Enabling and Enhancing Collaborations between Software Development Organizations and Independent Test Agencies

James A. Jones
University of California, Irvine
Department of Informatics
jajones@ics.uci.edu

Mark Grechanik
Accenture Technology Labs
mark.grechanik@accenture.com

André van der Hoek
University of California, Irvine
Department of Informatics
andre@ics.uci.edu

Abstract

While the use of independent test agencies is on the rise — currently estimated to be a $25B marketplace — there are a number of challenges to successful collaboration between these agencies and their client software development organizations. These agencies offer independent verification of software, skilled testing experts, and economic advantages that arise from differential global labor rates. However, these benefits are often offset by difficulties in effectively integrating the outsourced testing into software development practice. We conducted extensive discussions with test managers and engineers at software development organizations. This position paper presents the findings of these discussions that identify key difficulties of integrating independent test agencies into software development practice, and it describes our position on how these findings can be addressed.

1 Introduction

Outsourcing has become synonymous to the practice of procuring code development with outside suppliers, who typically reside in different countries [2]. But writing source code is not the only activity that is outsourced in today's global economy—software testing is also increasingly outsourced. In fact, numerous *independent test agencies* (*ITA*s) have emerged in the marketplace for test outsourcing, which is over $25B and growing at an annual rate of over 20%, making it the fastest growing segment of the application services market [1].

While many reasons exist as to why organizations use ITAs to test their software, two reasons are most prevalent. First, test outsourcing is often driven by economic factors, such as when an organization cannot harbor the cost of a full-time staff and tool set, or when it is cheaper to use an independent facility residing in a geographic region with lower labor cost. Second, there frequently is a need to establish additional confidence in software systems' quality by leveraging the expertise of outside test engineers, especially before the software is to be deployed at customer sites. Thus, outsourcing testing to ITAs may be internally driven, by management or other factors, or externally driven, when a customer mandates independent verification of functionality and quality by a separate, trusted third party.

The interaction between a software development organization and an ITA is cyclical and depends on the schedule by which new releases are created. During implementation of a new release, usually little to no interaction takes place. However, once a release candidate is finalized and shipped to the ITA, a period of intense interaction commences. Testing begins, and any potential faults are reported to the development organization, which in turn considers these reports and implements fixes when needed. All the while, new revisions of the software are continuously shipped to the ITA for re-testing, usually until the time that a release is made official and becomes available to customers.

This, not surprisingly, is a process fraught with problems. In an informal, preliminary study of ITAs (described in more detail in Section 2), we found the following issues:

1. Neither the ITA nor the development organization can adequately assess how well the software has been tested.
2. Behavioral changes in the software due to new features are often misclassified as failures.
3. Communication between the testers and the developers is too constrained and inefficient to quickly and adequately address questions and problems.

We also identified two key challenges that the nature of the business relationship between the development organization and the ITA imposes. These challenges prevent more straightforward approaches to address the difficulties faced.

1. Source code cannot be shared with the testers at the ITA. The source is often developed for a customer who does not or cannot allow the ITA to view it.

2. The number of program modifications between releases creates problems of scale. We want to enable developers and testers to be remotely aware of each others' process and progress without overwhelming either with extraneous details.

We are in development of a suite of awareness tools (and the methodologies that they support) called CADET (*C*ollaborative *A*wareness between *DE*velopers and *T*esters). The goals of such a system are to provide: (1) timely visibility of emerging issues, relevant activities, and test progress, (2) advanced predictive functionality to allow testers to pre-plan their work, and (3) direct communication channels from developers to testers, and back.

In this position paper, we outline the difficulties and challenges faced by software development organizations in outsourcing their system-integration testing to independent test agencies. We highlight how these difficulties reduce the effectiveness and efficiency of the testing process thus causing the program to be of lower quality upon delivery to the customer. Finally, we outline the solution that we are currently developing and show how it can address the difficulties and challenges that we found.

2 Preliminary Case Study

Our preliminary case study consists of extensive discussions that we conducted, through a series of dedicated meetings and email queries, with test managers and test engineers at Accenture and IBM, as well as a major insurance company, a US government agency, and a major bank. These organizations have widespread experience with independent testing agencies, both as providers of testing services to other organizations and as clients of test services offered by other organizations. These discussions provided us an overview of the difficulties and challenges involved in the use of ITAs.

2.1 Difficulties Encountered

Throughout our discussions with test managers and test engineers, it became clear that a number of problems recur each time an ITA is involved in a software development project. Although there are many manifestations of these problems, they can be effectively categorized into and exemplified by the following three key questions.

