Appendix to “The Health of Software Engineering Research”

David Lo
School of Information Systems
Singapore Management University
Singapore
davidlo@smu.edu.sg

Nachiappan Nagappan and Thomas Zimmermann
Research in Software Engineering (RiSE)
Microsoft Research
Redmond, USA
{nachin,tzimmer}@microsoft.com

Abstract - This appendix contains the full survey that we conducted in Microsoft (Appendix A) and the top-5 research ideas that were rated by developers, testers, and program managers, respectively (Appendix B).

APPENDIX A: FULL SURVEY

State of the Art in Software Engineering Research

1. Which best describes your primary work area? (required) *This question is required.

- Development
- Test
- PM
- Other Please enter an 'other' value for this selection.

2. Which of the following best describes your role? (required) *This question is required.

- Individual Contributor
- Lead
- Architect
- Manager
- Executive
- Other Please enter an 'other' value for this selection.

3. How many years have you worked in the software industry? (decimals okay) (required) *This question is required.

4. Please answer the following questions about demographics. (required) *This question is required.

Did you major in computer science or a related field (such as computer engineering, information systems)?	Yes	No
Do you have an advanced degree (MSc, PhD, etc.)?	Yes	No

5. In your opinion, how important are the following pieces of research? Please respond to as many as possible. (at least 1 response required) *This question is required.

Technique to identify files that contain a bug from the description of that bug (in a bug report).	○ Essential	○ Worthwhile	○ Unimportant	○ Unwise	○ I don't understand
Empirical study examining factors affecting the cost and effectiveness of test suite augmentation techniques	○ Essential	○ Worthwhile	○ Unimportant	○ Unwise	○ I don't understand
Empirical study of a community portal used by a closed source software project to characterize artifacts shared there and highlighting benefits and possible shortcomings	○ Essential	○ Worthwhile	○ Unimportant	○ Unwise	○ I don't understand
Mutation based test data generation approach that is capable of killing both first and higher order mutants	○ Essential	○ Worthwhile	○ Unimportant	○ Unwise	○ I don't understand
Defect prediction technique to identify defects that break pre-existing functionality (breakage defects) and defects in files that had relatively few pre-release changes (surprise defects).	○ Essential	○ Worthwhile	○ Unimportant	○ Unwise	○ I don't understand
A concurrency model that is better than existing model in terms of not extraneously introducing behaviors infeasible in the actual system, not extraneously excluding actual behaviors, and isolating the challenging features for analyses to focus on	○ Essential	○ Worthwhile	○ Unimportant	○ Unwise	○ I don't understand
An empirical study to test how effective are techniques that are able to help migrate client code between library versions with incompatible APIs work in practice	○ Essential	○ Worthwhile	○ Unimportant	○ Unwise	○ I don't understand
A technique to recover missing links between bug reports and bug fixing commits. It takes into account not only textual features but also source code features of the changed code corresponding to the commit logs.	○ Essential	○ Worthwhile	○ Unimportant	○ Unwise	○ I don't understand
An empirical study on the feasibility of CIT for 5- and 6-way feature interactions that takes into account constraints and test case prioritization	○ Essential	○ Worthwhile	○ Unimportant	○ Unwise	○ I don't understand
Technique to scale up concolic testing based on interpolation, that greatly mitigates path-explosion by subsuming paths that can be guaranteed to not hit a bug	○ Essential	○ Worthwhile	○ Unimportant	○ Unwise	○ I don't understand
A change interaction is when where several program changes are found to affect the result of a program statement via program dependencies. The paper proposes an approach to generate test cases which witness change interaction errors.	○ Essential	○ Worthwhile	○ Unimportant	○ Unwise	○ I don't understand
A reachability question is a search across feasible paths through a program for target statements matching search criteria. The paper reports the result of an empirical study that reported that reachability questions are common and often time consuming to answer.	○ Essential	○ Worthwhile	○ Unimportant	○ Unwise	○ I don't understand
A verification approach for parameterized system (i.e., a parametric infinite family of systems) which tries to improve efficiency by limiting search to a maximum path length	○ Essential	○ Worthwhile	○ Unimportant	○ Unwise	○ I don't understand
Technique to find links between e-mails and the software artifacts they discuss	○ Essential	○ Worthwhile	○ Unimportant	○ Unwise	○ I don't understand
A new approach to controlling concurrency which eases the task of implementing sophisticated locking schemes and provide static checks to automatically detect many data races. Views consist of view declarations that describe which views of an object may be simultaneously held by different threads, which object fields may be accessed through a given view, and which methods can be called through a given view.	○ Essential	○ Worthwhile	○ Unimportant	○ Unwise	○ I don't understand
Technique to dynamically infer likely deterministic specifications for parallel programs given a set of inputs and schedules	○ Essential	○ Worthwhile	○ Unimportant	○ Unwise	○ I don't understand

