Retrospective: Data Mining Static Code Attributes to Learn Defect Predictors

Tim Menzies

September 23, 2025

The Open Source Revolution (2004-2007)

The Portland Context

- Born from open source culture in Portland, Oregon
- *"We wore no suite and tie in our photos. We did not comb our hair"*
- Philosophy: `svn commit -m "share stuff"` will change SE research
- **Key Insight**: Walking around Chicago's Grant Park (2004)
  - **Tim Menzies** and **Jelber Sayyad** lamented: *"Must do better... Why don't we make conclusions reproducible?"*

The Radical Idea

- In 2025 hard to believe "reproducible SE" was radical
- **Lionel Briand** (2006): *"no one will give you data"*
- Yet we persisted...

Birth of PROMISE Project & Early Success

Two-Part Vision:

1. **Annual conference** on predictor models in SE (to share results)
2. **Repository** of 100s of SE datasets: defect prediction, effort estimation, Github issue close time, bad smell detection

Growth Trajectory:

- Repository grew large; moved to **Large Hadron Collider** (Seacraft data at Zenodo)
- Research students ran weekly sprints scouring SE conference tables of content
- **Gary Boetticher**, **Elaine Weyuker**, **Thomas Ostrand**, **Guenther Ruhe** joined steering committee $\rightarrow$ prestige for growth

PROMISE vs MSR:

- **MSR**: Gathering initial datasets (**Devanbu [Dev15]**)
- **PROMISE**: Post-collection analysis, consistent data upload and re-examination **[Rob10]**

Early Results:

- Other areas struggled with reproducibility, we swam in an ocean of reproducibility
- Papers applied elaborate tool sets to COC81, JM1, XALAN, DESHARNIS datasets
- First decade: Numerous successful papers using consistent data re-examination

The 2007 Paper's Core Contribution

Research Question: Can data mining algorithms learn software defect predictors from static code attributes?

Why This Matters:

- *"Software quality assurance budgets are finite while assessment effectiveness increases exponentially with effort"* **[Fu16]**
- *"Software bugs are not evenly distributed across a project"* **[Ham09]**, **[Ost04]**, **[Mis11]**
- Defect predictors suggest where to focus expensive methods

Counter-Arguments Addressed:

1. *"Specific metrics matter"* (1990s heated debates: McCabe vs Halstead)
2. *"Static code attributes do not matter"* (**Fenton & Pfleeger**, **Shepperd & Ince**)

Menzies's 1st Law: Specific metrics do not matter

"Specific metrics do not always matter in all data sets. Rather, different projects have different best metrics."

Supporting Evidence:

- Feature pruning experiment on **3 dozen metrics across 7 datasets**
- Results: Pruning selected just **2-3 attributes per dataset**
- **No single attribute** selected by majority of datasets
- Different projects preferred different metrics (McCabe vs Halstead vs lines of code)
- Theoretical debates of 1990s (metric X vs metric Y) proven empirically unfounded

Menzies's Corollary:

"To mine SE data, gather all that can be collected (cheaply) then apply data pruning to discard irrelevancies."

Practical Impact:

- Changed SE data mining methodology from "careful metric selection" to "gather everything, prune later"
- Influenced thousands of subsequent studies using this approach

Menzies's 2nd Law: Group metrics

"Static code attributes do matter. Individually, they may be weak indicators. But when combined, they can lead to strong signals that outperform the state-of-the-art."

