Test Benchmarks – what is the question?

Per Runeson, Mats Skoglund, Emelie Engström
Software Engineering Research Group
Lund University, Sweden, per.runeson@cs.lth.se

1. Introduction

"I am taller than you".
"My dad is stronger than yours".

Kids do not grow very old until they begin benchmarking. They benchmark to impress on their mates and to give themselves a position in the group. But what does the benchmark mean when the child wants to reach the cookies on the top shelf of the larder? Although being the tallest, he might not be tall enough to reach it anyhow, and his father might not be there to lift him up. And if he was, he would not allow his kids to take those cookies anyhow.

In the automotive press, there are lots of benchmarks. Acceleration from 0 to 100 km/h or 0 to 60 mph is a frequently used benchmark. But how often do you accelerate as fast as possible from 0 to 100 km/h? Similarly is the power and the torque of the engine benchmarked, but rarely it is noticed whether the power is delivered at revs which are useful in my daily driving or at top revs. And I rarely use more than some 25 kW to run my car, although I have access to hundreds. Furthermore, the EuroNCAP[1] and NTSB[2] do benchmarks on crash resistance and rate car models according to their resistance to the benchmark tests.

When software test researchers benchmark, they use some well specified sets of programs and apply and evaluate their test techniques. The programs are mostly selected based on availability, and sometimes also made available for others; see e.g. the Software-artifact Infrastructure Repository[3], although this particular example does not have the ambition of constituting benchmarks [2]. However, before judging whether the benchmarks are useful or not, we should consider what it should be used for. What is the question we want to answer with a benchmark?

[1] The European New Car Assessment Programme http://www.euroncap.com
[2] The National Transportation Safety Board http://www.ntsb.gov
[3] http://sir.unl.edu

2. Uses for test benchmarks

From a practitioner's point of view, the benchmark must focus on the feasibility for the use of the benchmarked techniques and tools in a specific context. "Is this test technique more efficient than the other for my software system?" This is however not a question that can be answered by a single benchmark.

From a researchers' point of view, we have learned that empirical evaluation is good research while blunt assertion is not [1]. Hence, we must have some context in which we may evaluate our techniques and tools. And there is always an issue of relevance; can this be used and useful in software industry?

The benchmarking question involves many degrees of freedom that may impact on the outcome. It is not only the program under test, but its test cases, the defects, its development environment, its development process etc. Hence, the issue of benchmarking is very complex and we find it too ambitious to search benchmarks that mirror all this variation,; rather some specific aspects may be studied at a time.

In the automotive domain, where benchmarking frequently takes place, the specific benchmarks may not be of highest relevance, but they are indicators that represent some attributes of the car that a customer may give priority or not. I would choose a car making 0-100 km/h in 5 seconds if I like fast driving (and I can afford it) while for a family car, 0-100 km/h in 10 seconds is sufficient to keep up the daily traffic pace. For crash resistance, I may prefer a five star Euro NCAP rated car before a three star, even though I do not intend to crash it from 64 km/h (40 mph) into a concrete barrier. In this area, the benchmarking procedures have forced car manufacturers to make more crash resistant cars in general in order to fulfill the customer's demands.

In the testing context, benchmarking may be used to indicate specific characteristics (like the acceleration) or be a driving force in a general improvement trend (like crash tests). One of the key issues in finding benchmarks is the representativeness of the benchmark as such. What does it mean in

practice that one technique is better than another for a given benchmark?

3. Representativeness

In order to generalize a result from a small set of subjects to a wider population, sampling is applied. For example in national polls or other surveys, a subset from the population are sampled, interviewed about their opinions and conclusions are drawn for the whole population [3]. The sample represents the whole population in a *statistical generalization*. The underlying principles are that the random variation among the subjects is captured in the sample within an acceptable error margin. This is the underlying principle for controlled experimentation.

