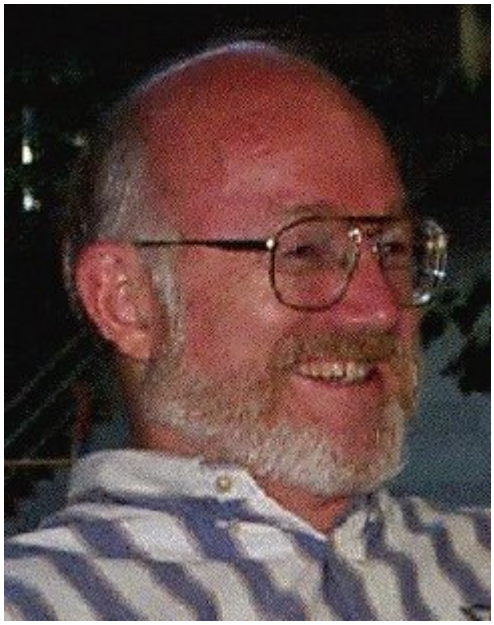
Software repositories have provided a rich source for retrospective empirical studies ranging from studying faults to studying evolution phenomena. As such they provide extensive primary data about software systems and software development processes to be characterized, classified and analyzed. Frustratingly, software repositories tend to be used mostly by researchers but seldomly by developers or project managers. For the latter, repositories could provide significant support for continuous process and product improvement. For example, even richer data about faults could be supplied at the time of version deposit and used to detect fault-prone activities or fault-inducing types of changes.

For both researchers and practitioners there are still innovative uses of software repositories untapped: repositories have been used extensively for *retrospective* studies but used very little for *prospective* studies (i.e., in experiments rather than just observational and relational studies). Change and version management repositories provide largely untapped resources for rigorously evaluating various software development techniques, methods, processes and tools. For example, they provide rich data with which to evaluate analysis and testing techniques and tools. For each component of a system we can build evolution and fault profiles, characterizing both the kinds of changes as well as the kinds of faults generated by these different changes. Given this change and fault data, we can rigorously evaluate fault detection and prediction techniques and tools to determine for which kinds of changes and faults they are the most effective. Given these rigorous evaluation approaches we can then effectively and fruitfully compare different analysis and testing approaches to measure how well they perform in the contexts most important for different types of projects.

This experimental approach side-steps a number of validity problems with techniques currently used in such analysis and testing evaluations, such as fault seeding (usually performed with little or no justification for the frequency and the types of faults seeded). With these repositories we avoid those problems. We can then focus on the primary problem of building benchmark sets of data covering a variety of different domains as well as a variety of different tradeoffs in non-functional requirements.

---

Prof. Dewayne E Perry is the Motorola Regents Chair in Software Engineering at The University of Texas at Austin. Prior to UT, he spent half his career as a practicing software engineer and half as a researcher in software engineering at Bell Laboratories in Murray Hill NJ.



not a new pic – but apparently the only one i have online ☺