How well is the software being tested? We identified that both the development organization and the ITA have little insight into how thoroughly a software system has been, is being, and should be tested. For the development organization, it is important from both an accountability and a quality point of view. The development organization needs to know whether the ITA indeed performs the rigorous kind of testing to which the two organizations have agreed, and needs to know whether the software that it developed is indeed of the quality that was anticipated. For the ITA, it is equally important from an accountability point of view, to prove to the development organization that it indeed performed the work, and from a management point of view, to be able to judge progress and steer the testing efforts on a day-to-day basis to make them most effective. Currently, the only basis on which development organizations can evaluate the quality of the testing is by assessing the number of faults reported by customers after deployment.

Is a change in behavior intended or a regression? We identified that often features were added, modified, or removed by the developers causing the testers to classify these changes as failures. It is critical for the development organization and ITA to stay synchronized with respect to the evolution of functionality of the software to be tested. In response to modifications in functionality, test cases must similarly be updated.

Unfortunately, testers are not always made aware of such modifications. As a result, testers frequently report failures that in reality represent desired functionality. Especially during the rapid delivery of new revisions of the system in response to previously found faults, new functionality might be added that must now be tested, existing functionality may be modified and need different tests, or existing functionality may be dropped altogether that should no longer be tested. Without proper and timely updates of these changes, test engineers spend a significant amount of time understanding and reporting faults that turn out to be valid features.

When issues or questions arise, with whom do I talk? We identified that software developers do not have adequate communication channels to interact with testers at the ITA. This third issue permeates the first two, and pertains to the need for communication between the development organization and the testing organization. Because an ITA is independent, and often resides in a geographical location several time zones away from a development organization, boundaries arise [3, 4].

On a broader level, it is also more difficult to find the right person to talk to in the first place. Testers do not know who wrote or modified the code. All they know is that they are testing a new revision that has undergone some cumulative set of changes. When a potential fault is detected, they therefore do not know to which developer they should talk. These possible faults, thus, have to be filed in the bug database, and it is a laborious process to prioritize, review,

assign, and document these faults. This process can take many hours, and in some extreme cases even days. A quick phone call or instant message might have avoided this process altogether.

For developers, the situation is somewhat better. Testers are required to put their identity in the bug database, so that a contact is available to developers who must examine a fault and wish to ask some questions from the tester who detected that fault. If developers want to be proactive, however, and give an early warning to the testers as to some upcoming functionality changes, the same problem arises. They do not know whose test cases will be impacted, and broadcasting to the entire set of testers is bound to be ineffective if it becomes regular practice.

2.2 Challenges

Two challenges unique to the nature and operation of ITAs drastically influence our solution space. First, source code cannot be shared under any circumstances, as it is highly proprietary to the development organization and usually is considered the property of the actual client for whom the software is being developed. Any approach addressing this problem must operate at an abstract level that does not reveal important aspects of the underlying source code to the ITA, particularly its individual testers, yet is sufficiently informative for improving the testing process. Finding this balance, as well as the limits this balance places on the efficiency and effectiveness of the testing process, is a key research challenge in our work.

The second challenge is scale: ITAs deal with large and sometimes even ultra-large systems, meaning that abstractions, visualizations, and underlying infrastructure must inherently deal with numerous events of possible interest. Even when appropriately filtered, neither developers nor testers can be placed in a situation that requires them to always deal with each of these events one-by-one, as it must still be possible for them to perform their day-to-day work. It is therefore imperative that we find ways of amalgamating the information that we collect, and find ways of presenting it so that developers and testers are provided with the right level of information at the right time, with the option of drilling down to obtain more detail as needed. Understanding the trade-offs involved in doing this, and the impact of these trade-offs on the efficiency and effectiveness of the testing process, is a second key research challenge of our work.

3 Proposed Solution

In approaching this problem, we start with a set of goals that address the issues found in Section 2 while meeting the requirements of the business relationship between the software development organization and the ITAs.

3.1 Goals

Our overarching goal is to empower individuals, teams, and entire development organizations and ITAs with a precise context for their work and the interactions flowing from this work. We propose an approach for *Collaborative Awareness between DEvelopers and Testers (CADET)*. Its salient features include the following.

1. Set up point-to-point communication channels between (teams of) developers and (teams of) testers such that communication and other forms of information exchange are enabled directly with the persons who are relevant to the task at hand;

2. Leverage the results of past actions from all individuals, whether developers or testers, to support the tasks being faced now, particularly in terms of leveraging the results from previous tests to influence which tests should and should not be run presently;

3. Provide awareness among developers and testers so everyone is informed of the decisions made by others at a time when those decisions become relevant to, or directly affect, one's work;

4. Provide awareness at an organizational level of how the efforts of all individuals involved combine to form an overall process that exhibits certain characteristics, qualities, and capabilities;

5. Provide this functionality without revealing sensitive, proprietary information about the structure, implementation, or design of the software under test; and

6. Provide this functionality in a scalable manner to support many developers and many testers engaged in large or even ultra-large systems.

