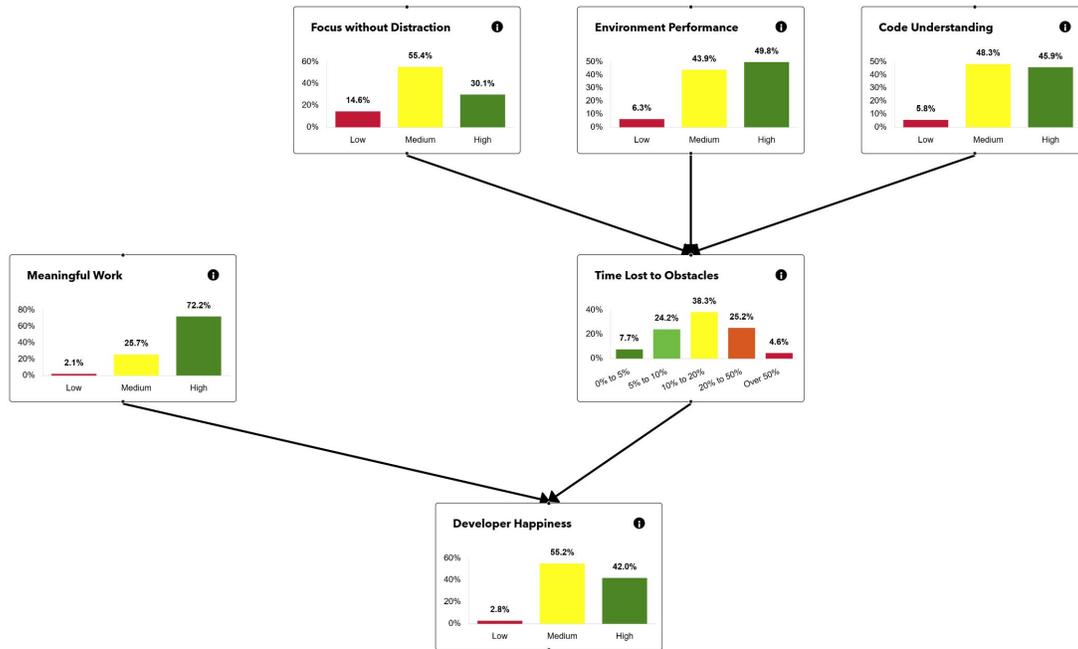
At Bloomberg, the DevX team distributes quarterly surveys to software engineers regarding the state of the firm’s software engineering (SE) practice. The survey comprises Likert-scale questions capturing engineers’ perceptions of their work environment, including satisfaction and engagement, and covers topics such as how meaningful they find their work, how much time is lost due to obstacles, the quality of peer feedback, as well as aspects of the SPACE framework [14] and the DevOps Research and Assessment (DORA) metrics [10, 13]. The results of these surveys, along with contribution metrics, are visualised via dashboards and reporting tools. While such dashboards provide visibility into delivery performance, they do not explain why certain outcomes occur or how changes in

one area might affect others. For example, a change in one metric, such as reduced deployment frequency, may reflect improvements in related factors like CI/CD or automated testing, making interpretation ambiguous [13]. As a result, teams can struggle to estimate trade-offs or predict the impact of engineering changes.

In response to this, we investigated the use of Bayesian Networks (BNs) for causal analysis in SE practices. We propose an interactive tool, dubbed BayesInsights, to model causal and probabilistic dependencies among key SE factors at Bloomberg. We use data from one internal engineering survey (20 questions, >2,000 responses) as the source for our BNs, in which each question is represented as a graph node (with the node states corresponding to the question options). We derive the structure of links between these nodes using a hybrid and iterative modelling approach consisting of ① reviewing established literature (§2.1), ② refinement through expert opinion (§2.2), and ③ validation using structure learning algorithms (§2.3).

Figure 1 illustrates an example of a BN in BayesInsights (with mocked probability distributions to protect sensitive data). Here, we can see that factors such as *focus without distraction*, *environment performance*, and *code understanding* influence *time lost to obstacles*, while both *time lost to obstacles* and *meaningful work* contribute directly to *developer happiness*. With BayesInsights, users can conduct impact analyses through “what-if” scenarios by selecting evidence. For instance, users can estimate the extent to which minimising time lost to obstacles would improve developer happiness, or how maximising development environment performance cascades through the system to influence outcomes.

