If It’s Broke…Fix It!

Problem: Long time elapses between customer problem identification and feature fixes caused frustration to customers of a major software firm.

Due to the multiple demands of new release functionality and customer requests for minor tweaks in existing software, the minor tweaks often were back-burnered. This led to a pattern of growing customer dissatisfaction of current clients, while new features were prioritized to attract new, ever more sophisticated clients.

Cause: Competing priorities between Customer Service and Engineering have traditionally provided tension for all software organizations. The siloed organizational structure of most service and Engineering groups adds to the tension, as Customer Service is speaking daily to the customers and Engineering gets most of its priorities from product managers.

Typically software support groups are set up with Level I, II, and III support, with more complex issues working through each level if not resolved at the prior level. If the Level III technicians can’t solve the problem, they report a bug to the Engineering teams, who are responsible for product development. Engineering has its own hierarchy, and usually newer engineers are tasked with bug fixes and feature modifications. These are rolled up for a future release, unless the bug is a security or production issue and requires immediate attention. So it can be months before non-critical bugs are fixed and released to customers. Additionally, customers requesting features are vetted through a similar process, and Engineering has to make trade off between these features and significant new functionality to keep the product competitive. If too much emphasis on major new functionality, then current customers are unhappy. But if there isn’t enough product advancement, it's hard to attract new customers and competitors have a chance to leapfrog.

Solution: Give Customer Service autonomy to fix bugs, and bridge the divide between service and Engineering by creating a career path into Engineering from Services.

The service organization was flattened to two levels, and the technical ability of service people was elevated. Ambitious goals were set to be able to resolve 95% of all calls by the first person to speak with the customer. This was achievable because the support people could write the patches and bug-fixes themselves. Instead of sending a ticket to Engineering after going through three levels of technicians, the bug and the coded solution were sent to Engineering. Engineering would package these up and test them and release interim releases between major releases.

With respect to feature requests, the services division was allowed to specify the priorities of interim releases, so that the large releases could reflect major changes and existing customer requests could be introduced more quickly.

Over time, as the system proved effective, Customer Service became the main feeder for Engineering talent. The people entering Engineering already had great code familiarity, and a good view of how the software was actually used. Eventually, the Services team folded into Engineering, creating a seamless process for the customers and for the company.

Impact - Customer satisfaction improved dramatically. Instead of months, non-critical bugs could take weeks or sometimes days to put into production, and features were released six times a year instead of semi-annually.

This model became part of the sales team's toolkit, as most enterprise customers were very frustrated with the standard industry support model, and the firm could offer something demonstrably different to customers.

Longer term, this model led to a steady pipeline of road-tested, customer-focused engineers. Turnover in the Services department, which is typically 25% in the software industry, dropped to single digits, as career progression became clear for service personnel.

This method was a precursor for the way Saas software is supported, where smaller release cycles happen because firms aren't dependent on the customers to upgrade.