3.2 Approach

In approaching the problem of more tightly integrating ITAs in the software development effort, we observe that we have three sources of data that we can leverage to build awareness. Changes to the code by developers are a first source, the test cases themselves are a second (static), and the execution traces from running the test cases are a third (dynamic). With this information we are developing CADET with the following five components (depicted in Figure 1):

Instrumentation. While it may not be possible to share source code with the ITA, this does not mean that a binary distribution of the raw source code compiled and linked into a system is the only form in which a system can be sent to an ITA. Instead, we propose to share instrumented binaries that produce additional data for purposes of generating

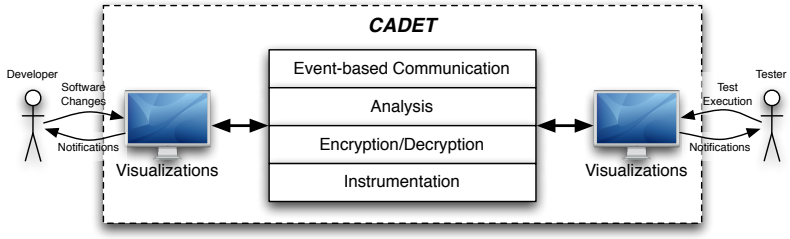

Figure 1. High-level view of the five areas of research for our approach.

awareness information. Naturally, the instrumentation produces data regarding code coverage by test cases, but we will examine other kinds of data as well, such as, paths of execution, input/output values, and stack traces.

Encryption/Decryption. The additional data that is produced is encrypted and stored in a database that can be accessed only after providing an appropriate decryption key. In fact, we propose to instill multiple levels of decryption, each enabling access to incrementally useful, but potentially incrementally revealing, data. For instance, code coverage can be reported in many different ways, from a generic percentage, to percentages assigned to different testers, to percentages assigned to amorphous blocks representing parts of the system, to fully disclosing mappings from test cases to modules.

Analysis. By analyzing code changes with respect to both the static test cases and dynamic traces that are produced, and utilizing any data produced by the developers' running unit tests, we anticipate being able to define heuristics that will form the basis for our awareness visualizations. These heuristics will aim to support both testers, by determining which test cases will likely be invalidated by some set of code changes, and developers, by leveraging failure information at the ITA to give developers early indications of where the faults that causes those failures are likely to be in the source code.

Event-based Communication. All of the information that is thusly collected must be appropriately distributed among the individuals and teams involved in the process, in a timely way yet without overwhelming each constituent party. Moreover, the information forms the basis for the communication paths that must be established to resolve the issues that arise. For this, we will be exploring an event-based communication infrastructure, for which the rules of event delivery, and how those rules automatically change and update themselves in response to emergence and resolution of issues, will be of interest.

Visualization. Once routed to the right person(s), events must still be conveniently presented to them. The visualizations that we will design to do so must be sufficiently informative to enable intelligent decisions to be made, not distract from the core tasks at hand, yet be forceful enough when the issue at hand is critically important and must be dealt with immediately. These factors must be carefully balanced to enable individuals to engage in the overall process as seamlessly as possible.

4 Conclusion

Our preliminary discussions have enabled us to identify an emerging problem domain that is likely to only worsen as independent test agencies are increasingly used by software development organizations. We identified some key difficulties and the challenges that face their resolution. We confirmed that these issues are widespread and cause: (1) great inefficiencies in terms of both time to adequately test and cost to perform such testing, and (2) lower software quality to reach the customers as software faults often go unrevealed by testing due to the inefficiencies and time spent on shallow faults.

We are currently developing methodologies and tools that support them to address these problems. Our solution is a multi-layer approach that combines research in a number of related areas. We will continue to work with the subject companies and agencies in our study to ensure that our solutions can address the issues as well as meet the practical requirements of the business relationships.

References

[1] Application testing services: Global market forecast model. http://www.datamonitor.com/industries/research/?pid=IMTC0147.

[2] W. Aspray, F. Mayades, and M. Vardi. Globalization and offshoring of software. Association for Computing Machinery, 2006.

[3] J. A. Espinosa, N. Nan, and E. Carmel. Do gradations of time zone separation make a difference in performance? A first laboratory study. In *Proceedings of the International Conference on Global Software Engineering*, pages 12–22, 2007.

[4] J. A. Espinosa and C. Pickering. The effect of time separation on coordination processes and outcomes: A case study. In *Proceedings of the Hawaii International Conference on System Sciences*, 2006.