We conducted a mixed-methods evaluation of BayesInsights that combines quantitative performance benchmarking with qualitative feedback collected through a post-session survey following focus group sessions with 28 senior Bloomberg practitioners (24 of whom responded). BayesInsights demonstrated both practical responsiveness and strong perceived usefulness: 95.8% reported that BayesInsights helps identify delivery challenges at the team or organisational level, and 75% found the outputs easy to interpret. Moreover, 79.2% would use or recommend the tool, and many suggested avenues for improvement, motivating further development. BayesInsights is currently deployed in early access to seven engineering teams, with a development roadmap in place for wider deployment in future. This paper provides an in-depth description of our BN creation methodology and insights into the value of causal modelling for SE practice at a large-scale software company.



**Figure 1: The DevEx Bayesian Network in BayesInsights showing causal links among factors influencing developer happiness. Users can explore “what-if” scenarios by selecting a bar within a node, fixing that outcome and updating the network to reflect its impact on related factors. Mocked probability distributions are shown to protect sensitive data; values are illustrative only.**

## 2 Bayesian Network Structure Definition

For the initial implementation of BayesInsights, we developed two BNs based on internal engineer survey data. The first BN models **software delivery performance**, represented by two dimensions: **throughput**, which captures the speed and frequency of delivering changes, and **stability**, which reflects the reliability of changes. Throughput is modelled using *change lead time*, *deployment frequency*, and *failed deployment recovery time*, while stability is represented by *change failure rate*. These metrics form the core model and are influenced by other upstream engineering practices. The second BN modelling **DevEx (Developer Experience)**<sup>1</sup> and the factors that shape it (Figure 1) is derived from other aspects covered in the internal engineering survey, such as perceptions of *meaningful work*, *time lost to obstacles*, and *focus without distraction*.

Constructing Directed Acyclic Graphs (DAGs) for BNs in SE contexts presents significant challenges [35]. Software delivery metrics exhibit complex interdependencies that are rarely observed in isolation. Hidden confounders, including organisational culture, toolchain maturity, and team composition, can create spurious associations between variables. This challenge is compounded by the inherent limitations of SE data. Surveys are prone to interpretation and recall bias, and the temporal nature of software development processes further complicates causal inference, as relationships between variables may differ across teams and evolve with project or organisational changes. These factors collectively render naive,

<sup>1</sup>In this context, DevEx refers to the satisfaction and engagement of engineers within their work environment.

data-only structure learning approaches unreliable and highlight the necessity of a multi-source methodology to construct credible DAG structures for causal inference [6, 35].

To devise our network structures, we adopted a hybrid modelling approach combining *theoretical insights from the literature*, *expert-defined causal links*, and *algorithmic learning techniques*, following prior work on integrating expert input with data-driven methods [26]. This process involved an iterative cycle of structure proposal, expert validation, and refinement. With this approach, we aim to achieve a comprehensive, context-aware representation of causal dependencies that goes beyond what standard DevOps metric dashboards provide. We detail each step below.

### 2.1 Established Literature

We began with a foundational DAG based on the DORA metrics [10] and their established causal links. Additional nodes were created from factors measured in one of Bloomberg’s quarterly internal engineering surveys, which captures engineering outcomes, processes, and DevEx. Standard definitions of factors covered in this study are provided in our online repository [11]. Each node in the BN maps to a specific survey question, meaning every node represents a measurable aspect of engineering practice. For example, the question “*Technical debt did not significantly impact my ability to complete new work*” was mapped to the node ‘*Tech Debt Impact*’. We hypothesised causal links between nodes using a process adapted from Pearl’s causal hierarchy and established BN methods [12, 29]. We used the following causal modelling idioms [28]: ① **Cause-Consequence**

**idiom**, where one variable plausibly influences another through a physical, productive, or intentional mechanism; and ② **Definition/Synthesis idiom**, where a concept is directly determined by, or made up of, its components. We defined a causal hypothesis for each proposed link in the DAG either using the idioms or through prior literature, and then evaluated them using expert opinion and algorithmic learning techniques described in the following sections.

