TAIVARA
Technology Innovation

Software Maintenance

You shipped! What's next?

hello@taivara.com

Index

Overview

You've done it! You've built and launched your product. You've found product-market fit, your customers are happy, and all is right in the world. Time to sit back, catch your breath, and relax. Right?

Absolutely!

However, the harsh reality is that software ages poorly. The moment it enters the market it begins to decay, much like the particles that make up our physical world.

The goals of this deck are to help you to save time and money while protecting your digital product investments. Highlights of what is included:

1. Explain code decay and the ecosystem that makes it a reality.
2. Build an understanding of the types of maintenance you will encounter.
3. Provide practical guidance around developing a maintenance schedule for your product.

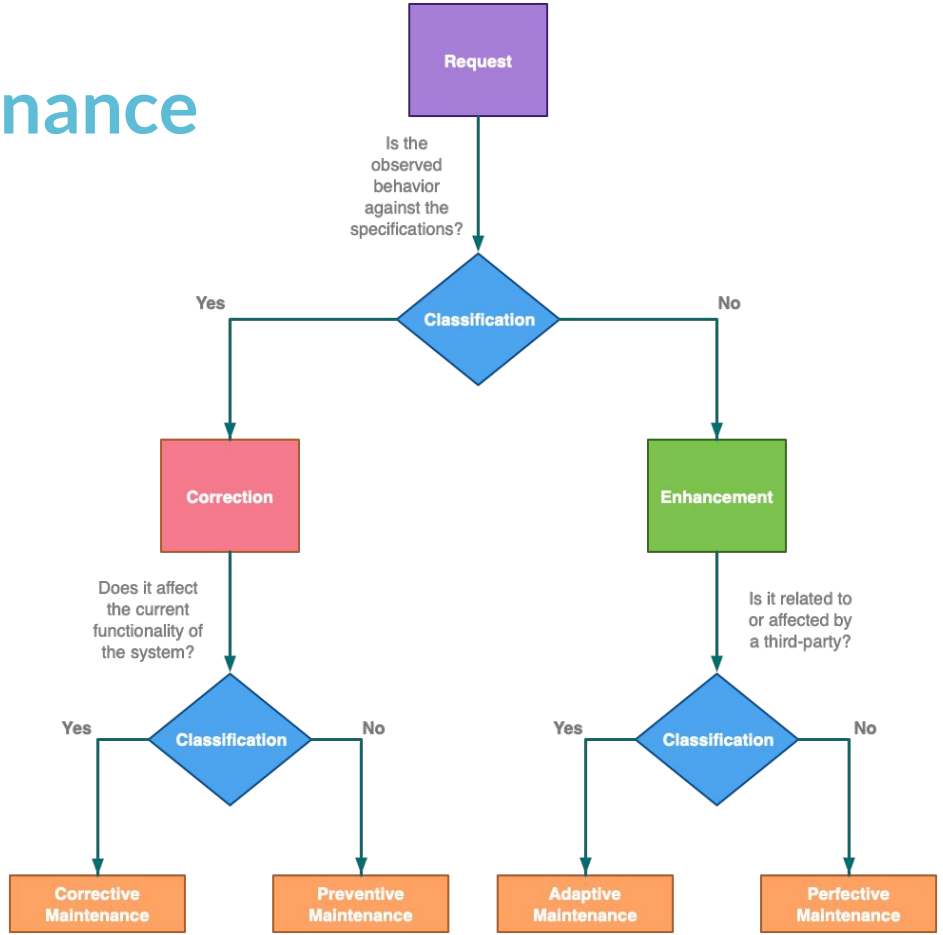
Types of Maintenance

Corrective Maintenance: All software is written to meet certain specifications. When a particular piece of code does not act as specified, we call it a bug! Fixing these bugs can be considered corrective maintenance; the goal here is to make sure software behaves as defined.	**Preventative Maintenance:** This class of maintenance is concerned with preventing code from failing before the failure occurs. For example, when building out an experimental new feature for your product, you likely want to ensure that the market truly desires this new feature before investing too much time. Thus, your initial implementation may be simple or naive to avoid too heavy an up-front cost. Hardening this feature for scale when usage starts to demand it is an example of preventative maintenance.
Adaptive Maintenance: Your software does not exist in a vacuum! Today more than ever, your custom code exists in a living and breathing ecosystem of other software (more on that later). Sometimes this third-party software evolves and requires you to adapt to it. For example, as Apple releases new iPhone devices, mobile apps must often refactor and update their usage of core iOS SDKs.	**Perfective Maintenance:** The behavior of users is notoriously difficult to predict (this is why we always recommend releasing software early and maintaining short feedback loops during development). Paying attention to user behavior patterns and modifying user flows or enhancing application capabilities accordingly is perfective maintenance.

Types of Maintenance

Take a look at the chart on the right for a summary of how to classify your maintenance needs.

Source:
https://medium.com/swlh/types-of-software-maintenance-2b0503848b43



The Software Ecosystem

The software ecosystem moves notoriously quickly. This is why:

- **Hardware Improvements.** Hardware is constantly improving and software must change in concert to properly leverage better hardware. Think Moore's Law.
- **Security Vulnerabilities.** The rise of libraries/frameworks have made developer infinitely more productive; however, this has led to a smaller surface area for hacker to focus efforts on and attack. As a result, keeping up with updates across the stack is more critical than ever.
