The (Im)maturity Level of Software Testing
Antonia Bertolino
Istituto di Scienze e Tecnologie dell’Informazione “A. Faedo” -
ISTI-CNR, via Moruzzi, 1
Pisa, Italy
+39 050 315 2914
antonia.bertolino@isti.cnr.it

ABSTRACT
A large gap exists between the state-of-the-art in software testing literature, and the state of software testing practice. Empirical research should (and could) play a first class role for bridging this gap. Empirical studies in software testing have focused mainly on the evaluation of techniques for test case selection. But effective selection of test cases by itself is not sufficient to warrant successful testing: we need also empirical studies to start collecting proven patterns that test practitioners can use to predictably solve software testing problems.

Categories and Subject Descriptors
D.2.5 [Testing and Debugging]

General Terms
Measurement, Reliability, Experimentation, Verification.

Keywords
Comparisons of Test techniques, Software Testing Maturity, Test Patterns

1. INTRODUCTION
A mature level of knowledge, formed by proven facts and undisputed statements, is what allows engineering disciplines to achieve predictable results [7]. Besides, mature engineering disciplines can rely on handbooks that describe successful solutions to known problems [13].

If we consider software testing in view of the previous statements, we would have to reach the inescapable conclusion that its current state of knowledge, as well as of practice, is far from mature. More precisely, our position here is that a large gap exists between the state of software testing knowledge in the academic community and among practitioners, and that, as a consequence, the current state of practice is still rudimentary and, notwithstanding the availability of advanced techniques and tools, remains largely based on tester’s skill, experience and intuition.

As software engineering researchers are now well aware, it is empirical research that paves the path from subjectivity to objectivity [7]. This short position paper contributes to the Workshop on Empirical Research in Software Testing WERST’2004 [15] by exposing some personal reflections on the current situation of affairs, and by launching some speculative ideas to further empirical research in software testing.

2. THE GAP BETWEEN RESEARCH AND PRACTICE
A large distance exists between “software testing” as it is depicted in the literature and as it is routinely practiced in industry (see for instance positions in [6]). As an example, if we mention “(code) coverage testing”, the academic researchers would naturally think of a spectrum of many different criteria among which to choose, variously based on the program control-flow or data-flow, as well as to various automated approaches proposed to facilitate test generation and to minimize compliant test suites. Besides, academic researchers would also be able to refer to several published studies, which compare the costs and the effectiveness of more or less thorough coverage criteria. But only a little part of these results has reached the software industry. By the practitioners “coverage testing” would be intended as a synonymous for “branch coverage” (if not even “statement coverage”). Moreover, they would employ commercial tools for dynamically monitoring branch coverage, but most likely generate additional test cases to cover yet unexercised branches manually.

A similar gap can be observed for functional testing. For years researchers have been developing systematic and automated approaches to derive from the formal or semi-formal specifications test cases that can check program conformance (this kind of testing is more trendily referred to today as “model-based testing”). Yet, ad hoc testing based on tester’s intuition remains a popular approach among practitioners. Some tentative experiences in using formalized and automated approaches show that model-based testing is still perceived as too difficult to apply [2], even in

1 Within some testing communities, the latter feature is felt more as a point in favor than as a problem. We do not deny the outmost value of the mentioned tester’s qualities. The point is they cannot be the major, or the only, instrument on which the testing discipline is based, simply because they cannot produce predictable performance. We should pursue instead objective, reproducible, testing methodologies and handbooks of use for both smart and dull testers.
those industrial contexts where a more advanced testing culture exists.

The negative consequences of this gap are clearly visible to everybody, with testing remaining one of the most expensive and effort-prone activities of the development process, and with delivered products of poor quality and low reliability. While some open problems are there to stay, there are certainly many techniques and tools which could be already applied with neat advantage to alleviate this situation.

Researchers should seriously question themselves about the reasons for this evident inefficiency to transfer their results to practice. In particular, empirical research can play a first-class role in helping advancing the status of practice. In fact, successful experiences of a proposed test technique with measured benefits could be the best way to capture the interest of managers. Who, anyway, after that also require mature processes and tools.

3. COMPARISONS OF TEST TECHNIQUES

In view of a proliferation of test techniques, several studies have been carried on, both analytical and empirical in kind, to compare the relative cost-effectiveness of the different approaches and provide advice for selecting among them.