## 2.2 Expert Insight Survey

After developing our node-link hypotheses based on the established literature, we sought expert insight to refine our notions and account for Bloomberg-specific practices. To achieve this, we conducted a structured survey that covered directional cause-and-effect relationships among the factors measured in the internal engineer survey. Each question, representing a possible link between two factors in the causal graph, asked whether one factor directly influences the other in practice. For example, a question assessing the link *Time Lost to Obstacles*  $\rightarrow$  *Developer Happiness* was phrased as “To what extent does time lost to obstacles influence developer happiness?” All questions featured the same set of options for level of influence: “Strong”, “Moderate”, “Weak”, “None”, and “Not sure/No opinion”. In total, our survey covered 24 potential relationships and allowed experts to suggest any additional ones.

Participants were purposely selected from Bloomberg’s DevX team ( $n = 8$ ) for their domain expertise and organisation-wide contextual understanding, with an average of 10 years of experience at Bloomberg. As part of their roles, they regularly work with engineering metrics and development workflows, making them well-suited to assess the relationships in question. While the sample size is small, this is typical in expert surveys where the focus is on depth of insight rather than broad statistical generalisation [38].

After collecting responses, we calculated a weighted score for each proposed relationship based on participants’ influence ratings. Responses marked as “Strong” were given a weight of 1.0, “Moderate” as 0.8, “Weak” as 0.2, and all other responses (“None” or “Not sure”) as 0. These values were averaged across all responses to produce a final score for each relationship, and after further consultation with participants, we set 0.70 as the minimum score for a relationship to be retained in the final BNs. Based on the scoring results, two relationships were removed, and three were introduced, yielding the final structures of the expert-refined BNs. To protect sensitive expert knowledge and internal development practices, the survey questions and results are omitted from this study.

## 2.3 Structure Learning Algorithms

To further refine the DAG structures, we applied the Hill-Climbing (HC) and Peter-Clark (PC) structure-learning algorithms (described in §6) to the internal engineering survey data to empirically estimate the BN structures. These methods detect statistical dependencies in the data and surface additional candidate edges not previously suggested by theory or expert input. To account for sampling variability, we used bootstrap resampling, repeatedly generating new samples from the data and assigning a confidence score to each recovered edge. This provided a stability check, highlighting consistently supported links versus those likely due to noise. We evaluated

the resulting structures by computing Bayesian Information Criterion (BIC) scores, which reward goodness of fit while penalising unnecessary complexity (with lower values indicating better model fit). We compared the BIC scores for the HC and PC structures to those of the expert-refined models and found that the expert models performed best, followed by HC, and finally PC. HC showed partial overlap with the expert structures and was most informative when constrained by DORA definitions, suggesting potential refinements worth expert consideration, whereas PC added little value due to weak overlap and a poorer BIC. This pattern is consistent with studies showing that constraint-based learners degrade quickly with noisier data and that adding hard constraints improves HC accuracy [34]. Finally, considering that scoring algorithms cannot be relied upon to guarantee causal validity [22], the expert-refined structures, which also achieved the best BIC scores, remained the most reliable foundation, with HC outputs used only as edge candidates for refinement subject to expert review.

## 2.4 Finalised Network Structure & Conditional Probability Table Computation

Not all sources we explored were given equal weight in determining the final network structures. We prioritised the DORA report [10] and expert opinions when deciding which relationships to include and how confident we were in them. The strongest relationships are those supported by multiple high-quality sources. By starting with a solid foundation based on DORA and carefully integrating established literature, expert feedback, and algorithmic results, our models are both grounded in theory and responsive to real-world data, while avoiding overreliance on any single source.

After defining the final model structures, each node was parameterised by computing Conditional Probability Tables (CPTs) to capture the dependencies between nodes. To estimate these probability distributions, we used the internal engineering survey data (20 questions, >2,000 responses). Since the data stem from Likert-scale questions, we treat them as discrete categorical variables. For root nodes (those without parents), probabilities were estimated as the marginal distribution:  $P(X = x) = \frac{\text{count}(X=x)}{N}$ , where  $N$  is the total number of valid observations. For child nodes, conditional probabilities were estimated from co-occurrence counts with parent configurations, e.g., a node with two parents  $A$  and  $B$ :  $P(X = x \mid A = a, B = b) = \frac{\text{count}(X=x, A=a, B=b)}{\text{count}(A=a, B=b)}$ . To address sparsity in rare or unobserved parent-child combinations, we applied Bayesian-Dirichlet equivalent uniform (BDeu) smoothing, preventing zero-probability CPT entries. Here,  $P = p$  denotes a specific joint configuration of all parent nodes of node  $X$ :  $P(X = x \mid \text{Parents} = p) = \frac{\text{count}(X=x, P=p) + \alpha}{\text{count}(P=p) + \alpha K}$ , where  $\alpha$  is the equivalent sample size controlling the strength of the Dirichlet prior, and  $K$  is the number of discrete states of the child node.