An automated tool that assists programmers with refactoring synchronized blocks into ReentrantLocks and ReadWriteLocks, to make exploring the performance tradeoffs among these constructs easier.	○ Essential	○ Worthwhile	○ Unimportant	○ Unwise	○ I don't understand
Empirical study that quantifies how the choice of programming language impacts software quality and developer productivity	○ Essential	○ Worthwhile	○ Unimportant	○ Unwise	○ I don't understand
Empirical study that investigates if social network analysis metrics computed based on information stored in software development artifacts represent actual socio-technical relationships. This is done by examining if developer networks can be matched with developer perceptions.	○ Essential	○ Worthwhile	○ Unimportant	○ Unwise	○ I don't understand
A tool that allows developers to reap the benefits of both static and dynamic typing, throughout the development process, and without the burden of manually separating their program into statically and dynamically-typed parts.	○ Essential	○ Worthwhile	○ Unimportant	○ Unwise	○ I don't understand
An algorithm to fix a bad configuration by generating range fixes (i.e., the options to change and the ranges of values for these options)	○ Essential	○ Worthwhile	○ Unimportant	○ Unwise	○ I don't understand
Empirical study about how developers practice program comprehension under time and project pressure, and which methods and tools proposed by researchers are used in industry.	○ Essential	○ Worthwhile	○ Unimportant	○ Unwise	○ I don't understand
A technique to classify email lines into five categories (i.e., text, junk, code, patch, and stack trace)	○ Essential	○ Worthwhile	○ Unimportant	○ Unwise	○ I don't understand
An automated technique to maintain CSS rules by identifying obsolete CSS rules, DOM elements that some rules affect, and impact if some CSS rules are removed.	○ Essential	○ Worthwhile	○ Unimportant	○ Unwise	○ I don't understand
A technique that takes a textual change request (e.g., a bug report), a single snapshot (release) of source code, and an initial source code entity that is impacted by the change, and outputs the other source code entities.	○ Essential	○ Worthwhile	○ Unimportant	○ Unwise	○ I don't understand
A tool for making software architecture consistent with implementations during software development. The tool supports deep separation of generated and non-generated code, an architecture change model, architecture-based code regeneration, and architecture change notification.	○ Essential	○ Worthwhile	○ Unimportant	○ Unwise	○ I don't understand
Approach to automatically extract FAQs from sources of software development discussion, such as mailing lists and Internet forums	○ Essential	○ Worthwhile	○ Unimportant	○ Unwise	○ I don't understand
Empirical study on how the increased transparency found on GitHub influences developers' testing behaviors.	○ Essential	○ Worthwhile	○ Unimportant	○ Unwise	○ I don't understand
Automatic unit test generation for programs written in C/C++. The proposed approach improves the coverage obtained by feedback-directed random test generation methods, by utilizing concolic execution on the generated test drivers and by employing non-linear solvers for numeric computations.	○ Essential	○ Worthwhile	○ Unimportant	○ Unwise	○ I don't understand
A technique that extends Alloy to presents a set of scenarios from a specification. Different from Alloy, Aluminium ensures that the generated scenarios are minimal and allows users to augment a scenario by providing new tuples.	○ Essential	○ Worthwhile	○ Unimportant	○ Unwise	○ I don't understand
An automatic testing technique to reveal subclasses that cannot safely substitute their superclasses.	○ Essential	○ Worthwhile	○ Unimportant	○ Unwise	○ I don't understand
Empirical study on the impact of issue report misclassification (feature request -> bug, bug -> feature request) to bug prediction.	○ Essential	○ Worthwhile	○ Unimportant	○ Unwise	○ I don't understand

A domain specific language and a distributed computing infrastructure to allow users to query and get information from a large number of software repositories.	○ Essential	○ Worthwhile	○ Unimportant	○ Unwise	○ I don't understand
A multi-objective decision support approach to help balance project risks and duration against overtime, so that software engineers can better plan overtime.	○ Essential	○ Worthwhile	○ Unimportant	○ Unwise	○ I don't understand
A technique that creates a context-aware edit script from two or more examples, and uses the script to automatically identify edit locations and transform the code.	○ Essential	○ Worthwhile	○ Unimportant	○ Unwise	○ I don't understand
Automated technique for reliability estimation that combines simulation, invariant inference and probabilistic model checking	○ Essential	○ Worthwhile	○ Unimportant	○ Unwise	○ I don't understand
A generic, extensible framework which brings sketching functionality to Eclipse Graphical Editing Framework based diagram editor. Sketch features can be dynamically injected and used without writing a single line of code.	○ Essential	○ Worthwhile	○ Unimportant	○ Unwise	○ I don't understand
An approach to automatically re-write web pages so that these pages can be displayed with less energy consumption.	○ Essential	○ Worthwhile	○ Unimportant	○ Unwise	○ I don't understand
Tiered framework for combining behaviour models that are used to generate operational strategies for adaptive systems to allow for graceful degradation when some assumptions are broken, and progressive enhancement when those assumptions are satisfied or restored.	○ Essential	○ Worthwhile	○ Unimportant	○ Unwise	○ I don't understand
An approach to help developers deal with errors due to failure in recognizing feature dependencies when maintaining software product line. The approach infers interfaces to features (a set of provides and requires clauses to describe dependencies among features) that are relevant to a code change task.	○ Essential	○ Worthwhile	○ Unimportant	○ Unwise	○ I don't understand