3.1 Analytical Comparisons

Analytical studies are appealing on one side because they can produce more generally valid results, i.e., results which are not tied to a specific experimental context. Thus, starting from the seminal paper by Duran & Ntafos [3], a series of papers have explored the conditions under which it can be expected that a test technique will perform more effectively than another in detecting failures. On the other side, analytical studies remain so far quite academic in kind: they are highly useful to enlarge our knowledge behind testing techniques but provide little practical guidance in selecting a test technique. The reason is that the conclusions provided by such analytical comparisons are based on assumptions that are far beyond what one can reasonably expect to know or even hypothesize about a program under test. Most commonly, the assumptions require that the tester can make conjectures about the distributions of the failure-causing inputs within the input domain, with respect to the subdomains induced by the selected criterion. For instance, one of the most recent studies [11] compares the expected number of failures detected by different test techniques based on the ordering of the failure rates of their respective subdomains. While such a result is certainly interesting from a theoretical viewpoint, because it generalizes the hypotheses of most earlier works, further investigation is required before it can be applied in practice.

Perhaps what we need also here is empirical research to close the circle: analytical studies give us hints on which conditions make a test technique more convenient than another, and by means of empirical studies we then check when it is that such conditions are met for real programs, e.g., which observable parameters can allow us to foretell how the failure rates are distributed. Although still in short supply, some studies collecting failure data from industries now start to appear, as for instance [10], [12], thus testifying a growing awareness of the crucial importance of past data for improving future products quality. Such kind of empirical studies could also be used to put the conclusions from analytical comparisons within a practical context.

3.2 Empirical Comparisons

Empirical comparisons of the effectiveness of test criteria are closer to a practitioner’s mindset: a selected test technique is applied to a case study and measures from the observed experimental outcomes are taken. Indeed, directly observed results can sound more realistic and convincing than mathematical formulas built on top of theoretical conjectures. A recent survey of empirical comparisons is provided in reference [7].

However, one of the conclusions of the cited survey is that further empirical research in software testing is needed, and that much more replication has to be conducted before general results can be stated [7].

Easier to say than to do, empirical comparisons of test techniques are very difficult and expensive for a series of obvious reasons. And, after an experiment is concluded, one could always find several objections against both the generality of the observed results, and the oversimplification of the experiment setting, which makes the job of the empirical researcher quite frustrating.

3.3 What does “better” means?

In the above discussed analytical and empirical studies, different techniques are compared to find out which one performs “better” under which conditions.

Clearly, any result on the above question can never be unconditionally valid: what makes a test case “better” than another does not have a unique answer, but it changes depending on the context, on the specific application, and on the testing goals. Test suites that are “good” for one test purpose, could not be as good for another.

The most common interpretation in the above studies for “better” would be “able to detect more failures”; but even so precision would require to further specify what kind of failures are being sought, as it is well known and experimentally observed that different test criteria trigger different types of faults (e.g., [16]). Moreover, fault removal is only one of the potential goals for software testing; other goals can include the evaluation of some specified quality, e.g., reliability testing, usability testing, performance testing, and so on. Therefore, many concur that it is preferable to spend the test budget to apply a combination of diverse techniques than concentrating it on just one, even if shown to be the most effective [8].

4. DISCUSSION

Empirical studies in software engineering should be carried out directly in the “real world”. Attempts to reproduce in academic laboratories realistic settings in which the test techniques are tried are naturally destined to be naive.

As a regular program committee member of software engineering conferences, I happen to be engaged in discussions with colleagues about the value of some submitted paper reporting the results of an empirical study, and in particular, given my expertise, of an empirical study in software testing, and often, more specifically, of an empirical study comparing the effectiveness of software testing techniques. Sometimes
colleagues are skeptical about the little that is gained or learnt after a reported experiment, and raise doubts about both the possibility to generalize the experiment conclusions, and the usefulness of the results to the general community.

I can sympathize with the questions raised about the limitations of laboratory studies, but there is no easy solution. Empirical work is on one side very laborious and time-consuming, but on the other side necessary to advance the maturity level of a discipline. Some hints to better appreciate the difficulties behind empirical work in software engineering are provided in reference [14] and I will not repeat them here.

The various topics proposed for discussion in the WERST website [15] are all sensible questions and constitute an extensive list of the many open issues and weaknesses suffered by empirical research in software testing. Trying to summarize, the three aspects that laboratory studies inherently fail to properly tackle in the experimentation of test techniques are:

1. scale (…of course 😊);
2. context: because clearly one thing is to investigate the performance of a test technique under well controlled conditions and in a sanitized environment, and another is to apply it under the pressure, tight constraints and changing boundary conditions posed by real world development. Another highly debated question is the representativeness of faults in the subject programs, which are often artificially introduced in the form of mutations (but industries are reluctant to distribute real faults, and thus as an alternative to mutants we are left to reuse over and over again the famous Siemens suite [5]);
3. tester’s background and culture: we all know that human factors heavily affect the outcome of empirical software engineering studies, but it is rare the case that a supply of professionals is available to participate to the experiments.