## 3 Implementation & Integration

After finalising the structures and CPTs for both the software delivery performance and DevEx BNs, our goal was to present them in an intuitive way so early-access users within the DevX team could experience them. We used a fully open source tech stack: BayesInsights is implemented as a client-server architecture built

with Django [9] and Django Ninja [8], exposing RESTful endpoints which return BN structures and process evidence-based inference queries. The backend retrieves and preprocesses the internal engineering survey data before model training and inference using pgmpy [2, 31]. The front-end for visualising and interacting with the BNs is implemented in TypeScript. Networks are rendered using React Flow [32], with nodes displayed as Chart.js [4] bar charts showing probability distributions. This UI was integrated alongside existing metric dashboards, inheriting robust access control policies and complementing other software analytics solutions.

We designed BayesInsights as an interactive tool. It presents a dynamic, directed graph in which key metrics are represented as connected nodes, forming a BN. It visualises how different engineering metrics influence one another based on historical survey data, enabling users to conduct powerful “what-if?” scenario analysis to identify the most valuable areas to target to improve delivery performance/engineer satisfaction. For example, with the software delivery performance BN, managers can investigate how a change in lead time might affect their team’s delivery throughput. Each node shows the probability distribution of outcomes and updates in real time as users select evidence, triggering a back-end query and re-rendering the network with updated probabilities.

## 4 Evaluation

To evaluate BayesInsights, we conducted quantitative performance testing to assess its practical usability and gathered qualitative insights from focus groups with prospective users to establish its value in providing insights into software delivery practice.

### 4.1 Performance Testing

To assess the responsiveness of the deployed BayesInsights service, we conducted performance testing using the Hyperfine [20] benchmarking tool for single-request latency and Locust [24] for concurrent load testing. For single inference requests, the average response time was 24 ms. Under load with 50 concurrent users, median response times were under 40 ms. These results confirm that the tool is suitable for real-time exploration of BNs.

### 4.2 User Study

**4.2.1 Methodology.** To understand the value of BayesInsights, we conducted a series of focus group sessions. Each 45-minute session began with a short presentation familiarising participants with the tool, including the challenge it addresses, the modelling method, and its potential for reasoning about delivery performance. This was followed by a live demo during which participants were introduced to the interface and taught how to enter evidence and interpret the outputs. Participants were then given access to BayesInsights and encouraged to explore scenarios on their own. Finally, feedback was collected through a short, anonymous questionnaire comprising 22 questions: 4 related to participants’ roles, development experience, and familiarity with software delivery and DevEx metrics, and 18 eliciting their opinions on the tool in terms of interpretability, model quality and trust, usefulness and practical application, overall feedback, and suggestions for improvements. The full set of questions is provided in our repo [11].

**4.2.2 Results.** We conducted seven sessions, each with 3–5 participants. In total, 28 attended the sessions, and 24 completed the questionnaire. Given the potential managerial value of BayesInsights, we primarily targeted senior practitioners: Of those who completed the questionnaire, 21 had more than 10 years of experience, while the remaining three had 4–10 years. This cohort spanned 8 Team Leads, 6 Software Managers, 5 Product Managers, 4 Software Engineers (including one DevX Engineer), and 1 Manager of Coaching.

Responses indicated that participants found BayesInsights accessible and easy to understand. Most (75%) stated that the outputs were easy to interpret, and a large majority (83.3%) clearly understood how the model visualised changes, particularly how altering one metric affected others in what-if analyses. Participants valued how the tool clearly visualised relationships between metrics and highlighted its dynamic nature, with one praising the “ability to change inputs and get immediate feedback” and another remarking that “seeing the most significant impacts [helps] with prioritising improvements.” This functionality was regarded as particularly useful because it was “grounded in real data” and aligned with practical ways of reasoning about delivery challenges. Trust in the tool was also positive, with 70.9% of respondents expressing confidence in its outputs when reasoning about software delivery and DevEx.

