DATAART

A Case for Quality Assurance in Financial Technology Testing

May 23, 2014

Quality assurance testing in a project's early stages is sometimes viewed as an unnecessary cost, but a proper strategy and collaboration can still save money and reputation.

In his article, Closing the Quality Gap: Firms Look To Short-Term Corrective Testing Strategies, Stephen O'Reilly makes a case that, while financial services firms incur significant cost and effort in fixing software defects, they are largely under-investing in testing software before it is put in production. He rightly points out that the cost (both monetary and operational impact) of addressing production bugs is a lot higher than fixing the same issues earlier, yet that is not convincing firms to try addressing those issues upfront. This argument is valid across industries, and it is widely accepted that the cost/impact of software bugs grows exponentially as software products grow from early stages of development to live operation.

Stephen provided quite interesting and trustworthy statistics about willingness to invest in quality assurance (QA), but the question I would like to address is: "Why don't firms do more testing upfront when it is obviously cheaper and less risky?"

There are a few answers that go deeper than the straightforward cost/benefit analysis:

- First, testing on its own is no guarantee of a bug-free product. Anyone making a claim to the contrary is misleading the customer. The argument is not whether to spend a thousand dollars so you don't lose a million later, but rather to spend a thousand dollars so you are less likely to lose a million later. This sort of risk assessment is a very natural way of thinking for financial firms.
- Decision makers often cannot accurately evaluate the quality of the testing process. By its very definition, testing is less visible in terms of tangible results.
- For appropriate testing you need QA engineers with deep understanding of business process and use cases, maybe even deeper knowledge than a development team. This is a very rare skill set on the labor market.
- Testing naturally extends time to market. Even if funds are available, the cost of waiting for the product to hit the market may overweigh any potential risks.

So how does one address these issues?

First, while testing indeed is no guarantee of a bug-free product, a well organized testing process dramatically improves defect localization and fixing process. For a mission-critical system, downtime means money, and a well designed test framework will mean shorter downtime and faster recovery.

The very process of testing focuses one's attention on the blind spots, which are notoriously difficult to predict defects in. This means we can anticipate potential issues better. Whether it is scalability issues or cross-browser support -- depending on how well-tested certain product areas are, any potential issues will be less surprising or dramatic. On the other hand, poor testing or under-testing can be even worse than no testing at all, as it can give you illusion of safety and confidence, while your product still may have a lot of blind spots.

While testing quality is indeed harder to witness/evaluate than product functionality, it should not be underestimated. Professional, well organized QA processes, combined with modern testing and reporting technologies, provide much better visibility now than in years past. Some of the core parameters we track in our QA efforts are functionality testing coverage, code test coverage, and configuration test coverage. Combined with reasonable acceptance criteria, these metrics enhance the team's understanding of the system and efficiency of the development and testing processes.

It is very hard to find QA engineers with deep knowledge of the financial industry, but there still can be some possible solutions. First, you can extend the QA team with a dedicated business analyst. These professions have a lot in common, so such augmentation would be very natural. Next you can organize some industry-oriented courses for quick immersion in industry-specific topics. In my experience, this approach works well, taking into account that one of the key features of a qualified QA engineer is a short learning curve. The best results will be achieved if these two approaches are combined.

Finally, in order to mitigate time-to-market risk, one has to get in the habit of getting QA involved as early as possible in the system lifecycle. Test plans, cases, environment, and automation framework are but a few things that can and should be done often and early. If included from the start, QA adds only about 10% of time to the development timeframe, instead of 20 to 25% that is popularly thought to be inevitable.

Testing can save you money and reputation, but it is not solely about increasing the budget on QA in the early stages. You need to have the right process and the right people at the right time. And even more importantly, you need to make the right decisions based on information provided by testing.