In qualitative design research, like case studies, the selection is different. The case to be studied is selected to represent e.g. the typical or the special case [4]. The case cannot be generalized to a wider population through statistical analyses. Still one may learn from a specific case and apply the knowledge to another specific case. In case studies you apply *analytical generalization*. In analytical generalization, the case is characterized and compared to other cases to identify patterns which may indicate some general understanding drawn from the specific case.

The search for testing benchmarks may take either way: the *statistical* or the *analytical* approach. The former means defining a population of software programs, sampling from that population and selecting a representative subset which the test techniques may be applied to for evaluation. The statistical approach is desirable but impractical and must hence be excluded. The analytical approach is closer to what is already done, i.e. using a set of programs, and then generalize the results from the studies analytically.

The analytical approach may be supported by categorization scheme that guides the analytical generalization. Depending on the scope of the evaluated item, benchmarking may be very different, which is elaborated in the next chapter.

Refer to the car crash tests again. Sampling from all possible crashes and repeating a subset in the laboratory would enable calculating a risk factor for a certain car with a specified statistical significance, i.e. statistical generalization. The approach actually used is that some typical crash situations with frontal and side impact are repeated in the laboratory, i.e. analytical generalization.

4. Variation factors

In the effort for finding typical or special cases or subject programs to be used for benchmarking purposes, many variation factors must be considered. Variation factors may be regarding the program under test, its specifications, the test technique or tool, or the test process or the defects. Factors may be related to the product under test, the test process or the test resources. Below we list some, based on our experience from test research:

Process factors

- Does the technique require specification documents, e.g. UML diagrams?
- Programming language(s) – is the technique applicable to the programming languages used? What if the there are different languages? If source code is not accessible?
- How many and which type of changes are made between successive releases?
- What is the purpose of the test technique/tool? Test case selection? Test case prioritization?
- Is the technique deterministic, i.e. selects the same test cases independently of who applies it?
- Which types of test are within the scope? Unit test? System test? GUI tests?

Product factors

- Size and complexity – is the program large and complex enough to be relevant for the real world problem?
- System type – which type of system is it? Real-time systems vs. batch?
- Libraries – how is it dependent on code libraries and their changes?
- Test cases – what size are the test cases, and do they depend on each other?
- Test data – are they complex enough to be relevant for real world problems?
- Defects – are the numbers, types and distribution of defects relevant?

Resource factors

- Which skills and knowledge do the testers and test designers have?

These variation factors must be taken into account when defining test benchmark programs and processes.

5. Proposal

Based on the considerations above, we propose the following for test benchmarks:

1. Define categories for benchmarked methods to avoid comparing "apples with oranges", e.g. comparing safe test selection methods with unsafe.
2. Look upon benchmarks as selected cases, not representative samples, and interpret benchmarking results accordingly.
3. Define a characterization scheme to capture the relevant degrees of freedom that characterize a test environment.
4. Define not only a set of benchmarking programs, but also the corresponding test cases, defects, execution environment and test processes used.
5. Combine benchmarking results with case studies to analyze both a controlled environment and a real world environment where the interactions between the test technique and its environment can be studied as well.

In summary, the answer is not only a benchmark, but a benchmark in its context. Benchmarking is not aimed at statistical generalization, but analytical. The focus is on the typical or the special situation, not on the "average" situation.

References

[1] Shaw M What makes good research in software engineering? *International Journal on Software Tools for Technology Transfer* Springer 4(1):1-7, 2002

[2] Do H, Elbaum S, Rothermel G, Supporting Controlled Experimentation with Testing Techniques: An Infrastructure and its Potential Impact, *Empirical Software Engineering*, 10, 405–435, 2005

[3] Wohlin C, Höst M, Ohlsson MC, Regnell B, Runeson P, Wesslén A *Experimentation in Software Engineering - An Introduction*, Kluwer 2000

[4] Yin R K *Case Study Research. Design and Methods* (3rd edition) London: Sage, 2003