Almost all respondents (95.8%) reported that it was useful for understanding delivery challenges at a team or organisational level. Many mentioned contexts in which they would use BayesInsights, such as reviewing team practices (37.5%), supporting leadership decisions (62.5%), identifying root causes of delivery issues (50%), and running what-if scenarios (62.5%). A strong majority (79.2%) also indicated that they would use or recommend BayesInsights in their own teams. Notably, 25% of participants reported that even within the 15-minute demonstration period, the tool helped them generate concrete ideas for improving delivery performance (e.g., “advocating for fewer distractions and more release ease”, “exploring ways to enable focus and deep work with [my] team”). This highlights immediate value and suggests that prolonged use could help teams identify additional avenues to improve their practices.

Participants also identified a few areas for improvement. The most frequent suggestion was to incorporate natural language summaries of the tool’s outputs, offering clearer explanations of what had changed and why, along with possible actions informed by internal documentation on best practices. Another recurring request was for a results-first interface layout, in which inputs and outputs are shown above, with the BN diagram presented below as a reference. Several participants expressed interest in having an indication of the scale of and confidence in the influence between factors, and in being able to view their team’s data alongside the overall network for comparison. Overall, participants’ suggestions highlight the need to foreground explainability, transparency, and practicality when using causal techniques in a user-facing tool.

## 5 Journey & Outcomes

BayesInsights is the result of a joint effort between Bloomberg and University College London (UCL), a long-standing partnership that has led to several cutting-edge industrial projects and publications [40]. The project was completed over the summer of 2025 with participation from the DevX Insights team at Bloomberg, as well

as one professor, one Ph.D. student and six master's students from UCL Computer Science via the Industry Exchange Network (IXN) programme [37]. UCL members were onboarded as SE contractors at Bloomberg to contribute to the design and development of BayesInsights and to conduct the user studies to evaluate it. As of October 1, 2025, BayesInsights was made available in early access to seven teams within DevX, and development roadmaps have been put in place to continue its development towards a state suitable for wider deployment across the organisation. Although this implementation is in the early stages, feedback received throughout the project has been highly encouraging, with keen interest from senior engineers, team leads, and managers. We posit that a major component of this project's success is the nature of industrial-academic collaboration, which enabled academics to explore and evaluate techniques not yet deemed industry-standard in a practical setting using large-scale engineering data. Through this partnership, we demonstrated the value of causal techniques for modelling SE delivery and developer experience in a real-world setting.

## 6 Related Work

This section discusses SE metrics, outlines challenges in interpretation, and reviews causal modelling approaches, focusing on BNs, their alternatives, and structure learning in SE modelling.

**Interpreting Software Metrics.** Software delivery performance is often benchmarked with DORA metrics [10], along with measures such as test coverage, code review speed, and technical debt management [10, 39]. DevEx metrics complement delivery outcomes by capturing the human side of engineering work, with frameworks like SPACE covering satisfaction, performance, and collaboration [10, 14, 25]. These metrics provide valuable benchmarks, but they are usually tracked separately. Delivery data often comes from pipelines and incidents [19], while DevEx relies on surveys or HR systems, which makes them difficult to integrate.

Software teams are also complex socio-technical systems shaped by many confounders. As noted in Goodhart's Law [16], focusing on individual indicators can be misleading. In addition, the lack of formal causal reasoning limits the ability to perform systematic, data-driven analysis and hinders the exploration of how specific changes may influence engineering outcomes [3]. This gap highlights the potential of BNs, which can model dependencies explicitly and enable data-driven reasoning under uncertainty.

**Bayesian Networks in Software Engineering.** BNs model probabilistic dependencies between variables in a DAG and support forward and backward causal reasoning [17, 23], making them powerful for root-cause analysis and scenario exploration under uncertainty [12]. Rather than evaluating metrics in isolation, BNs represent interacting influences and propagate their effects throughout a system. In SE contexts, BNs have been applied to defect prediction, effort estimation, and project management, but these have typically been narrow in scope and challenging to generalise across teams or projects [7, 33]. Little work has addressed how BNs can unify delivery metrics with human-centred outcomes [7, 26]. Given evidence that delivery and DevEx are coupled, BNs are promising for modelling trade-offs and dependencies across both domains.