Supporting Evidence:

- **Fenton & Pfleeger**: Same functionality, different constructs → different measurements
- **Shepperd & Ince**: Static measures often "no more than proxy for lines of code"
- **Our Response**: Stress-tested these views by documenting baselines, then showing detectors from static attributes **much better** than baselines
- **Key Finding**: Multi-attribute models outperformed single-attribute models

Key Quote: *"Paradoxically, this paper will be a success if it is quickly superseded."*

Unprecedented Success Metrics

Citation Impact:

- **2016**: Most cited paper (per month) in software engineering
- **2018**: 20% of Google Scholar Software Metrics IEEE TSE papers used PROMISE datasets **[Men07]**
- **Current**: 1924 citations (paper) + 1242 citations (repository)

Industrial Adoption:

- **Wan et al. [Wan20]**: 90%+ of 395 commercial practitioners willing to adopt defect prediction
- **Misirli et al. [Mis11]**: 87% defect prediction accuracy, 72% reduced inspection effort, 44% fewer post-release defects
- **Kim et al. [Kim15]**: Samsung Electronics API development - 0.68 F1 scores, reduced test case resources

Comparative Analysis with Static Tools

Rahman et al. [Rah14] Comparison:

- **Static analysis tools**: FindBugs, Jlint, PMD
- **Statistical defect prediction**: Logistic regression models
- **Result**: *"No significant differences in cost-effectiveness were observed"*

Critical Advantage:

- Defect prediction: Quick adaptation to new languages via lightweight parsers
- Static analyzers: Extensive modification required for new languages
- **Implication**: Broader applicability across programming ecosystems

Evolutionary Applications (2007-2025)

Extended Applications:

- **Security vulnerabilities [Shi13]**
- **Resource allocation** for defect location **[Bir21]**
- **Proactive defect fixing [Kam16]**, **[LeG12]**, **[Arc11]**
- **Change-level/just-in-time prediction [Yan19]**, **[Kam13]**, **[Nay18]**, **[Ros15]**
- **Transfer learning** across projects **[Kri19]**, **[Nam18]**
- **Hyperparameter optimization [Agr18]**, **[Che18]**, **[Fu17]**, **[Tan16]**

Research Evolution:

- From binary classification to multi-objective optimization
- From release-level to line-level prediction (**Pornprasit et al. [Por23]** - TSE Best Paper 2023)

The Four Phases of Repository Lifecycle

Phase Evolution:

1. *"Data? Good luck with that!"* - Resistance and skepticism
2. *"Okay, maybe it's not completely useless."* - Grudging acknowledgment
3. *"This is the gold standard now."* - Required baseline, field norms
4. *"A graveyard of progress."* - Stifling creativity, outdated paradigms

The Problem:

- Decade 2: Continued use of COC81 (1981), DESHARNIS (1988), JM1 (2004), XALAN (2010)
- **Editorial Policy Change**: Automated Software Engineering journal now desk-rejects papers based on 2005 datasets

Slide 10: Menzies's 3rd Law & Transfer Learning

Menzies's 3rd Law:

"Turkish toasters can predict for errors in deep space satellites."
Supporting Evidence: - **Transfer learning research [Tur09]**: Models from **Turkish white goods** successfully predicted errors in **NASA systems** - Expected: Complex multi-dimensional transforms mapping attributes across domains - **Reality**: Simple nearest neighboring between test and training data worked perfectly - **Implication**: *"Many distinctions made about software are spurious and need to be revisited"*
Broader Transfer Learning Success: - Cross-domain prediction often works better than expected - Suggests universal patterns in software defect manifestation - Questions assumptions about domain-specific modeling requirements

Slide 11: Menzies's 4th Law & Data Reduction

Menzies's 4th Law:

"For SE, the best thing to do with most data is to throw it away."
Supporting Evidence: - **Chen, Kocaguneli, Tu, Peters, and Xu et al.** findings across multiple prediction tasks: - **Github issue close time**: Ignored 80% of data labels **[Che19]** - **Effort estimation**: Ignored 91% of data **[Koc13]** - **Defect prediction**: Ignored 97% of data **[Pet15]** - **Some tasks**: Ignored 98-100% of data **[Che05]** - **Startling result**: Data sets with thousands of rows modeled with just **few dozen samples [Men08]**
Theoretical Explanations: - **Power laws** in software data **[Lin15]** - **Large repeated structures** in SE projects **[Hin12]** - **Manifold assumption** and **Johnson-Lindenstrauss lemma [Zhu05]**, **[Joh84]**
Caveat: Applies to regression, classification, optimization - generative tasks may still need massive data