6. On the previous page, you selected the following research idea as "Unwise":

"Technique to identify files that contain a bug from the description of that bug (in a bug report)."

To help us better understand your response, could you please explain why.

APPENDIX B: TOP RESEARCH IDEAS PER ROLE

TABLE I. TOP RESEARCH IDEAS (DEVELOPERS)

Paper Summary	Total	E-Score	EW-Score	U-Score
A new technique that not only detects leaks, but also points developers to the locations where the underlying errors may be fixed.	13	0.69	1.00	0.00
An approach to help developers identify and resolve conflicts early during collaborative software development, before those conflicts become severe and before relevant changes fade away in the developers' memories.	22	0.68	0.82	0.00
Technique that clusters callstack traces to help performance analysts effectively discover highly impactful performance bugs (e.g., bugs impacting many users with long response delay).	15	0.67	1.00	0.00
Debugging tool that uses objects as key abstractions to support debugging operations. Instead of setting breakpoints that refer to source code, one sets breakpoints with reference to a particular object.	13	0.62	0.92	0.08
Automatic generation of efficient multithreaded random tests that effectively trigger concurrency bugs.	15	0.60	0.93	0.07

TABLE II. TOP RESEARCH IDEAS (TESTERS)

Paper Summary	Total	E-Score	EW-Score	U-Score
A technique to monitor if a system fulfils its requirements expressed as probabilistic properties (e.g., performance, reliability, safety, and availability requirements) at runtime.	6	0.83	1.00	0.00
A methodology to drive the adaptation of a service-oriented system to meet QoS requirements of several concurrent users in its volatile operating environment.	7	0.71	1.00	0.00
A technique to engineer applications with a self-healing layer for service-oriented systems that dynamically reveals and fixes interoperability problems.	7	0.71	1.00	0.00
A technique to recover missing links between bug reports and bug fixing commits. It takes into account not only textual features but also source code features of the changed code corresponding to the commit logs.	7	0.71	0.86	0.00
A semi-automated lightweight code analysis tool to help create an explicit, end-to-end argument, based on concrete evidence, that a system satisfies a critical property. It generates a list of side conditions that correspond to assumptions to be discharged about the code and the environment in which it executes.	6	0.67	0.83	0.00

TABLE III. TOP RESEARCH IDEAS (PROGRAM MANAGERS).
LAST 5 RESEARCH IDEAS RECEIVE THE SAME SCORES AND THUS WE LIST ALL OF THEM.

Paper Summary	Total	E-Score	EW-Score	U-Score
Empirical study on how agile teams self-organize themselves by identifying roles that developers play.	5	0.80	0.80	0.00
Debugging tool that uses objects as key abstractions to support debugging operations. Instead of setting breakpoints that refer to source code, one sets breakpoints with reference to a particular object.	5	0.80	0.80	0.00
Combination of highly configurable project, team and contributor dashboards along with individual event feeds to help developers accomplish extensive awareness (i.e., awareness of various different aspects ranging from overall project status and process bottlenecks to current tasks and incoming artifacts)	10	0.70	1.00	0.00
A technique to find null-pointer dereferences as a target for finding bugs in concurrent programs using testing. It observes an execution of a concurrent program under test and predicts alternate interleavings that are likely to cause null-pointer dereferences.	6	0.67	1.00	0.00
Technique to make runtime reconfiguration of distributed systems in response to changing environments and evolving requirements safe and being done in a low-disruptive way through the concept of version consistency of distributed transactions	6	0.67	1.00	0.00
Technique to merge N models into one model. This technique is relevant when one would like to merge a set of related products into a product line or consolidating model views of multiple stakeholders.	6	0.67	1.00	0.00
Automated testing of JavaScript code in isolation from the server code and database contents by automatic inference of formal server interface description.	6	0.67	1.00	0.00
Symbolic analysis algorithm for buffer overflow detection that scale to millions of lines of code (MLOC) and can effectively handle loops and complex program structures.	6	0.67	1.00	0.00