**Alternative Techniques for Causal Reasoning.** Researchers have used statistical causal inference (SCI) methods for causal

reasoning in SE, such as propensity score matching (PSM) and difference-in-differences (DiD). A recent mapping study found only 25 SE papers (published between 2010 and 2022) using any SCI method, with fewer than 10 applying the classical designs mentioned above [35]. Most of them focused on code quality or defects, with little attention to human-centred outcomes like developer satisfaction. Another review of 45 causal studies in SE found that two used DiD, one used PSM, and five used BN models [3]. While valuable, DiD and PSM rely on assumptions (i.e., that all relevant group influences are measured, and that groups affected and unaffected by an intervention would have otherwise followed the same trajectory, respectively) that often do not hold in SE contexts [1, 21, 30].

**Structural Learning Algorithms.** Building appropriate BN structures poses a significant challenge. Prior reviews on BN applications in SE have highlighted that most structures rely solely on expert knowledge, with few using data-driven approaches [26, 36]. Such reliance limits reproducibility and risks embedding bias. While data-driven learning offers an alternative, SE survey data can be noisy and biased, making reliable structure recovery difficult [27]. Thus, some studies advise constraining algorithms with expert input and carefully verifying results [22]. Two main families of algorithms exist. Constraint-based methods, such as the Peter-Clark (PC) algorithm, use independence tests to remove unsupported edges, producing graphs that avoid many false links but risk missing true ones [22]. In contrast, score-based methods, such as the Hill Climbing (HC) algorithm, search for graphs balancing fit and simplicity [18]. Although these methods capture more relationships, they risk overfitting data and including spurious edges. Hybrid algorithms combine the two, using constraint tests to narrow candidates before applying scoring to refine them [22]. In practice, researchers rely on (i) how well the network explains new data, often quantified using the Bayesian Information Criterion (BIC), which balances model fit against structural complexity to avoid overfitting [22], (ii) stability checks such as bootstrapping to see which edges consistently reappear [15], and (iii) expert review of whether inferred links make sense [5, 26].

In this work, we first construct a hypothesised DAG informed by literature, and refine it with expert input. For further refinement, we apply structure learning (PC and HC) with bootstrapping to assess edge stability, and compare models using BIC scoring.

## 7 Conclusion

This study presents BayesInsights, an interactive tool developed and released in preview at Bloomberg that uses Bayesian Networks to allow users to explore causal relationships across various aspects of the software engineering process. Our testing and prospective user evaluations demonstrate the tool's practical viability and effectiveness in revealing bottlenecks and high-impact areas of improvement in software engineering performance and developer satisfaction.

## Data Availability

As the BN has been developed within Bloomberg's internal systems and repositories, data and source code are not publicly accessible. However, the technologies we used in the project are all open source, and we explained in detail how to implement such a tool in

practice, so that future academics and practitioners alike can adopt it. Moreover, we make available the questionnaire we distributed to carry out the user evaluation [11].

## Acknowledgments

We thank all Bloomberg engineers who participated in our user studies and the UCL alumni *Shahzeb Ahmad, Mizbah Celik, Sultan Insan Geosrinov Muhammad Rifqi, Narmin Mustafali, Ting-Yun Wang, and Farhan Zaki*, who contributed to this project at Bloomberg through the UCL Industry Exchange Network (IXN).

This work received ethics approval from Bloomberg and University College London (LREC no.: 2025-1647).