Banally, for all three aspects, the underlying and almost insurmountable barrier is cost: careful empirical studies on large scale products, within real world contexts, and replicated by several professional testers so to attain generally valid results would be of course prohibitively expensive.

A possible way out to overcome such difficult challenges could be that of combining the efforts of several research groups, currently conducting separate experimentations, and join their forces to carry out a widely replicated experiment, i.e., factorize a large experiment in pieces among several laboratories. Roughly the idea would be that of launching sort of an “Open Experiment” initiative, similarly to how some Open Source projects have been successfully conducted. But keeping also in mind that not all open source projects are necessarily successful, and experimentation, to be credible, needs very careful planning and control. Moreover, such an enterprise could perhaps overcome the problems of scale, but the issues of context and tester’s background would further require that industries be actively involved in the initiative.

5. TOWARDS LARGER SCOPE EXPERIMENTATION

As said, the main focus of empirical research in software testing has been so far on the technique to be adopted for the selection of test cases. Indeed, the problem of test cases selection has been the largely dominating topic in software testing literature to the extent that “software testing” is often taken by researchers as a synonymous for “test case selection”. But, the latter is just one piece of the testing puzzle, and after having derived a test suite, many other complex pieces have to be put in place before the picture is complete [1]. Other test related activities present technical and conceptual difficulties that are under-represented in academic research, but well present to practitioners: the ability to launch the selected tests (in a controlled host environment, or worse in the tight target environment of an embedded system); deciding whether the test outcome is acceptable or not (the so-called test oracle); if not, evaluating the impact of the failure and finding its direct cause (the fault) and indirect one (Root Cause Analysis); judging whether the test campaign is sufficient, which in turn would require having at hand effectiveness measures of the tests: one by one all these tasks present though challenges.

Hence, the focus of empirical research should move from just “test selection” to the widest possible scope of evaluating “complete test solutions”. By complete test solutions I mean experimenting also different test oracles, test reporting procedures, repair strategies, etc, and also how different combinations of such test activities may be cost/effective in the testing of complex systems.

A related notion to this ideas is that of “test patterns”. Patterns are a recent emerging discipline within software engineering, earlier adopted by the object-oriented community² to solve software design problems. An often reported definition for a pattern is that it is a proven solution to a problem in a context.

The idea of test patterns is not new [9]. However, they have been proposed so far mainly as sort of qualitative lessons matured by experience and as such offered by the “test experts”. What I am thinking here is a more ambitious idea of empirical studies that rigorously assess the effectiveness of proposed test patterns. More precisely, a pattern is a three-part rule, which expresses a relation between a certain context, a problem, and a solution.

Is not this sort of relations what we mostly lack in software testing? We need to understand how to characterize test problems, to recognize what are the relevant conditions that identify the surrounding context, and finally how to associate a problem and a context with a working solution. All this involves much more than just deciding a criterion for selecting the test cases.

Hence we should methodically design empirical studies to evaluate some complete test solutions to “recurring” and relevant test problems.

6. CONCLUSIONS

I have sketched some green ideas for further empirical research in software testing. These would need further elaboration and design

² Adapting them from the work of architect Alexander in buildings engineering.
before they can be put in practice (the advantage of position papers is in fact that one can fantasize on attractive, even far-fetched ideas, without having also to sustain them with feasibility proofs or evidences.)

I summarize here below the main ideas proposed:

- empirical research could be used to support and possibly substantiate the conclusions reached by analytical comparisons of test techniques.
- to accomplish large scale experimentation a coordinated effort could be organized (in occasions like WERST’04) to factorize a wide experiment among several laboratories (similarly to what is done in open source development efforts).
- empirical research should focus not merely on test selection, but on all aspects and activities of testing: we need to produce quantitative conclusions supporting complete test solutions.

The ultimate goal of empirical research should be to advance the maturity level of software test practice. The existing gap between research and practice can be eliminated only by bridging the theoretical results and the academic tools towards the industrial realities. This bridge is made by measured evidences of the benefits gained by complete test solutions embracing advanced techniques and research results.

7. REFERENCES


