Resistance to performance testing for a big bang rewrite?

Asked 16 years, 1 month ago Modified 12 years, 10 months ago

Viewed 445 times

6

I've been working on a "big bang" rewrite for, literally, over two years. The management has consistently and relentlessly ignored and belittled my calls to allocate time / resources for performance measurement, capacity planning, and optimization before the app replaces their mega-millions money maker flagship web app.

Finally, they have agreed to do it (and we successfully prevented them from big-banging by bringing up a parallel beta server that is in production now and will be the target of the tests). I don't like that they waited until the end to prioritize this, but it's better late than never.

What suggestions does everyone have for dealing with situations like these in the future? What is the best way to educate managers / clients about the need for these kinds of tests.

I've shown them Microsoft's performance guide on CodePlex, complete with its stark warnings from seasoned professionals in the opening pages. I've also shown them the book "Release It!" and the guidance its author gives about "the 3 am call". That has finally convinced them reluctantly, but the truth is that this should have been prioritized into the development and partly measured during development prior to final complete system testing.

Many managers and old-school engineers who wrote ASP only, but never did .NET, are used to coding everything themselves and don't understand all the options for caching, tuning, and health monitoring in newer .NET apps.

Thanks

performance-testing

Share

Improve this question

Follow

asked Nov 8, 2008 at 16:35

Copperfield

5 Answers

Sorted by: Highest score (default)

6

What you didn't realize (and many engineers don't) is that this was a "sales situation", not an engineering one. It doesn't matter if the customer is in-house or not, the process is largely the same.

Sales is all about finding out what kind of problems drive your customers and then showing how your product solves one or more of their problems. If they don't think they have a performance problem, then they don't -- it's that simple. Although you may be able to educate them to the point where they see things your way, "educational selling" is expensive in time and money, and many customers resent being told "something they already know." It sounds like you had to educate this group by beating them over the head with the book, but there may have been easier ways to accomplish your goal.

What would it have been? I don't know, but they do, so ask them. Ask what it was that ultimately pushed them to making the decision. It might have been a sudden realization that you were right, but more likely it was something more basic, like a growing fear of being humiliated in the boardroom or the marketplace. They are unlikely to say so directly, but if you really listen to their answers you may be able to read between the lines. In sales, doing a postmortem on a sales call (successful or not) is critical to understanding what motivates your customer and how you can tune your own skills in presenting ideas.

And, next time, you will know to ask open-ended questions about what your customer wants to achieve, and what his/her problems are now and in the long run. Will it always work? Of course not, but learning to deal with the social side of engineering issues is a valuable skill to acquire.

Share Improve this answer Follow

answered Nov 8, 2008 at 17:24

Peter Rowell

17.7k ● 2 ● 51 ● 65

Excellent points. On the other hand, sometimes I have tried asking, and gotten only vague, evasive answers. That means the decision makers don't know what they want, and are hoping they'll know it when they see it. – Bill Karwin Nov 8, 2008 at 18:58

1 Indeed. User Stories are very valuable for forcing people to be concrete and specific. They say, "We want a website that does <some vague hand wave>," and you say, "Excellent idea! Now, when the user is on this page and they click this box ... what do *you* want to happen?" Etc. – Peter Rowell Nov 8, 2008 at 19:05

5

Get them to agree on solid numbers for what they expect the system to be able to support (number of concurrent users/tasks/etc), then it becomes an obvious part of the development work to make certain the system can meet the requirements.

Share Improve this answer Follow

answered Nov 8, 2008 at 16:49

tloach

8,040 ● 1 ● 35 ● 44

2

Don't discuss this as an open-ended performance tuning and benchmarking process, as that will make older managers concerned that you're on a fishing expedition or gold-plating the system.

Instead, discuss it as a certification exercise. Identify your current traffic levels, add in a safety margin, and explain that your testing is intended to certify that the system will stand up to real life.

You can still do the performance hotspot work; you just need to give the pointy haired bosses comfrt that all of your work is going to tangible business objectives.

Share Improve this answer Follow

answered Nov 8, 2008 at 19:08

Tim Howland
7,970 ● 4 ● 29 ● 46

2

There are all sorts of ways of convincing people - the examples you mention are "invoke higher authority". Most managers, however, would not necessarily be persuaded by technical guidance.

For situations like this, I've used a risk-based approach. For each project, I keep a risk log, identifying the biggest risks to the project, their likelihood, impact, and mitigation options. Often, you can quantify those items - and that allows managers to make a good decision.

At the very start of the re-write, your risk log might have had the following entry:

> Risk: System performance fails to meet user expectations
>
> Likelihood: unknown

> Impact: end users abandon the website due to excessive load times. Project fails.
>
> Cost of impact: $$$whatever your project cost.
>
> Mitigation: fortnightly performance tests.
>
> Mitigation cost: $$$whatever you think it would cost in time and money
>
> Recommendation: run performance test to quantify the risk.

Most managers would be very uncomfortable with a risk whose likelihood is unknown, but whose cost is the failure of the project. On the other hand, you're not asking for a huge commitment - just enough to quantify the risk.

I like to review the risk log regularly with the project stakeholders - at least monthly. I always start with the "high impact/high likelihood" risks, but then move to the "high impact/unknown likelihood" risks. It's also a good idea to distribute meeting notes, recording the stakeholder decisions on each risk. Again, a manager who sees their name attached to a decision to ignore a high-impact risk, in a written record, will think carefully about the decision.

Once you can quantify the risk - by running some performance tests - you can make further risk-based decisions, based on the cost and likelihood of performance problems. This is also a good way to manage the other classic non-functional issues like security, accessibility and scalability.

By quantifying the issue, you turn it into a business decision, not an engineering decision.

Share Improve this answer Follow

edited Jun 20, 2020 at 9:12

Community Bot

1 ● 1

answered Jan 31, 2012 at 13:16

Neville Kuyt

29.6k ● 2 ● 39 ● 52

0

Take careful notes about this development project, including what performance problems crop up after deployment. People will bemoan the problems, and you can tactfully suggest that they prioritize that sort of problem higher earlier. Some people will only accept direct first-person evidence.

Share Improve this answer Follow

answered Nov 8, 2008 at 16:46

Ned Batchelder

375k ● 77 ● 578 ● 673