Slide 12: Menzies's 5th Law & LLM Reality Check

Menzies's 5th Law:

"Bigger is not necessarily better."
Supporting Evidence - LLM Hype Analysis: - Systematic review [Hou24]: 229 SE papers using Large Language Models - **Critical finding**: Only **13/229 around 5%** compared LLMs to other approaches - *"Methodological error"* - other PROMISE-style methods often better/faster **[Gri22]**, **[Som24]**, **[Taw23]**, **[Maj18]**
Tree-based Models Superiority: - Grinsztajn et al. [Gri22]: *"Why do tree-based models still outperform deep learning on typical tabular data?"* - **Johnson & Menzies [Joh24]**: *"AI over-hype: A dangerous threat (and how to fix it)"*
Trading Off Complexity: - Scalability vs. privacy vs. performance **[Lin24]**, **[Fu17]** - Often simpler methods provide better cost-effectiveness - **Personal Pattern**: *"Year later, I have switched to the simpler approach"* **[Agr21]**, **[Tan16]**, **[Fu16]**

Slide 13: Menzies's 6th Law & Data Quality Paradox

Menzies's 6th Law:

"Data quality matters less than you think."

Supporting Research: - **Shepperd et al. [She13]**: Found numerous PROMISE data quality issues - Repeated rows, illegal attributes, inconsistent formats - **Critical gap**: Never tested if quality issues decreased predictive power

Our Experiment: - Built **mutators** that injected increasing amounts of their quality issues into PROMISE defect datasets - **Startling result**: Performance curves remained **flat** despite increased quality problems - **Implication**: *"There is such a thing as too much care"* in data collection

Practical Impact: - Effective predictions possible from seemingly dirty data - Questions excessive data cleaning efforts in SE research - Balance needed: careful collection without over-engineering

Slide 14: Menzies's 7th & 8th Laws

Menzies's 7th Law:

"Bad learners can make good conclusions."
Supporting Evidence: - **Nair et al. [Nai17]**: CART trees built for multi-objective optimization - **Key finding**: Models that **predicted poorly** could still **rank solutions effectively** - Could be used to prune poor configurations and find better ones - **Implication**: Algorithms shouldn't aim for predictions but offer **weak hints** about project data

Menzies's 8th Law:

"Science has mud on the lens."
Supporting Evidence: - **Hyperparameter optimization** lessons **[Agr21]**, **[Tan16]**, **[Fu16]** on PROMISE data - Data mining conclusions **changeable in an afternoon** by grad student with sufficient CPU - **Critical Questions**: Are all conclusions brittle? How build scientific community on such basis? - **Where are stable conclusions** for building tomorrow's ideas? **Bayesian Approach Needed**: Address uncertainty quantification and robust foundations

Slide 15: Menzies's 9th Law & Simplicity Challenge

Menzies's 9th Law:

"Many hard SE problems, aren't."
Supporting Philosophy: - **Cohen's Straw Man Principle [Coh95]**: *"Supposedly sophisticated methods should be benchmarked against seemingly stupider ones"*
Personal Experience Pattern: - *"Whenever I checked a supposedly sophisticated method against a simpler one, there was always something useful in the simpler"* - *"More often than not, a year later, I have switched to the simpler approach"* **[Agr21]**, **[Tan16]**, **[Fu16]**
Important Caveat: - **Not all SE problems can be simplified** - Generation tools probably need LLM complexities - Safety-critical software certification remains complex - *"Just because some tasks are hard, does not mean all tasks are hard"*
Challenge to Community: *"Have we really checked what is really complex and what is really very simple?"*
Current Focus: Minimal data approaches - landscape analysis **[Che19]**, **[Lus24]**, surrogate learning **[Nai20]**, active learning **[Kra15]**, **[Yu18]**