## References

- [1] Joshua D. Angrist and Jörn-Steffen Pischke. 2009. *Mostly Harmless Econometrics: An Empiricist's Companion*. Princeton University Press. <http://www.jstor.org/stable/j.ctvc4j72>
- [2] Ankur Ankan and Johannes Textor. 2024. pgmpy: A Python Toolkit for Bayesian Networks. *Journal of Machine Learning Research* 25, 265 (2024), 1–8. <http://jmlr.org/papers/v25/23-0487.html>
- [3] Patrick Chadbourne and Nasir U. Eisty. 2023. Applications of Causality and Causal Inference in Software Engineering. In *2023 IEEE/ACIS 21st International Conference on Software Engineering Research, Management and Applications (SERA)*. 47–52. doi:10.1109/SERA57763.2023.10197835
- [4] Chart.js Developers. 2026. Chart.js. <https://www.chartjs.org/>. Accessed: March 25, 2026.
- [5] Anthony C. Constantinou, Yang Liu, Kiattikun Chobtham, Zhigao Guo, and Neville K. Kitson. 2021. Large-scale empirical validation of Bayesian Network structure learning algorithms with noisy data. *International Journal of Approximate Reasoning* 131 (2021), 151–188. doi:10.1016/j.ijar.2021.01.001
- [6] R. Daly, Q. Shen, and S. Aitken. 2011. Learning Bayesian networks: approaches and issues. *The Knowledge Engineering Review* 26, 2 (May 2011), 99–157. doi:10.1017/S0269888910000251
- [7] Abraham L. R. de Sousa, Cleidson R. B. de Souza, and Rodrigo Q. Reis. 2022. A 20-year mapping of Bayesian belief networks in software project management. *IET Software* 16, 1 (2022), 14–28. doi:10.1049/sfw2.12043
- [8] Django Ninja Developers. 2026. Django Ninja. <https://django-ninja.dev/>. Accessed: March 25, 2026.
- [9] Django Software Foundation. 2026. Django. <https://www.djangoproject.com>. Accessed: March 25, 2026.
- [10] DORA Research. 2024. *2024 Accelerate State of DevOps Report*. Technical Report. Google Cloud. <https://dora.dev/research/2024/dora-report/>
- [11] Kirbas et al. 2026. BayesInsights Additional Materials Repository. <https://github.com/SOLAR-group/bayesinsights-bloomberg>. Accessed: March 25, 2026.
- [12] Norman Fenton and Martin Neil. 2012. *Risk Assessment and Decision Analysis with Bayesian Networks* (1st ed.). CRC Press, Inc., USA.
- [13] Nicole Forsgren, Jez Humble, and Gene Kim. 2018. *Accelerate: The Science of Lean Software and DevOps Building and Scaling High Performing Technology Organizations* (1st ed.). IT Revolution Press.
- [14] Nicole Forsgren, Margaret-Anne Storey, Chandra Maddila, Thomas Zimmermann, Brian Houck, and Jenna Butler. 2021. The SPACE of Developer Productivity: There's more to it than you think. *Queue* 19, 1 (March 2021), 20–48. doi:10.1145/3454122.3454124
- [15] Nir Friedman, Moises Goldszmidt, and Abraham Wyner. 1999. Data analysis with bayesian networks: a bootstrap approach. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence* (Stockholm, Sweden) (UAI'99). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 196–205.
- [16] C. A. E. Goodhart. 1984. Problems of Monetary Management: The UK Experience. In *Monetary Theory and Practice: The UK Experience*. Macmillan Education UK, London, 91–121. doi:10.1007/978-1-349-17295-5\_4
- [17] David Heckerman. 1999. *A tutorial on learning with Bayesian networks*. MIT Press, Cambridge, MA, USA, 301–354.
- [18] María José Hernández-Molinos, Angel J. Sánchez-García, Rocío Erandi Barrientos-Martínez, Juan Carlos Pérez-Arriaga, and Jorge Octavio Ocharán-Hernández. 2023. Software Defect Prediction with Bayesian Approaches. *Mathematics* 11, 11 (2023). doi:10.3390/math11112524
- [19] Lyndsi Hughes and Vanessa Jackson. 2021. A Framework for DevSecOps Evolution and Achieving Continuous-Integration/Continuous-Delivery (CI/CD) Capabilities. Carnegie Mellon University, Software Engineering Institute's Insights (blog). <https://doi.org/10.1184/R1/13954388.v1>
- [20] hyperfine Developers. 2026. hyperfine. <https://github.com/sharkdp/hyperfine>. Accessed: March 25, 2026.
- [21] Guido W. Imbens and Donald B. Rubin. 2015. *Causal Inference for Statistics, Social, and Biomedical Sciences: An Introduction*. Cambridge University Press, USA.
