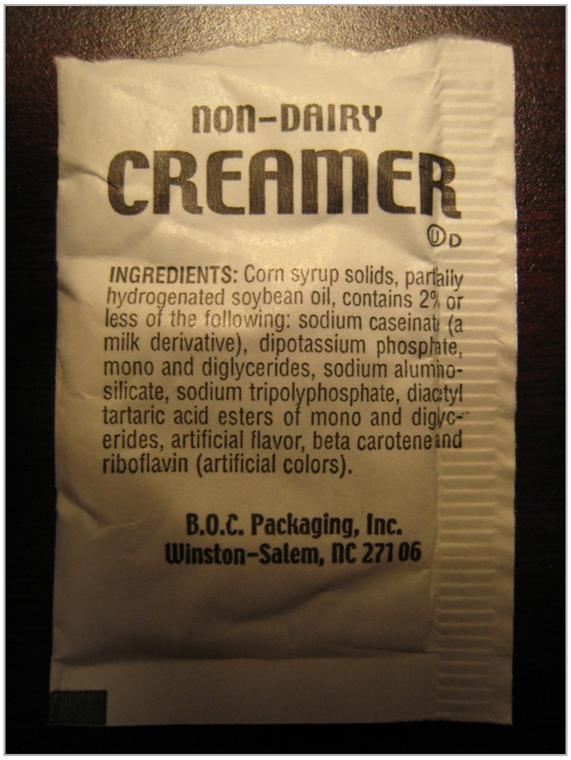
Selling to the Bottom

by Jason Cohen on January 25, 2007

More than a decade before “Product-Led Growth” became a thing, I was selling non-recurring-revenue-software to Fortune 500 companies from the bottom up.


non-DAIRY
CREAMER
Ⓤ D
INGREDIENTS: Corn syrup solids, partially hydrogenated soybean oil, contains 2% or less of the following: sodium caseinate (a milk derivative), dipotassium phosphate, mono and diglycerides, sodium aluminosilicate, sodium tripolyphosphate, diacetyl tartaric acid esters of mono and diglycerides, artificial flavor, beta carotene and riboflavin (artificial colors).
B.O.C. Packaging, Inc.
Winston-Salem, NC 27106

The golden rules of coffee, cream, and sugar:

1. The worse the coffee the more cream it needs.
2. In the limiting case, great coffee needs no cream. (See: Turkish)
3. No coffee is improved by sugar.
4. Not even sugar in the raw.
5. (In the raw? WTF?)
6. Cream is supposed to be made from liquid that comes from cows. Why do I even have to say this? “Non-dairy creamer” is not cream. I don’t write code with a “Non-Java compiler.”
7. Oh really? Read the ingredients and tell me that’s “cream.” Go on, I know the lighting is bad, but see if you recognize any of those ingredients as “food.”

Today we’re pitching our software to the CTO of a public company. This is not the usual route for us.

We usually sell our code review software to software developers who have to review code. (Is that a palindrome?) Put another way, we sell a pain reliever. How?

It’s easy to see why managers like peer code review. It will result in finding and fixing bugs, especially those related to documentation and maintainability and mentoring that no automated tool or unit test can find. They get nice reports they can show to each other “proving” productivity advantages.

But it’s the software developers who bear the burden of this process. They have to gather files and file-differences for review, packaging them and sending them to someone else. They have to paw through comments that start with “On lines 834-837 of //depot/trunk/project/foo/bar.c.” (Of course those lines have shifted around by now.) They have to track review comments over a pile of email across two continents. They have to track the problems-found and bother other developers to fix them. They have to sit there with stop-watches and line-counters to generate those precious metrics.

This is dull. It’s a waste of time. Proof-reading code is common sense, but all this process baggage sucks.

That’s where our tool comes in. Sparing you the features chart, let’s just say we remove this baggage.

Code Collaborator removes pain.

This is really the only reason anyone buys our tool. If it only generated metrics and reports, the managers would like it but developers would (rightly!) revolt and circumvent the system. And then the reports wouldn’t mean anything anyway.

This is a "bottom-up" sales strategy: Genuinely solve a genuine problem for the folks in the trenches, then they beg their bosses to buy the tool to alleviate pain. Sounds reasonable, and even sounds like a positive contribution to the world.

But this next sale is "top-down." Start with the CTO and show why he should push this down everyone else's throats.

Of course we hope it's not that negative. But you always have to be skeptical when the boss comes by and says "Hey we just bought a new tool! Time to do code reviews! I've need those report,s so get cracking!"

Some software tools companies work solely this way. There are ROI charts showing how you'll save $142,245 in the first 9 months of operation. There is the promise of "reports and metrics" (never mind interpretation). Sure the software is $75k just to get started but think of the long-term rewards.

Not us. I don't mind selling to the top, but to me this isn't about some imagined ROI. If you don't believe that proof-reading code will uncover defects, an ROI formula isn't going to convince you otherwise! If you do believe review will uncover defects, the cost of our little tool is clearly trivial compared to the cost of an undiscovered defect—especially if delivered to a customer.

So I sell to the top exactly as I sell to the developer: "This will save your developers time and increase happiness." The sales books say otherwise, but I think you have to stick to the truth and stick by your mates in the trenches. If we're not truly making software better and people's lives better, screw it!

The brass aren't as insensitive as the sales books would have you believe.

Editor's Note from 2024: This was written more than a decade before the world decided that "Product-Led Growth" was a good idea. So, maybe it always was!

Printed from: *A Smart Bear*
https://longform.asmartbear.com/selling-to-the-bottom/
@asmartbear