Slide 16: Contemporary Challenges & Solutions

PROMISE Revival Strategy (**Gema Rodríguez-Pérez**): - Data sharing now expected for almost all SE papers - PROMISE must differentiate: accept higher quality datasets - Focus on enhancing current data space, conducting quality evaluations

Steffen Herbold's Caution: - Early PROMISE: Collections of metrics (not raw data) - MSR shift: Raw data + fast tools (e.g., PyDriller, GHtorrent) - **Risk**: *"Little curation, little validation, often purely heuristic data collection without quality checks"* **[Her22]**

Modern Data Access: 1100+ recent Github projects **[Xia22]**, CommitGuru **[Ros15]**

Slide 17: Current “Hot” Research Directions

Contemporary Approaches: - **DeepLineDP** (**Pornprasit et al. [Por23]**): Deep learning for line-level defect prediction (TSE Best Paper 2023) - **Model interpretability**: Growing research focus **[Tan21]** - **Multi-objective optimization**: Hyperparameter selection **[Xia22]**, unfairness reduction **[Cha20]**, **[Alv23]**

CPU-Intensive Algorithms: - MaxWalkSat **[Men09]** - Simulated annealing **[Men02]**, **[Men07]**
- Genetic algorithms

Minimal Data Approaches: - How much can be achieved with as little data as possible? - Suspicion of “large number of good quality labels” assumption

Slide 18: Transfer Learning Surprises

Cross-Domain Success [Tur09]: - **Turkish white goods** $\rightarrow$ **NASA systems** error prediction - Expected: Complex multi-dimensional transforms - **Reality**: Simple nearest neighboring between test and training data

Implication: *"Many distinctions made about software are spurious and need to be revisited"*

Power Laws & Repeated Structures: - **Lin & Whitehead [Lin15]**: Fine-grained code changes follow power laws - **Hindle et al. [Hin12]**: Software naturalness - large repeated structures - **Result**: Thousands of rows modeled with few dozen samples **[Men08]**

Slide 19: Key Takeaways & Community Call-to-Action

Lessons Learned: 1. **Open science communities** can be formed by publishing baseline + data + scripts 2. **Reproducible research** drives field advancement when embraced collectively 3. **Simple solutions** often outperform sophisticated ones 4. **Data quality** matters less than expected for predictive tasks 5. **Transfer learning** works across surprisingly diverse domains

Call-to-Action: - *"Have we really checked what is really complex and what is really very simple?"* - Challenge assumptions about problem complexity - Benchmark sophisticated methods against simpler alternatives - Focus on stable, reproducible conclusions

Slide 20: Key References (Part 1: A-K)