- [22] Neville Kenneth Kitson, Anthony C. Constantinou, Zhigao Guo, Yang Liu, and Kiattikun Chobtham. 2023. A survey of Bayesian Network structure learning. *Artificial Intelligence Review* 56, 8 (2023), 8721–8814. doi:10.1007/s10462-022-10351-w
- [23] Daphne Koller, Nir Friedman, Lise Getoor, and Ben Taskar. 2007. Graphical Models in a Nutshell. In *Introduction to Statistical Relational Learning*. The MIT Press, 13–56. doi:10.7551/mitpress/7432.003.0004
- [24] Locust Developers. 2026. Locust. <https://github.com/locustio/locust>. Accessed: March 25, 2026.
- [25] André N. Meyer, Earl T. Barr, Christian Bird, and Thomas Zimmermann. 2021. Today Was a Good Day: The Daily Life of Software Developers. *IEEE Transactions on Software Engineering* 47, 5 (2021), 863–880. doi:10.1109/TSE.2019.2904957
- [26] Ayse Tosun Misirli and Ayşe Başar Bener. 2014. A mapping study on bayesian networks for software quality prediction. In *Proceedings of the 3rd International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (Hyderabad, India) (RAISE 2014)*. Association for Computing Machinery, New York, NY, USA, 7–11. doi:10.1145/2593801.2593803
- [27] Jefferson Seide Molléri, Kai Petersen, and Emilia Mendes. 2020. An empirically evaluated checklist for surveys in software engineering. *Information and Software Technology* 119 (2020), 106240. doi:10.1016/j.infsof.2019.106240
- [28] Martin Neil, Norman Fenton, and Lars Nielsen. 2000. Building large-scale Bayesian networks. *Knowledge Engineering Review* 15, 3 (2000), 257–284. doi:10.1017/S0269888900003039
- [29] Judea Pearl. 2009. *Causality: Models, Reasoning and Inference* (2nd ed.). Cambridge University Press, USA.
- [30] Judea Pearl, Madelyn Glymour, and Nicholas P. Jewell. 2016. *Causal Inference in Statistics: A Primer*. Wiley. 102–103 pages.
- [31] pgmpy Developers. 2026. pgmpy. <https://pgmpy.org/>. Accessed: March 25, 2026.
- [32] React Flow Developers. 2026. React Flow. <https://reactflow.dev/>. Accessed: March 25, 2026.
- [33] Thiago Rique, Emanuel Dantas, Mirko Perkusich, Kyller Gorgônio, Hyggo Almeida, and Angelo Perkusich. 2024. Insights into the Applications of Bayesian Networks in Software Engineering. In *2024 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*. 1–6. doi:10.23919/SoftCOM62040.2024.10721646
- [34] Marco Scutari, Catharina Elisabeth Graafland, and José Manuel Gutiérrez. 2019. Who learns better Bayesian network structures: Accuracy and speed of structure learning algorithms. *International Journal of Approximate Reasoning* 115 (2019), 235–253. doi:10.1016/j.ijar.2019.10.003
- [35] Julien Siebert. 2023. Applications of statistical causal inference in software engineering. *Inf. Softw. Technol.* 159, C (July 2023), 16 pages. doi:10.1016/j.infsof.2023.107198
- [36] Ayse Tosun, Ayşe Basar Bener, and Shirin Akbarinasaji. 2017. A systematic literature review on the applications of Bayesian networks to predict software quality. *Software Quality Journal* 25, 1 (March 2017), 273–305. doi:10.1007/s11219-015-9297-z
- [37] UCL. 2026. UCL Industry Exchange Network (UCL IXN). <https://www.ucl.ac.uk/engineering/computer-science/collaborate/ucl-industry-exchange-network-ucl-ixn>. Accessed: March 25, 2026.
- [38] Linda C. van der Gaag, Silja Renooij, Hermi J. M. Schijf, Armin R. Elbers, and Willie L. Loeffen. 2012. Experiences with Eliciting Probabilities from Multiple Experts. In *Advances in Computational Intelligence*. Springer Berlin Heidelberg, Berlin, Heidelberg, 151–160. doi:10.1007/978-3-642-31718-7\_16
- [39] Yuqing Wang, Mika Mäntylä, Zihao Liu, and Jouni Markkula. 2022. Test Automation Maturity Improves Product Quality – Quantitative Study of Open Source Projects Using Continuous Integration. doi:10.48550/arXiv.2202.04068
- [40] David Williams, James Callan, Serkan Kirbas, Sergey Mechtaev, Justyna Petke, Thomas Prideaux-Ghee, and Federica Sarro. 2024. User-Centric Deployment of Automated Program Repair at Bloomberg. In *Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Practice* (Lisbon, Portugal) (ICSE-SEIP '24). Association for Computing Machinery, New York, NY, USA, 81–91. doi:10.1145/3639477.3639756