FROM THE EDITOR

Data, Data Everywhere…

Forrest Shull

THE PATH FROM RESEARCH to practice is never an easy one. Many new methods and technologies that show great promise in the research world end up either never being tried in widespread practice or are tried but yield disappointing results. As an applied researcher, better managing tech transition is a topic near and dear to my heart, so I was pleased to participate in a panel discussion at this year’s International Conference on Software Engineering (ICSE) on early work in this area a decade ago. It has also had success in industry contexts, with a useful set of examples provided in our twin special issues last year: “Software Analytics: So What?,” July/August 2013, and “The Many Faces of Software Analytics,” September/October 2013.

It’s now transitioning to a new phase in which analytics research is continuing, but implementation is occurring in an increasing number of organizations. Given this context, the challenge set by the panel organizers was to share experiences with the application of software analytics technologies: “While there may be no single best ‘right’ way to analyze software data, there are many wrong ways. As data techniques mature, we need to move to a new era where data scientists understand and share the strengths and drawbacks of the many methods that might be deployed in industry.” Perhaps not surprisingly, more of the discussion focused on experiences with getting an analytics program in place in organizations, rather than on the good and the bad of any given technique.

Intentional versus Opportunistic

Much discussion at the ICSE panel focused not on the analytical or data-mining techniques themselves but on the underlying data on which they operate, which is entirely appropriate. Having experienced the many data quality problems that come along with any real-world dataset (a problem that’s well documented in software engineering datasets), I worry sometimes that the sophistication of our analysis technologies is far outstripping the level of confidence anyone should actually have in the data itself. (Elsewhere at ICSE, I delivered a paper giving an example of a situation in which, for reasonable levels of inaccuracy in a commercial dataset, very wrong conclusions can be drawn from analysis.) Thus, applying analytics to the wrong problem can be not only expensive but counterproductive.

Many of the issues the panel discussed were related to one underlying question: Does putting an ef-

I worry sometimes about the sophistication of our analysis technologies.
ffective analytics program in place require investments in new data and new data sources?

By way of context, let’s not forget that one of the primary causes of failed or non-useful software measurement programs is when organizations spend time and effort collecting measures simply because they seem to be useful, rather than because they’re tied to organizational goals. Although this seems to be an obvious point, enough software practitioners have stories of encountering “write-only” metrics databases in their organizations that the mistake must still be fairly common.

Goal-directed methods such as the Goal–Question–Metric (GQM) approach have long been successful at addressing this problem for measurement programs, by ensuring that measurement goals are articulated and that connections can be made between those goals and the actual metrics being collected. But, do newer data-mining technologies and the ability to more easily extract data from new data sources (for example, instrumenting the code base in the version control repository) avoid the problem and make it possible to find new value in old datasets?

The discussants broke down into two camps: one strongly advocated that analysts need to be intentional and to work on what’s useful, not just what is convenient to collect, and the other argued for reflecting on what we have in hand, which can pay dividends by maturing the analytic technologies and infrastructure.

Although it led to several good points in the discussion, I feel this distinction sets up a false dichotomy. Both intentional and opportunistic modes are necessary at different points in time—the trick is knowing how and when to switch between them.

Addressing Stakeholder Expectations
An approach often recommended in tech transfer discussions is to “use what you have,” exploiting resources already in hand to make the case from early successes that more resources can move the effort into a more desirable or impactful direction. For software analytics, this means starting with data already in existence (or that can be generated relatively inexpensively, such as from development repositories) to provide insight to stakeholders. Those stakeholders, once they see the benefit of that additional insight, will presumably be more likely to make resources available for more data collection to answer additional questions.

But there are dangers to this approach. In my experience, managers...
are just as likely to be turned off by what they see as “trivial” initial findings that don’t address their particular burning questions and decline further funding based on that sense of disappointment. From the number of nodding heads I saw around the room when I raised this issue, I’m far from alone in having seen this.

It’s hard to generalize, but in my experience, the data that’s usually easiest to come by is in the form of code and defect metrics, which let the analyst create defect prediction models that can help focus V&V effort where it’s most needed. However, it’s important to remember that the usefulness of those models depends a great deal on how well the defect data is recorded. If effort data at a useful level of fidelity is available in the environment, then basic effort prediction models also become possible. In most cases, the factors predicting effort and defect-proneness are going to be characteristics of the code itself, which is useful, especially because values extracted directly from the code are likely to be more accurate and reliable measures around.