- **Evolving Best Practices.** Software engineering is a relatively young field. Programming paradigms old and new are constantly battling it out for developer adoption. When a new best practice has proven to be more maintainable, faster, safer, etc, it pays dividends to adopt the new pattern.
- **Platform Dependencies.** For the reasons stated above, the platforms you build on or target will often "require" updates from your codebase. For example, the large Cloud platforms have a vested interest in keeping their servers secure and thus often drop support for older OS/software versions.

Code Decay & Technical Debt

Due to to the nature of the Software ecosystem, code begins to decay the moment it is released into the wild. It can be helpful to conceptualize this decay as an ever-accruing technical debt. Now, this debt is often required and perfectly reasonable to take on (e.g. skipping this quarter's update in order to ship Feature X to meet business goal Y) yet it must be accounted for and paid off responsibly. The consequences of unmanaged code decay & technical debt can be serious:

- **Expensive Upgrades**: Difficulty of upgrades compound over time. Eventually, cost to refactor will exceed cost to rebuild.
- **Expensive New Development**: Cost & time to build new features may increase.
- **Inability to Launch**: Platform requirements may shift and prevent launch.
- **Poor User Experience**: Bugs can confuse and frustrate users.
- **Security Breaches**: Hackers may use known vulnerabilities for out of date software to compromise your system.

Half-Life of Stack

We've established that software maintenance if important. So, how often should you be making updates? The truth is that is varies by component and the specific technologies in play. You should work with technology professionals near you to evaluate your system and build the right schedule for you. However, we have put together a sample template with good rules of thumb.

Component	Maintenance Frequency	Example Technologies
Client Side Web	Every 3 - 6 months	React, Angular, jQuery
Client Side Native	Every 6 months	iOS (Objective-C), Android (Java)
Server Side Frameworks & Tools	Every 6 months	Rails, Django
Server Side System & DBs	Every 12 - 18 months	Ubuntu, PostgreSQL, Ruby, Python

Conclusion

Every piece of software is unique and every organization has different values & constraints. The interplay between these factors needs to be taken into account and will lead to a different “right” maintenance plan for your product. If you’d like help developing yours, get in touch!

However, these recommendations are usually helpful and may serve as a solid, concrete place to start:

- **Write it out** - Make sure that you have a maintenance plan written out and resources allocated by the time your product launches.
- **Stay consistent** - Do not miss consecutive maintenance “blocks”. If you do, you may be setting yourself up for expensive, unplanned software emergencies.
- **Experiment** - Find the right maintenance plan for your organization. You may be able to build maintenance into blocks of feature development or prefer a separate quarterly/biannual process. The right plan is the plan that works for you!

Contact Us

To learn more about software maintenance and support contracts,
or to talk with an expert in this area:

Email us at hello@taivara.com

Call us Today at 614-300-7374