[Agr18]: A. Agrawal and T. Menzies, "Is better data better than better data miners?: On the benefits of tuning smote for defect prediction," in *Proc. IST*, ACM, 2018, pp. 1050–1061. **[Agr21]:** A. Agrawal *et al.*, "How to"DODGE" complex software analytics?" *IEEE Trans. Softw. Eng.*, vol. 47, no. 10, pp. 2182–2194, Oct. 2021. **[Alv23]:** L. Alvarez and T. Menzies, "Don't lie to me: Avoiding malicious explanations with STEALTH," *IEEE Softw.*, vol. 40, no. 3, pp. 43–53, May/Jun. 2023. **[Cha20]:** J. Chakraborty *et al.*, "Fairway: A way to build fair ML software," in *Proc. FSE*, 2020, pp. 654–665. **[Che19]:** J. Chen *et al.*, " 'Sampling' as a baseline optimizer for search-based software engineering," *IEEE Trans. Softw. Eng.*, vol. 45, no. 6, pp. 597–614, Jun. 2019. **[Coh95]:** P. R. Cohen, *Empirical Methods for Artificial Intelligence*, Cambridge, MA: MIT Press, 1995. **[Dev15]:** P. Devanbu, "Foreword," in *Sharing Data and Models in Software Engineering*, T. Menzies *et al.*, Eds. San Mateo, CA: Morgan Kaufmann, 2015, pp. vii–viii. **[Fu16]:** W. Fu, T. Menzies, and X. Shen, "Tuning for software analytics: Is it really necessary?" *Inform. Softw. Technol.*, vol. 76, pp. 135–146, 2016. **[Gon23]:** J. M. Gonzalez-Barahona and G. Robles, "Revisiting the reproducibility of empirical software engineering studies based on data retrieved from development repositories," *Inf. Softw. Technol.*, vol. 164, 2023, Art. no. 107318. **[Gri22]:** L. Grinsztajn, E. Oyallon, and G. Varoquaux, "Why do tree-based models still outperform deep learning on typical tabular data?" in *Proc. NeurIPS*, 2022, pp. 507–520. **[Ham09]:** M. Hamill and K. Goseva-Popstojanova, "Common trends in software fault and failure data," *IEEE Trans. Softw. Eng.*, vol. 35, no. 4, pp. 484–496, Jul./Aug. 2009. **[Has08]:** A. E. Hassan, "The road ahead for mining software repositories," *Frontiers Softw. Maintenance*, pp. 48–57,

Slide 21: Key References (Part 2: K-Z)

[Kim15]: M. Kim *et al.*, "REMI: Defect prediction for efficient api testing," in *Proc. FSE*, ACM, 2015, pp. 990–993. **[Kri19]:** R. Krishna and T. Menzies, "Bellwethers: A baseline method for transfer learning," *IEEE Trans. Softw. Eng.*, vol. 45, no. 11, pp. 1081–1105, Nov. 2019. **[Men07]:** T. Menzies, J. Greenwald, and A. Frank, "Data mining static code attributes to learn defect predictors," *IEEE Trans. Softw. Eng.*, vol. 33, no. 1, pp. 2–13, Jan. 2007. **[Men24]:** T. Menzies, "A brief note, with thanks, on the contributions of guenther ruhe," *Inf. Softw. Technol.*, vol. 173, 2024, Art. no. 107486. **[Mis11]:** A. T. Misirli, A. Bener, and R. Kale, "AI-based software defect predictors: Applications and benefits in a case study," *AI Mag.*, vol. 32, no. 2, pp. 57–68, 2011. **[Nai17]:** V. Nair *et al.*, "Using bad learners to find good configurations," in *Proc. 11th Joint Meeting FSE*, ACM, 2017, pp. 257–267. **[Nam18]:** J. Nam *et al.*, "Heterogeneous defect prediction," *IEEE Trans. Softw. Eng.*, vol. 44, no. 9, pp. 874–896, Sep. 2018. **[Ost04]:** T. J. Ostrand, E. J. Weyuker, and R. M. Bell, "Where the bugs are," *ACM SIGSOFT Softw. Eng. Notes*, vol. 29, no. 4, pp. 86–96, 2004. **[Por23]:** C. Pornprasit and C. K. Tantithamthavorn, "DeepLineDP: Towards a deep learning approach for line-level defect prediction," *IEEE Trans. Softw. Eng.*, vol. 49, no. 1, pp. 84–98, Jan. 2023. **[Rah14]:** F. Rahman *et al.*, "Comparing static bug finders and statistical prediction," in *Proc. ICSE*, ACM, 2014, pp. 424–434. **[Rob10]:** G. Robles, "Replicating MSR: A study of the potential replicability of papers published in the mining software repositories proceedings," in *7th IEEE Work. Conf. Mining Softw. Repositories (MSR)*, IEEE Press, 2010, pp. 171–180. **[Ros15]:** C. Rosen, B. Grawi, and E. Shihab, "Commit guru: Analytics and risk prediction of software commits," in *Proc. ESEC/FSE*, 2015, pp. 966–969. **[She13]:** M. Shepperd *et al.*, "Data