In the panel discussion, Curtis called out a frequent mismatch between academic theories and the variables that are really important to industry, and I heartily second that. Certainly in the work I conduct, managers are looking for something more than defect and effort models—namely, the ability to connect how the team is operating with its outcome. Decisions about how teams organize themselves and their work are the ones that managers and organizations have control over and can usefully influence. But the data to address these issues isn’t always collected and tends not to be automatically extractable. Something extra is needed to answer these types of questions, for example:

- Process, policy, and team considerations. These factors describe not just what kind of code the team ended up with but the approach by which they produced that code. Many empirical studies have demonstrated that various decisions in this category have a strong impact on team performance. These values either need to be recorded manually in the database or perhaps can be extracted by other metrics that cover process conformance issues.

- Execution logging. Measures extracted from these logs provide detailed insight into specific code behaviors. Collecting such data typically requires inserting instrumentation in the code itself.

- Architectural rules. These measures examine how the code was structured, so that structural decisions can be compared to the software’s resulting fitness for a task. Useful metrics go beyond generic heuristics such as “avoid high coupling” to take a more detailed look at, for example, conformance to domain-specific reference architectures. Collecting such data clearly requires having a sense of the reference architecture itself as well as being able to measure architectural conformance.

My examples of additional data might not match stakeholder interests in every environment, but the point is to illustrate the need for analysts to be sufficiently tied into team and organizational priorities to be able to recognize different stakeholder groups and have a sense of their interests and goals. From there, of course, it’s necessary to connect those goals with appropriate measures, whether they’re available today or not. For this, GQM and similar approaches, despite being mature (some would say ancient) technologies in a fast-moving field such as ours, can still be useful.

By having this mapping of goals to metrics, the analyst can get a sense of which of the metrics available today (if any) will answer any stakeholder’s questions of interest. If some exist, then those become the obvious candidates for demonstrating early successes that can be used to buy more data collection later. The opposite is also true: such knowledge helps us recognize when “you can’t get there from here,” and no amount of data analysis is going to answer the questions that stakeholders care about. In this case, rest assured that this measurement program is in for a rough time unless re-
sources are available to start collecting new data.

Goal-directed metrics also help the analyst by directing effort away from datasets that won’t lead to important stakeholder answers. In my experience, there’s too often a danger in spending significant effort on repurposing existing datasets for answering new questions, the assumption being that they were expensive, so they must be valuable. However, there’s no guarantee that data collected in a particular environment and for a particular purpose will have interesting insights for newer problems. A goal-directed approach allows understanding for which datasets such effort will be well-spent.

Goal-directed metrics help analysts understand when they’re coming close to exhausting the useful analyses on a given dataset. Analysts need to be careful about knowing where the line is between working in an initial dataset and getting some useful insights, and overextending themselves and tackling goals that can’t be addressed. When the analyst reaches that line, she needs to be able to recognize it and start marshaling existing successes to make an argument for collecting more and/or better data. That’s when the effort would be well-spent on crossing the line and moving smoothly into a new phase of the work.

References


Selected CS articles and columns are also available for free at http://ComputingNow.computer.org.

IEEE SOFTWARE CALL FOR PAPERS

Special Issue on Trends in Systems and Software Variability

Submission deadline: 1 October 2014 • Publication: May/June 2015

Cyber-physical systems (CPS) are systems whose computational elements collaborate to control physical entities. A wide variety of CPS can be found in areas such as aerospace, automotive, energy, healthcare, manufacturing, transportation, entertainment, robots, and consumer appliances. The challenge for building CPS product families relies on new methods able to address more efficiently, dynamically, and intelligently than ever before the context information needed by self-adaptive and CPS systems.

The IEEE Software special issue on trends in systems and software variability will present a variety of techniques, tools, and approaches for software variability that support the challenge of adaptation and awareness of CPS systems and their impact on recent software product-line development approaches. We invite contributions related but not limited to the following:

• architectures for cyber-physical systems that effectively support dynamic variability;
• modeling techniques and viewpoints directed to modeling the contextual and physical properties of CPS;
• new software product-line development methods supporting runtime variability models and their impact on the SPL development life cycle;
• case studies on the impact on software evolution of dynamic variability models;
• new variability realization, configuration, and deployment methods;
• tools and models for managing both static and dynamic variability in CPS using context information;
• industry cases and experience reports managing dynamic variability for different types of CPS;
• software variability techniques for self-adaptive and cyber-physical systems;
• self-adaptive and CPS systems viewed as a dynamic software product lines (DSPLs); and
• integration of runtime variability solutions into current SPL/DSPL practice.

Questions?

For more information about the focus, contact the guest editors:

• Jan Bosch, jan@janbosch.com
• Rafael Capilla, rafael.capilla@urjc.es
• Rich Hilliard, richh@mit.edu

Full author guidelines: www.computer.org/software/author.htm
Submission details: software@computer.org
Submit an article: https://mc.manuscriptcentral.com/sw-cs